SEIS8S, A LIVE-CODING LANGUAGE FOR LATIN DANCE MUSIC

CULTURALLY SITUATED PROGRAMMING PLATFORMS: SEIS8S, A LIVE-CODING LANGUAGE FOR ELECTRONIC LATIN DANCE MUSIC

BY LUIS NAVARRO DEL ANGEL, B.A., M.A

A Thesis Submitted to the School of Graduate Studies in Partial Fulfilment of the Requirements for the Degree Doctor of Philosophy

McMaster University DOCTOR OF PHILOSOPHY (2023) Hamilton, Ontario (Communication Studies and Media Arts)

TITLE: Culturally situated programming platforms: Seis8s, a live-coding language for electronic Latin dance music

AUTHOR: Luis Navarro Del Angel, B.A. (Academia de Música Fermatta), M.A. (McMaster University)

SUPERVISOR: Dr. David Ogborn

NUMBER OF PAGES: viii, 365

Abstract

This dissertation speculates on culture, social spheres, and programming to gain insight into how computer platforms can be (re)thought and (re)designed around the consciousness and struggles of Latin American communities. This dissertation uses an interdisciplinary methodology emphasizing approaches to live-coding performance, platform design and software development, participatory action research, and interpretive and semiotic analysis.

The research in this dissertation starts with the argument that computer languages are influenced by social spheres (e.g., science and arts), economic models (e.g., knowledge economy), communication systems (e.g., natural language), and infrastructures (e.g., software collaboration protocols and institutions). Next, it is discussed how I deployed this argument by ideating and coding a computer language based on specific social spheres (i.e., live coding practice and popular music), communication systems (i.e., Spanish), and infrastructures (i.e., cultural centers and online spaces) of Hispanoamerica. Finally, this computer-music language is compared and contrasted against collective reflections and uses by this dissertation's author and members of the general public through a series of conversation circles and live coding performances.

This research results in developing Seis8s, a computer-music language inflected by Spanish constructs borrowed from Latin dance music. Seis8s blends Latin American music sensibilities and live coding techno-politics to promote critical reflection. Seis8s emphasizes resistance to asymmetric types of computer-music abstraction by bringing Afro-Latin instruments and rhythms to the center of the technology and the performance. Results of this

research also include ten public presentations using Seis8s involving individual and collective live coding performances and conference presentations. These public presentations showcased Seis8s and promoted reflection toward universal understandings of bodies, culture, politics, and economies of these technological and artistic milieus.

This research also gives insight into mestizaje and latinidad as concepts still present in the belief systems and ways of knowing Spanish speakers in Latin America and, consequently, in the software they develop. Mestizaje and latinidad are challenged collectively by positioning the white-mestizo ideology as a shared problem that could be resisted through reflection on the irreducibility of the Latin American identity and its potential to coexist with other identities.

Acknowledgements

I thank my supervisor Dr. David Ogborn and my supervisory committee members, Dr. Christina Baade and Dr. Rossana Lara, for their insightful feedback on this dissertation and continued support throughout my doctoral studies. Thank you for being an inspiration inside and outside the university through your kindness and patience.

Thank you to the Communication and Media Arts Department at McMaster University for supporting this research through the departmental doctoral scholarship. Thank you to Canada's Social Sciences and Humanities Research Council for supporting this research through my supervisor's grant, "Platforms and practices for networked, language-neutral live coding". Thank you to the Mexican Fund for Culture and the Arts (FONCA) and the Mexican Council for Science and Technology (CONACYT) for supporting this research through the grant "Doctoral Scholarship for Studies Abroad 2019".

Thank you to the Communication and Media Arts Department faculty at McMaster University for their meaningful teaching. Thank you to McMaster professors Dr. Andrea Zeffiro, Dr. David Harris Smith, and Dr. Sara Bannerman for their ongoing mentorship. Thank you to the department administrators, Cassandra Weimann and Lorraine Bell, who guided me through the operative aspects of the university.

Thank you to my parents, Fany Del Angel and José Luis Navarro, for encouraging me to pursue this doctoral degree and for their emotional and economic support. Thank you to my partner Jessica Rodríguez for all the insightful discussions and feedback about this dissertation.

Thank you to all the participants and interviewees of this dissertation. Some of the interviewees who permitted me to include their names are Tania Alejandra, Gabriel G. aka alom, Jessica Rodríguez, Andres Miramontes, Laura Zapata, Milagros Saldarriaga, Marco Valdivia, Patricia Cadavid, and Dr. Irene Soria. Thank you to the Factory Media Centre in Hamilton, Ontario, for their support in promoting this research. Thank you to the Factory Media Centre's Operations Coordinator, Kristina Durka, for facilitating the connection with participants and interviewees of this dissertation.

Thank you to Emilio Ocelotl and Marianne Teixido, members of the collective RGGTRN, who were a source of inspiration through the many performances and workshops we did together before and during my Ph.D. project. Thank you to my after-school music teacher Gerardo Lopez Silva who encouraged me to explore all types of music, including cumbia.

Finally, I would like to provide a list of Latin dance music that inspired me while writing this dissertation.

Los Ángeles Azules - Cumbia del Acordeón ft. Los Claxons • Los Ángeles Azules - Viernes Cultural ft. Haydée Milanés • Joe Arroyo - La Rebelion • Megapuesta - Blinding Lights • Celso Piña - Reina de Cumbias • Los Mirlos - La Danza de Los Mirlos / Cumbia de Los Pajaritos • Raymix - Oye Mujer • Los Van Van - Soy Todo • Los Van Van - Aquí el que Baila Gana • Juan Formell y Los Van Van - Soy Todo / Ay Dios Ampárame • Willie Colon and Hector Lavoe - Aguanile • Elito Revé y su Charangón - La Rumba feat Paulo FG • La Delio Valdez - La Cancioncita • La Delio Valdez - De noche al amanecer • La Delio Valdez - Joyita Norteña • Amantes del futuro - Fanatasía Olmeca • Grupo Carabo - El Afilador • Grupo Soñador - El paso del gigante • Grupo Jalado De Oscar Bakano - Que Te Vaya Bien • Kumbia Queers - Chica de Calendario • Porfi Baloa - Persona ideal • Juan Luis Guerra 4.40 - Vale la Pena • Jerry Rivera - Amores Como el Nuestro • Oscar D'Leon - Llorarás • Septeto Acarey - Eres Mi Sueño • Los Ángeles Azules - Cómo Te Voy A Olvidar • Celso Pina - Cumbia Sampuesana • Selena - Como La Flor • Aniceto Molina - Cumbia Sampuesana • Sonora Dinamita - Que nadie sepa mi sufrir • Irakere - El volcán del caribe

Contents

Introduction 1
- Anecdotes on culture and computer languages . . . 2
- Questions on culture and situated knowledge . . . 6
- Questions concerning this dissertation . . . 8
- Description of chapters . . . 11

1 State of the art: programming languages as texts and cultural objects 15
- 1.1 Theoretical framework . . . 16
- 1.2 Logo . . . 18
- 1.3 Cornrow Curves and Rhythm Wheels software . . . 21
- 1.4 Lenguaje Latino . . . 25
- 1.5 Alb . . . 30
- 1.6 Conclusion . . . 33

2 Seis8s, a live-coding language for electronic Latin dance music 35
- 2.1 Motivation . . . 36
- 2.2 Background: A personal anecdote on postmodernity and live coding in Mexico 38
- 2.3 Introducing Seis8s . . . 43
  - 2.3.1 Lexicon . . . 43
  - 2.3.2 Syntax . . . 45
  - 2.3.3 Semantics . . . 46
- 2.4 Seis8s as text and cultural object . . . 52
- 2.5 Conclusion . . . 57

3 Seis8s in context: talking circles on music and software 59
- 3.1 Description of the circles . . . 60
- 3.2 Methodology for gathering and analyzing participants' contributions . . . 62
- 3.3 Part 1: Software in Latin America . . . 64
  - 3.3.1 What is software? . . . 64
  - 3.3.2 Parallel coexistence, delinking digital technology . . . 66
  - 3.3.3 The concepts of race, multiculturalism, and pluriculturalism in software 71
  - 3.3.4 On software orientations . . . 76
- 3.4 Part 2: Latin dance music . . . 79
  - 3.4.1 Cultural appropriation in Latin music . . . 79
  - 3.4.2 Sophisticated listening and instrument substitutions . . . 84

3.4.3 Kumbia Queer, commands, lyrics, and dance . . . 87
3.5 Conclusion . . . 91

4 Seis8s in practice: public performances and exhibitions 96
4.1 Seis8s usage aesthetics: Live Coding and Networked Music performance . . . 97
4.2 Performances and Installations with Seis8s . . . 98
4.2.1 *D'Binis at Campamento Extendido <impendingvoid> by Posternura Records* . . . 98
4.2.2 *RGGTRN at Digitropics* . . . 100
4.2.3 *Luis Navarro Del Angel at* Galvanized Suns *by Diasporic Futurisms* . 103
4.2.4 *Tania Alejandra, Gabriel G. aka alom, Jessica Rodríguez, Andres Miramontes, and Luis N. Del Angel: Rehearsal for a Performance at the Factory Media Centre* . . . 106
4.2.5 *Cybernetic Orchestra at Estuary's 5th Anniversary by the NIL* . . . 108
4.2.6 *The ASCIItos: Barush, Nuriban and Gabriel G at Estuary's 5th Anniversary by the NIL* . . . 111
4.2.7 *Grupo D'Binis at PUMPUMYACHKAN Festival 16th by Asimtria* . . 114
4.2.8 *Seis8s as an online Installation at International Conference on Live Coding* . . . 117
4.2.9 *Genuary 2022 Seis8s In Space* by Illest Preacha . . . 119
4.2.10 *Seis8s as an online Installation at the New Interfaces for Musical Expression Conference* . . . 120
4.3 Conclusion and further work . . . 122

5 General conclusions 125
5.1 General findings and main takeaways . . . 125
5.2 Reflections on the research process . . . 127
5.3 Contributions to computer programming and live coding practice . . . 128

References 129

A Appendices: Talking circles, list of materials and themes 140
A.1 Thursday sessions . . . 140
A.2 Saturday sessions . . . 144
A.3 Categories and overarching themes . . . 146
A.4 Sei8s' code . . . 160
A.4.1 The Main module . . . 160
A.4.2 The Program module . . . 183
A.4.3 The Parser module . . . 183
A.4.4 The Layer module . . . 318
A.4.5 The Global Material module . . . 334
A.4.6 The Style module . . . 335
A.4.7 The Rhythm module . . . 348
A.4.8 The Harmony module . . . 349

Introduction

This dissertation speculates on culture, social spheres, and programming to gain insight into how computer platforms, such as computer languages, can be (re)thought and (re)designed around the consciousness and struggles of Latin American communities. This dissertation uses an interdisciplinary methodology emphasizing approaches to live-coding performance, platform design and software development, participatory action research, and interpretive and semiotic analysis. The research in this dissertation starts with the argument that computer languages are influenced by social spheres, economic models, communication systems, and infrastructures.

This dissertation heavily discusses programming languages from a Spanish-speaking perspective; nonetheless, it tries to understand Latin America not as a homogenizing appellation of origin but as the label of political movements resisting violence from colonial and imperial power during the 20th century onwards (see Agurto, 2022; Bard Wigdor and Artazo, 2017; Quijano; 2006). I identify myself as part of these political movements and communities of resistance as a subject trying to push further the liberatory notions often attributed to digital arts in favor of women and minority groups of Latin American descent. Nonetheless, I am aware that being a Mexican, mestizo, settler, middle-class, heterosexual male modulates my political action.

Anecdotes on culture and computer languages

The Flow-Matic: American inventories and respectable computers

At the 1977 History of Programming Languages Conference held in LA, Admiral Grace Murray Hopper explains how in the mid-1950s, computer programming took a clerical orientation focused on a type of programmer dealing with data processing. This new programmer was business trained and often lacked mathematical knowledge to code. Commissioned by the Radio Corporation of America (RCA), a team led by her sat down to examine the jargon at the workplace of this new programmer — for example when doing inventories. This way, her team drew up a list of 30 verbs that later became a new computer notation based on English-speaking business statements called the Flow-Matic (Hopper, 1981).

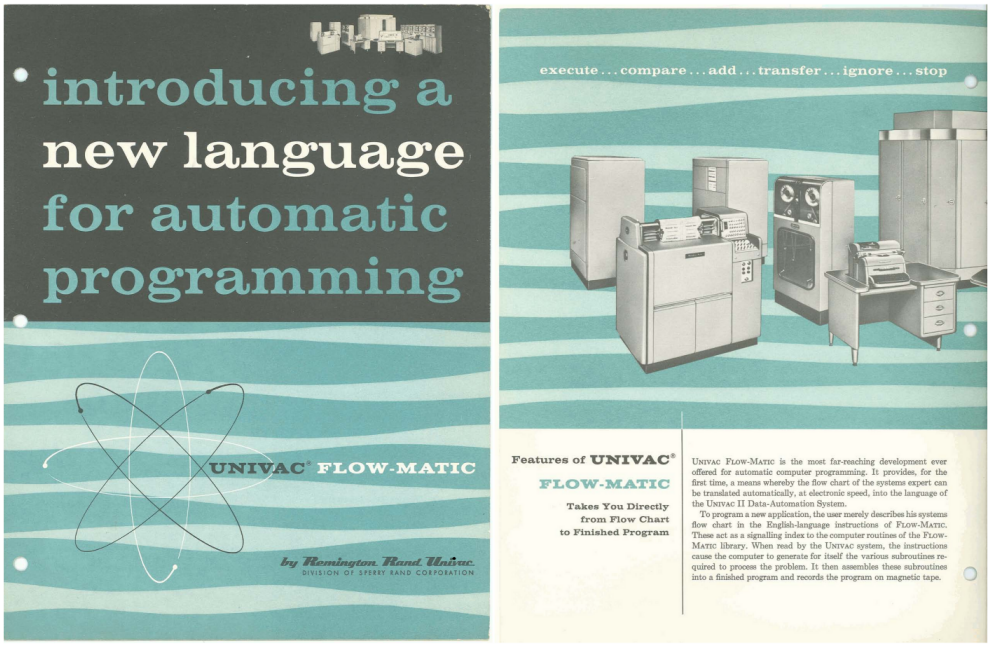


Figure 1: Cover and page 1 of the Flow-Matic user manual (Sperry Rand Corporation, 1957)

As Hopper continues her narration, she mentions that following a logic of translation from mathematical to natural language, it was possible to generate code not only in English but in other languages as well. To make the latter explicit, her team created some commands

in French and German along with the ones in English. Surprisingly enough, Hopper mentions that after presenting these to the RCA directors, she and her team had to spend four months apologizing for programming in languages other than English. As for these directors, "[i]t was absolutely obvious that a respectable American computer, built in Philadelphia, could not possibly understand French or German!" (Hopper, 1981, p. 16)

COMIC: Spanish-derived functions amidst a coup d'etat in Argentina

Between 1964-1966 in Argentina, mathematician Wilfred Oscar Durán Salvador developed one of the first programming languages in the region called COMIC, an acronym for "Compilador del Instituto de Cálculo [Compiler of the Calculus Institute]". It was developed for a Mercury computer bought by the University of Buenos Aires from British manufacturer Ferranti Ltd. One of the main motivations for developing COMIC was finding a more friendly way to express the mathematical ideas of Argentinian scientists of the time.

Strategies that Durán Salvador used to make the language friendly, included Spanish-derived functions, for example, the function *azar*, which returned a random number between 0 and 1 – the word *azar* translates to *random* in English. COMIC also "introduced control structures that took years to appear in other programming languages, such as mark tables, a primitive version of 'case' [expressions]" (Durán Salvador, 2009, p. 9).

```
PROGRAMA = 21        ← identifica esta parte del programa
I = 1
21) ΨLEER (A)
ΨLEER (B)
ΨLEER (C)
£(4) = (£(1)=(=A=0))*(£(2)=(=B=0))*(£(3)=(=C=0))      ← £(4)
verdad si leyó tres 0
ΨBIFURCAR (£(4)) 31
```

Figure 2: A COMIC program with commands in Spanish for solving a series of equations (Durán Salvador, 2009, p. 101)

However, one of the most interesting things about this language is the Argentinian sociopolitical context in which it was developed: the period of dictatorship known as *Revolución Argentina [Argentinian Revolution]* that lasted from 1966 to 1973. COMIC became usable

in May 1966, when the *coup d'etat* happened, as its creator Durán Salvador (2009) narrates.

> *"I still had the penultimate COMIC master tape with me, dated 5-2-66, from the blue color reserved for compilers, in which two pieces of red tape are tied together, from small changes that were made later, as we continued making touch-ups and preparing larger aggregates. When President Illia was overthrown on June 28, 1966, the vigil began" (p. 46, translation mine).*

A goal of the dictatorship was to forbidding the political movement of *Peronismo*[1] and its so-called communist ideology.

> "Friday, July 29, 1966, was exam day. I left the Faculty early in the afternoon because, at seven at night, I had to take a train to Tandil, where that year, I taught a couple of courses on Saturdays at the University of Tandil ... In the morning, when I had breakfast and read La Prensa newspaper, I learned about the bloody intervention of the dictatorship's police. I tried to contact someone from the [institute] by phone, and it took me a while to do so since everyone was meeting somewhere to orchestrate their resignation, to which I immediately included myself" (p. 47, translation mine).

One of the effects of this dictatorship was brain drain, which is a massive flight of highly trained individuals. Because the University of Buenos Aires did not copyright the language, Durán Salvador was able to take it with him to a Venezuelan university, where he continued developing it.

> "And so, on February 22, [1967] we arrived in Venezuela, my wife, our son five years old, and me; the delay was due to the delay of the university administration

[1] "Peronism, also called justicialism, is an Argentine political movement based on the ideas and legacy of Argentine ruler Juan Perón (1895–1974). It has been an influential movement in 20th and 21st century Argentine politics" ("Peronism"., 2022, para. 1). "The pillars of the Peronist ideal, known as the 'three flags', are social justice, economic independence, and political sovereignty. Peronism can be described as a third position ideology as it rejects both capitalism and communism" (para. 7)

in obtaining the visas and sending me the tickets for the trip, a typical tropical way of working" (p. 49, translation mine).

What do these two stories say about the cultural aspects of computer languages?

The anecdotes above point to some cultural aspects and social spheres influencing computer languages, opening the potential to ask about what other cultural aspects influence these technologies and about the implications of this. Hopper's account, for example, highlights a shift from commands based on mathematical constructs to commands based on natural language in the 1950s. Her anecdote also unveils institutional attitudes and modes of knowing that could be interpreted as part of a historical moment in the early-1950s United States focused on the "widespread adoption of modern education and cultural values" under the 1949 Truman Doctrine (Escobar, 1995, p. 4). This way, it could have been that the RCA directives were displeased because it was at this moment that the American people understood English as the language of business, modernity, and truth inside and outside the US[2].

The second anecdote allows one to envision cultural analyses applied to programming technologies developed outside the US, for example, in Latin America. Such analysis is essential to see that speculation on the relationship between culture, social spheres, and programming in Latin America is not new, and could lead to understanding how natural languages spoken in Latin America influenced these technologies throughout history. It could also lead to knowing more about national ideologies and historical circumstances concerning these technologies. Durán Salvador's anecdote, for example, unveils some of the effects of Argentina's dictatorships on the development of computer programming in that country. Moreover, it makes one wonder whether there has been a connection between computer programming and *peronismo* – a political movement said to be ingrained in the mind and customs of Argentinians?

[2]This analysis of the Flow-Matic is of my own research; however, another analysis of this languages is discussed in a recent article from Mark Marino (2020b).

Questions on culture and situated knowledge

In the *Dictionary of Latin American studies* (Szurmuk et al., 2009), Cuban literary critic Nara Araújo discusses the term *culture*. From a Latin etymology, "this word is associated with the action of cultivation or of practicing something" (p. 71, translation mine). It is also related to the action of "honouring ... a religious deity, the body or the spirit". These two definitions conflated position *culture* as "the result or effect of cultivating human knowledge". And, on a larger scale, "as the set of modes of life, and of customs of a time period, or a social group" (p. 71, translation mine).

The Marxist cultural critic Raymond Williams (1977) follows this train of thought when describing the 18th-century understanding of *culture* as a "noun of process", meaning "the culture of something - crops, animals, minds" (p. 14). Towards the 19th century, however, *culture* became redefined as "a process of 'inner' or 'spiritual' [development]" (p. 13). And, it is at this moment when *culture* became "a general classification of 'the arts', religion, and the institutions and practices of meanings and values" (pp. 13-14). Further into the 19th century, "the religious emphasis weakened", and was replaced by the "secular forms [of] 'subjectivity', 'the imagination', and in these terms 'the individual'" (p. 15).

In this sense, *Culture* is understood as "a network of signs; a communicative act, a supposedly constant exchange ... necessary in the relationship between the sender and the receiver" (Szurmuk et al., 2009, pp. 72-73, translation mine). From this perspective, *culture* is limited by the boundaries (e.g. cultural, economic, geographic) that shape the sender and receiver. And as a result, the actions of naming, labelling, and thus *constructing* knowledge are constrained too (see Hall, 1973). Furthermore, these boundaries are precisely what makes it difficult to define the term *culture*.

It "can be understood as a dimension and expression of human life, through symbols and artifacts; as the ground of production, circulation, and consumption of signs; and as a praxis that is articulated in a given theory" (Szurmuk et al., 2009, p. 72, translation mine). Nowadays, "one can talk about urban culture, media culture, popular culture, mass culture,

[and] literate culture", to name a few (p. 72). And, according to Araújo, mass culture, and media culture are two categories of great interest within Latin America since the 1950s. This is due to increased circulation, often connected to dominant economic models and values, and which creates patterns within day-to-day areas of life, for example in "fashion, sports, food, the arts, literature, and taste" (p. 72, translation mine).

The latter is not particular to Latin America, nevertheless. French sociologist Pierre Bordieu (1984), for example, describes how cultural practices and preferences are linked to social and socialization practices. According to him, this socialization not only provides societies with the "capacity to produce classifiable practices and works" but also guides the individual's "capacity to differentiate and appreciate practices and products (taste)" (p. 170). From this perspective, *culture*, takes upon the role of a *structure* or *series of structures* that condition a given society.

This conditioning, as a *constitutive* aspect, has been described by Jamaican-British cultural theorist Stuart Hall (1958), who suggests a cyclical process where not only economy but also culture serve as a 'base' structure that shapes the "consciousness of working people" (p. 32). Hall points, for example, to the continuous technological advancement in the workplace, which in turn "demands a higher level of culture, education and consciousness on the part of a skilled labour force" (p. 35). Demands that are at the same time, modulated by "new possibilities for consumption" promising a higher social status and lifestyle (p. 36). This way, "culture, education and learning" can also be seen as "commodities" that have "a social value in a hierarchy of status symbols" (p. 38).

Latin American studies of *culture* integrate the above with a particular focus on the relation of *culture* as an "agent of resistance ... for its social action in times of globalization, neoliberalism, and electronic media technologies" (Szurmuk et al., 2009, p. 73, translation mine). Within this perspective, *culture* "is understood as entangled with the social, in the transformation of popular culture and the cultural industry, as well as intersecting in social discourses and symbolic processes, power formations, and construction of subjectivities:

gender, race, citizenship" (p. 73, translation mine).

Argentinian cultural scholar Nestor García Canclini (2005) exemplify the above through the following shared questions in Latin American Cultural studies: What are the ways in which Latin American "workers reformulate their work culture in the face of new productive technologies without abandoning old beliefs"? What are the ways in which inhabitants of this region "make sense - personally and collectively - [about] invest[ing] in extended studies in order to end up in low-paying jobs"? And, what are the ways in which popular movements appropriate technology to call attention to their issues and demands? (1-2).

García Canclini goes on by saying

> "We have, then, three questions at issue. How to study the hybrid cultures that constitute modernity and give it its specific profile in Latin America. Next, to reunite the partial knowledges of the disciplines that are concerned with culture in order to see if it is possible to develop a more plausible interpretation of the contradictions and the failures of our modernization. And third, what to do—when modernity has become a polemical or suspect project—with this mixture of heterogeneous memory and truncated innovations" (p. 3).

Questions concerning this dissertation

Borrowing from the questions above, this dissertation asks: What are some specific modes of consciousness – i.e., sensibilities, moral systems, belief systems, modes of knowing and being – of inhabitants of Latin America? What are their struggles – i.e., generalized difficulties and their modes of resistance? In which ways do these modes of consciousness and struggles intersect with music software? How can music software be (re)thought and (re)designed around these modes of consciousness and struggles? What would be the implications of this (re)design?

Self-reflexive questions

In this dissertation, among the broad questions I propose above, I also propose more individual explorations and speculations based on my personal experience as a person belonging to Latin America. An experience that is, however, modulated by borders delimiting me as a Mexican, middle-class, heterosexual, and male settler currently living in Canada. These personal explorations and speculations include, for example, the ways in which the Mexican *mestizo* ideology with which I grew up modulated the development of the computer language I present in Chapter 2 of this dissertation, Seis8s.

This dissertation also deals with issues I have experienced which are related to interdisciplinary practice. For example, how to make sense of things that seem so apart from each other and at the same time are deeply conflated? That is, when working with culture, music, and programming, how can talk one about issues that seem particular to one discipline but not to other, and still maintain focus on the broader picture? On this, various authors agree that there exists uncertainty in defining and categorizing media technologies and practices, as they lay in between human and exact sciences (see Elsaesser, 2006; Adler, 2016). For some, this positioning of media not only prevents the media itself but also the artist from being revolutionary in terms of narrative and technological development. That is, *Jack of all trades, master of none.*

Another perspective, however, could be that it is actually the possibility of being in between human and hard sciences that is revolutionary. For the interdisciplinary scholar Beatriz da Costa (2008), it is the artist engaged with technology and science who has become the link between art, audience, and science. In this sense, the new media artist has the potential of being a political and disruptive "context provider" within the "ever-expanding information society" (p. 367). They make art and science accessible and at the same time, they are aware of the different social domains they are embedded in.

This dissertation subscribes to a larger degree to the latter perspective as it gives a positive note in regard to interdisciplinary practices. However, it is still important to consider

the difficulty of finding the target of the investigation. That is to say, for the *new media artist*, it could be very easy to get distracted and lean towards one or other practice. For example, it is often the case that the artist-researcher leans more toward technological development and/or technological display. In other cases, they lean more toward the production of the artwork. And, in other cases, they lean more toward the theoretical and/or descriptive aspects.

A question could then be if the *new media artist* needs to find a balance between all these possibilities or if leaning toward some and not others is just fine. This dissertation leans towards finding a balance, trying to look into different ways to approach the study of culture, music, and programming on the same level. This approach, however, comes from the possibility that my doctoral program allowed me, in that it is an interdisciplinary program in Communication, New Media, and Cultural studies. This possibility was also enabled by Dr. David Ogborn, Dr. Christina Baade, and Dr. Rossana Lara who are members of my Supervisory Committee and who are interdisciplinary artists and researchers themselves. Thus, throughout this dissertation, I investigate the communicative aspects of computer languages, and how music is or is not part of these. I also entertain the idea of creating a computer-music language and I investigate the uses of this language through live performance. Finally, I investigate how specific cultures understand and make sense of the aspects investigated themselves.

I am not claiming in any way, that this dissertation offers complete answers or methodologies, but the opposite. My intention is for it to open questions that someone else could pick up and explore further. For this to happen, I would like to lay out the key issues of each chapter, making sure to name the methodologies and authors used.

Description of chapters

Chapter 1

This chapter intends to add to current investigations around culture and software (see Galloway, 2006; Chun, 2011; Allen, 2012; Lialina and Espenschied, 2015; Ali, 2016; McPherson, 2018; Eubanks, 2018). The latter, by investigating cultural traits in programming languages through interpretative and semiotic analysis. The languages analyzed are Logo, Cornrow Curves and Rhythm Wheels Software, Lenguaje Latino, and Alb. Initial findings include these languages having strong influence from linguistic, pedagogical, artistic, and collaborative approaches but also an orientation toward the labor force within the global economy.

This chapter draws upon Critical code studies (CCS) which is an interpretive framework concerned with the task of *reading* code. That is, looking at code as "a form of communication with layers of meaning for its wide and varied audiences" (Marino, 2018, p. 472). CCS applies hermeneutics to investigate the interaction between various systems and "frames code as a *cultural text*" by "assum[ing] its significance arises from its history, circulation, and reception" (pp. 472-473). For CCS "[t]o treat code as a cultural text is to examine its accrual of significance through the way it is regarded as well as the ways people respond to and interact with it" (p. 473). CCS draws upon "media archeology and science and technology studies" and "follows in the footsteps of critical race studies and critical legal studies in its calls for reflexive interpretation" to investigate "issues of ethics, social relations, ideology, or other aspects of epistemology and ontology" (p. 474).

Chapter 2

This chapter discusses Seis8s, a computer language designed by me, that allows real-time interaction with digital audio and localized musical knowledge, particularly of Latin American music – also known as urban Latin music or Latin popular music. This is a 20th-century derivation of music "based on Afro-Cuban [and Afro-Caribbean] rhythms, as developed and

performed throughout the Hispanic Caribbean basin and its diaspora" and which is "designed for accompanying social dancing" (Manuel, 1998, p. 127). At the moment of writing, Seis8s revolves around Mexican cumbia and Salsa. Seis8s is also an art piece that invites us to reflect on how culture gets embedded into computing technologies. Seis8s runs in the web and was implemented using Haskell, the Reflex functional reactive programming platform, and WebDirt.

This chapter as well as the development of Seis8s, draws upon the fields of New Interfaces for Musical Expression (NIME) and of Latin American musicology. NIME analyses how technology affects the performer's creative processes and the audience's experiences by exploring human-computer interface design, development, testing, and evaluation (Lyons and Fels, 2014). This field sees interface building as a method for learning through problem-solving, embodiment, and kinesthetics, something that invites one to reflect on the material position of the interface itself (see Drucker, 2013).

Latin American musicology involves the study of musics particular to this region in connection with social, historical, cultural, and artistic processes to understand the surrounding politics, economies, bodies, and power relations. In its critical form, it seeks to raise awareness of the paradigms of being Latino that are constructed from national and global locations (González, 2009). And it draws upon decolonial research and subaltern studies to defy the "distinctive discursive faces of modernity" embodied in how musical knowledge is constructed – e.g., the modes of musical institutionalization and professionalization that are distanced from "cultural traditions and from the expressive and socio-affective needs of people" in this region (Shifres and Rosaba-Coto, 2017, 86).

Chapter 3

This chapter reports on a series of online talking circles, held from August to December 2020, with people from Latin America distributed in different countries to discuss sociocultural aspects of software and music. During these meetings, we discussed how software and

music relate to issues of gender, ethnicity, and social class in Latin America. Through these discussions, participants were encouraged to raise their own questions and to discuss issues revolving around their specific concerns and localities. This way, participants used this space as a platform potentially situated within their context.

The methodology behind these conversations drew upon the frameworks of Participatory Action research and Border Thinking. Participatory action research (PAR) involves "plac[ing] researchers in the service of community members" to "address the imbalances that hinder our world" by "combining knowledge and action" (Elliott, 2013, p. 2). To address this imbalance, PAR looks for approaches that are "situated" in the individual locale and that are aligned with "social justice", "participatory democracy", and with the "critique of socio-scientific colonialism" (pp. 5, 20).

Border thinking deals with the ambiguity of the colonial subject who finds themself operating between modes of indifference/complicity and resistance/innocence (see Anzaldúa, 1987). Border thinking is also a form of activism that invites people to subvert exploitative structures by proposing possible futures that "transcend the imperial monologue established by the European-centric modernity" (Grosfoguel, 2011, 26). Border thinking is aligned with reflections from diasporic thinking (Hall, 2015; Rizvi, 2015), Critical Race Theory (Collins, 2015), and Latin American feminist thought (Bard Wigdor and Artazo, 2017; Di Próspero, Di Próspero, 2016) to mention a few.

Chapter 4

This chapter reports on collaborative and individual performances and presentations of Seis8s – the language described in Chapter 2. Each discussion starts with contextualizing the place and event where they happened. This is followed by short vignettes narrating the performances. Each section ends with a brief discussion identifying and reflecting upon the project's objectives: 1) Engaging the Latin American community worldwide with live coding performance and asking them to reflect upon the options and restrictions of computer music

languages. 2) Engaging users in activities such as learning and producing electronic Latin dance music.

The performances described in this chapter fall within the field of live coding practice, which is the performance art of writing code in front of an audience to generate expressive results such as real-time musical presentations (Collins et al., 2003; Nilson, 2007). As a practice that emphasizes the theatrical use of textual and visual programming languages, live coding works as a pedagogical tool for people to experience the affordances – i.e. options and restrictions – of music-programming technologies.

Chapter 1

State of the art: programming languages as texts and cultural objects

This chapter builds upon current investigations around culture and software [1] by investigating cultural traits in programming languages through interpretative and semiotic analysis. The languages investigated are Logo, Cornrow Curves, Rhythm Wheels, Lenguaje Latino, and Alb. These languages are not all connected to Latin America or to music. However, I chose them because, in addition to their computational focus, they have an explicit orientation toward pedagogical, representational, linguistic, and infrastructural spheres, thus allowing for a broader overview of the topic. Guiding questions include: *How are cultural traits embedded consciously and unconsciously into computer languages? How do these traits orient computer languages? What does this orientation do to their users and their societies?*

My analysis starts with Logo, a well-known computer language developed in the 1960s US but whose reach extended internationally. I focus on the *familiar* culture of English-speaking children in the US and the moment where a type of *computer culture* emerged around Constructivist theories. By discussing the resulting emergence of programming practices based on natural language, procedural thinking, and anthropomorphism of computers, I briefly suggest how this cognitive theory helped orient a generation of computer developers

[1] Galloway (2004); Chun (2011); Allen (2013); Lialina and Espenschied (2015); Ali (2016); McPherson (2018); Eubanks (2018); and Noble (2018)

and users.

After this analysis, I look at the languages Cornrow Curves and Rhythm Wheels, providing a discussion of some pedagogical frameworks that have developed in response to another type of universality. That is the one connected to the Westernized orientations that logic and mathematics took in elementary schools in the US in the 20th century. Here, I pay particular attention to the ways computing both absorbed and resisted these orientations through *culturally situated software*.

After, I move on to discuss Lenguaje Latino, a general-purpose Spanish-inflected language, where I reflect on the linguistic hierarchies that emerge from its use of *standard Spanish*. These hierarchies seem to intersect with the concept of *general-purpose technology*, which I also explore in this section. I provide nuance by discussing Spanish-speaking user forums where both positive and negative attitudes tend to emerge from the use of Spanish in these technologies. And I conclude by pointing out how the so-called digital divide and the knowledge economy in Latin America might influence these comments.

The last discussion revolves around Alb, another general-purpose language but now Arabic-inflected, where I put forward the practice of *reading* code from cultural studies and software development perspectives. I highlight the complications that exist with the latter, some of which arise from the current socialization practices in programming. For example, in open-source software where social and technological infrastructures of collaboration for reading, critiquing, and improving code are only possible if one operates in English. I conclude by discussing how Alb borrows from speculative design to visualize these matters.

1.1 Theoretical framework

In the mid-20th century, literary theorist Roland Barthes applied "European linguistic structuralism" theories to the analysis of mass-culture artifacts to unveil how they were "capable of displaying an idea, or even an ideology" (Calefato, 2008, p. 72). From this perspective,

magazine photographs, films, food, plastic, and even soap are texts conveying social patterns resulting from discourse. For example, in his essay about French toys, Barthes (1957/1927) discusses how these represent the "microcosm of the [Frenchman] adult world", reproducing socializations found in "the Army", "the Post Office", and in "Hair-Styling", among others (p. 53).

In a similar way, in the 1970s, cultural studies scholar Stuart Hall developed a theory about the ways mass communication flows through the "distinctive moments [of] production, circulation, distribution, consumption, and reproduction" (Hall, 1999, p. 508). Applied to television, this theory allowed seeing how this artifact resulted from a production language tied to the structures of broadcasting institutions (p. 509). The latter, orienting it to convey messages tied to these institutions' standardized social practices and their societies at large.

More recently, borrowing from the authors above, e-literature scholar Mark (Marino, 2020a) proposed investigating computer programs as cultural objects to enable discussions about them and their cultures. This methodology, called Critical Code Studies, investigates computer programs in the physical and virtual worlds and the "realms of production, commerce ... and scientific and academic inquiry" (p. 31). Furthermore, Critical Code Studies pays attention to "studies of technology and innovation" in connection with current "discussions of race and ethnicity, gender, sexuality, socioeconomic status, political representation, [and] ideology", among others (p. 32).

In this chapter, I suggest using the toolkit described above not only for the analysis of computer programs but also for the computer languages they are built with. The latter makes sense as many of these languages rely on characters, such as letters and numbers connected to specific modes of knowledge — e.g., the Latin alphabet or the Arabic numbers. It also makes sense as many programming languages use keywords or commands, which often relate to objects and actions described through nouns, adjectives, verbs, and adverbs. Moreover, it makes sense as programming languages are objects designed for specific purposes in specific parts of the world and around specific cultural logics and historical moments.

1.2 Logo

Logo was a programming language for children, designed in Massachusetts in 1966 by Seymour Papert, Wallace Feurzeig, and others (Solomon et al., 2020). Just by this brief introduction, it becomes obvious that, in addition to having a computational orientation, Logo also had a cultural orientation related to children. But what are the specifics of this orientation? That is, what makes this language particularly oriented to the social sphere of children? One answer could be found in that Logo was influenced by the scientific realm of psychology, particularly from Constructivist theories. Logo designers focused on children being able to construct their knowledge as opposed to passively receiving it. They achieved this through target experiences or activities that Logo designers embedded into the language.

Some consequential questions could be about the type of activities children were oriented to do in Logo, or perhaps, about the relation between these activities and the society children were members of. Logo's creators Solomon et al. (2020) highlight that Logo was developed around the idea of microworlds. The first of these microworlds revolved around natural language, particularly the use of "English words and sentences" that children could experiment with by coding "nonsense poetry [and] word games" (p. 79:21). As Logo was created in Massachusetts, one could presuppose that —at least at the beginning— it was oriented to children in that geographical area. And, as there is a large amount of English-speaking people there, one could assume that Logo's first microworld was oriented towards English.

One can also wonder if all the English-speaking children in Massachusetts speak the same English, just to discover that in Massachusetts there are the dialects of the New England accent and the Boston Accent —both with distinct vocabularies. And, this could lead one to wonder if meaning in Logo was affected by these or other linguistic variations. Solomon et al. (2020) recount the difficulty of translating the command names in the so-called English Logo. Here, a procedure is named using the infinitive mood, and the command to request it uses the imperative mood. Take, for example, the Logo procedure name to `gossip`, which

```
TO VIOLETS
 PRINT [ROSES ARE RED.]
 PRINT [VIOLETS ARE BLUE.]
 PRINT PRAISE
 PRINT RHYMOO
END

TO RHYMOO
 OUTPUT ( SENTENCE [AND] WHO MOO )
END

TO MOO
 OUTPUT PICK [[DOES TOO] [LOVES YOU] [SNIFFS GLUE] [SIPS DEW]]
END
```

Figure 1.1: A Logo program that generates a poem (Goldenberg and Feurzeig, 1987, p. 32)

after defining it can be called back with the command `gossip`. In English, both moods have few differences because the language has low inflection, but in more inflected languages, the moods vary more (p. 79:48). In Spanish, for example, the infinitive of the verb gossip would be chismear, whereas the imperative could be either chismea, chismee, chismeemos, or chismeen — depending on their grammatical context.

```
TO GOSSIP
 OUTPUT ( SENTENCE WHO DOESWHAT )
END

TO WHO
 OUTPUT PICK [SANDY DALE DANA CHRIS]
END

TO DOESWHAT
 OUTPUT PICK [CHEATS. [LOVES TO WALK.] ⟶
              [TALKS A MILE A MINUTE.] YELLS.]
END
```

Figure 1.2: Code that generates a sentence from a subject and a predicate (Goldenberg and Feurzeig, 1987, p. 18)

Reflecting on the relation between Logo and the region of the world in which it was created might enable conversations about the sociopolitical aspects of this region. Furthermore, it might enable seeing a relation between sociopolitics and the sensibilities the language intended to awaken. For example, Solomon et al. (2020) explain that in 1957 USA, one major

concern was the space race, which generated a societal crisis after Russia put the Sputnik satellite in orbit. This event oriented the U.S. to focus on improving the math and science programs within their public education system. And, by consequence, this oriented Logo designers to create a second microworld for the language, which by 1970 allowed children to explore mathematics, with the long-term expectation of awakening in them the desire to become mathematicians (p. 79:16).

Discussing geographical specificities also unfolds into questions about Logo's influence in virtual worlds. For example, from the many existent forms of mathematics, Logo creators chose to develop learning activities around geometry. The geometry they used, however, was not formal geometry such as the Euclidean and Cartesian styles. Rather, they developed their own "computational style of geometry" called Turtle Geometry, where children were able to direct the movement of a point in space —also called a Turtle— to create shapes (Papert, 1980, pp. 55-58). This geometry was based on the idea of learning through association. That is, in Turtle Geometry, children learned how bodies occupy space by associating the movement of the human body with the movement of the turtle's body (pp. 53, 63).

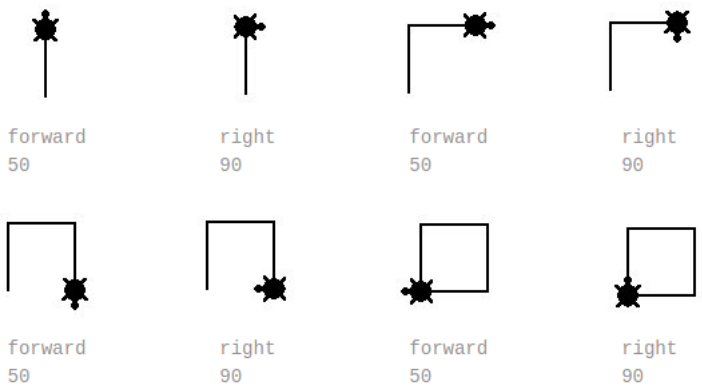


Figure 1.3: Steps of a Turtle program that generates a square ("Off the Floor and Onto the Screen", 2022)

Reflecting on the cultures that made it into Logo makes us wonder about the cultures that came out of it. For example, we have already discussed how Logo creators instilled values for children to become active users, and creative researchers interested in mathematics. But what about the modes of thinking these children developed? Solomon (1978),

for example, describes how Logo was a "computer culture" in that it was both "a way of thinking about computers and about learning" through computers (p. 20). This learning culture encompassed, for example, the "anthropomorphic view of the computer" drawn from Constructivism and explained in the paragraph above. It also encompassed a compassion culture where the error was normalized and where debugging was expected. And, it also encompassed a way of thinking in terms of procedures, that is, where the smaller parts of a complicated process are given meaning by naming them and describing them based on one's prior knowledge (pp. 21-22). The latter unfolds into questions: How different would the children's development be if other modes of thinking were encouraged, such as declarative thinking? [2] How does procedural thinking impact the ways *real-world* things are abstracted into computer programs? What has been the impact of this mode of knowledge in the development of other computer languages and in other areas of life?

1.3 Cornrow Curves and Rhythm Wheels software

Cornrow Curves and Rhythm Wheels is a software series for elementary and high-school students made in the early 2000s by cybernetics scholar Ron Eglash, ethnomathematician Jim Barta, and Indigenous educator Ed Galindo. This software sought to help students gain a "sense of ownership over math and computing" by exploring culturally responsive activities and their underlying mathematics (Eglash et al., 2006, pp. 349, 352). Specifically, the Cornrow curves software invites students to play with "fractal patterns" found in "cornrow hairstyles" to learn about fractals themselves, and about geometrical transformations like rotation and translation (p. 350). The Rhythm Wheels software invites students to experiment with the "ratios between beats in percussion" in rhythms such as Afro-Cuban bembé to learn about topics like the Least Common Multiple (p. 356).

[2] Declarative thinking, also known as declarative knowledge, refers to the "awareness and understanding of factual information about the world—*knowing that* in contrast to *knowing how*. Its necessary and sufficient conditions are that the information must be true, that the person must believe it to be true, and that the person must be in a position to know it" (Colman, 2015).

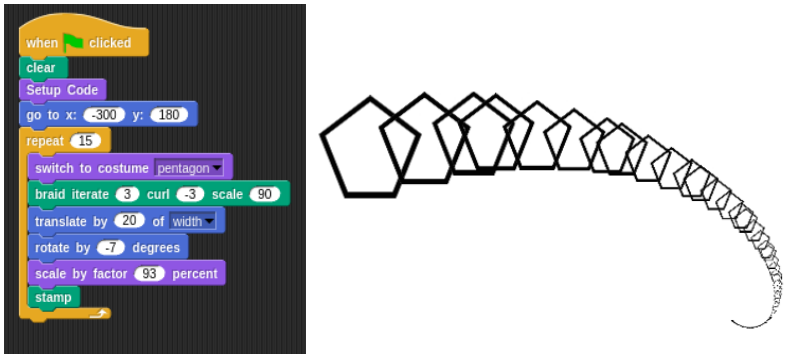


Figure 1.4: Code generating a cornrow fractal. It was created by the author of this dissertation from https://csdt.org/applications/128/run

The first thing to notice is the connection this software has to a series of pedagogical approaches that emerged in the 1980s U.S. These include "culturally appropriate" teaching, "culturally congruent" teaching, "culturally compatible" teaching, and "culturally relevant teaching" (Ladson-Billings, 1995, p. 467). One main difference between them was how they focused more or less on the "accommodation of student culture to mainstream culture" and the creation of "a more dynamic or synergistic relationship between home/community culture and school culture" (p. 467). These approaches made sense at the time as a way to address poor content in math textbooks and pedagogical approaches where children failed to connect their prior mathematical knowledge to the knowledge acquired at school (Gustein et al., 1997). The latter, being more evident with students of color, keeping them from "[developing] the tools for active participation in democratic public life" (pp. 710-711).

Pedagogies connecting the students' cultures draw from anthropological fields like "computational ethnography", which focuses on developing computational and mathematical models based on the cultural traits of a given community (Eglash et al., 2006, p. 350). The anthropological aspects of Cornrow Curves software are described by one of its creators Ron Eglash, who was investigating the "iterative scaling" patterns —also known as fractals— found in architecture in West and Central Africa (p. 349). Here, Eglash saw strong potential in applying this fractal model into the classroom so children could engage with "African mathematics" (349). For Rhythm Wheels software, the anthropological aspects came from

the interaction with Puerto Rican children living in New York, leading to the development of activities focusing on the mathematical components of Caribbean music found in rhythmic claves (p. 356).

Cornrow Curves and Rhythm Wheels are not programming languages per se. Rather they are small libraries for the computer language Snap!, which is a visual language "that [works] by snapping blocks together" and where "each block [represents] a procedure call" (Solomon et al., 2020, 79:48). The idea behind visual languages is that they potentially provide a more intuitive way to program by representing objects graphically, eliminating convoluted syntax, and suggesting what data has to be given to the functions (Solomon et al., 2020; Begel, 1996). Snap! started as a sub-project of another language called Scratch, which at the same time is based on a Logo version called LogoBlocks, developed in 1996 for the Danish toy company LEGO.

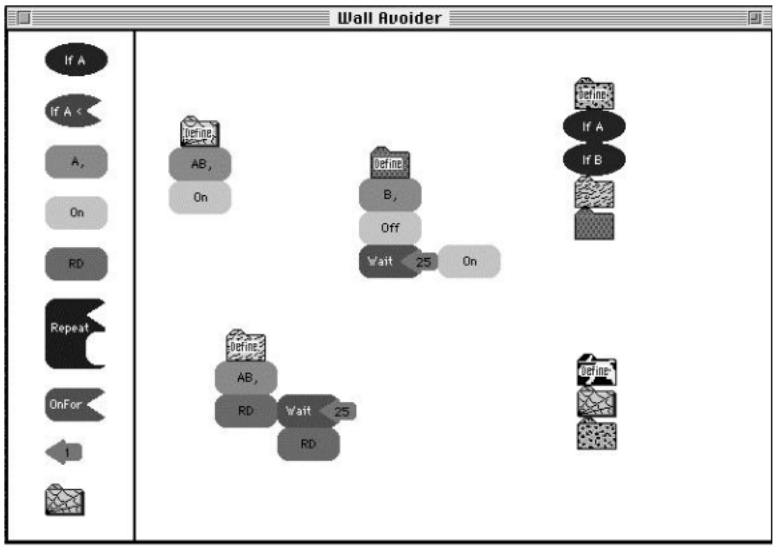


Figure 1.5: A LogoBlocks Program (Begel, 1996, p. 12)

An interesting aspect to consider here is that although visual programming existed before as flowcharts, the idea of interlocking programming blocks was initiated with LogoBlocks, arguably for two reasons. First, LEGO was all about interlocking blocks (or bricks), and secondly, because at the time, Logo developers were experimenting with alternative code

representations (Solomon et al., 2020; Begel, 1996). Here, it might be interesting to ask what else can one infer from the block/brick representation? One thing to say is that although the brick, as a construction material, has been used by many cultures throughout the millennia, it is also strongly tied to Denmark's architectural heritage where there was a lack of other materials like rock ("Architecture of Denmark", 2022). The value of the brick is also connected to broader European 20th-century aesthetics like Brick Expressionist architecture, happening in places like Germany and the Netherlands. This is not to say that the brick-/block representation in visual programming was necessarily Eurocentric but it might suggest that because of its tie to these architectural histories, it became —perhaps unconsciously— an epistemology worth pursuing [3].

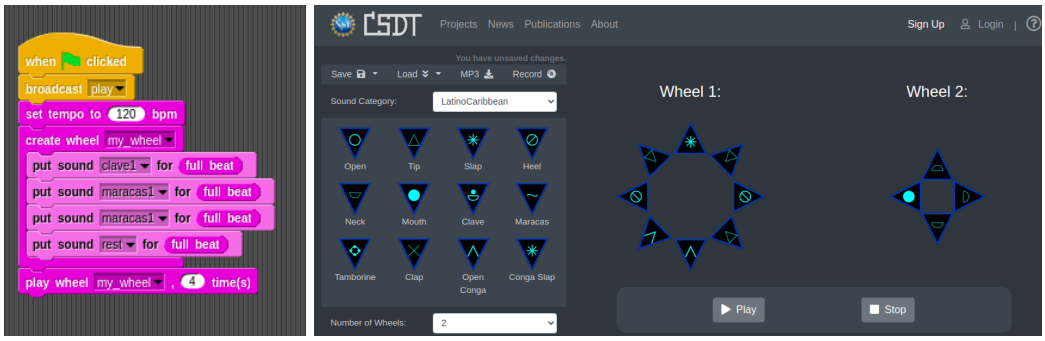


Figure 1.6: A picture of Rhythm Wheels code (left) and graphical interface (right) taken by me from https://csdt.org/projects/18209/run & https://csdt.org/applications/39/run

The brick also invites one to think about the construction of national identity through working-capital standards revolving around software like Cornrow Curves and Rhythm Wheels. About this, Labor studies scholar Teitelbaum (2016) describes that since World War II, the U.S. has had recurrent concerns about "falling behind in a global 'race of talent' that will determine the country's future prosperity, power, and security" (p. 1). As a result, these concerns have produced public policies "[addressing] all levels of education together, from kindergarten through postdoc"; and where "worries about the performance of the [elementary] education system over the long term [has been conflated] with concerns about the

[3]For more discussion on how bricks are metaphors of ways of knowing please see Jimmy, E., Andreotti, V., and Stein, S. (2019). Bricks and Threads. In Towards braiding. Musagetes Foundation, pp. 13-20

adequacy of U.S. production of advanced degrees (especially doctorates) in science and engineering in the present-to-near term" (p. 26).

In the late 1990s, these worries appeared again due to a boom in "information technology (IT), Internet, telecommunications, and biotech" which produced a "sudden rapid growth in demand" for workers (54). In the early 2000s, concerns about insufficient workers influenced education policies as mentioned above, resulting in curricula development focused on science, technology, engineering, and mathematics (also known as STEM). These curricula, however, is often seen as imbued in a corporate ideology that prevents students from focusing on "lifeways which are not based on alienated labor and value extraction" (Eglash et al., 2017, p. 6). And where the student is expected to be a "problem solver, logical thinker, technologically literate and able to relate his (or her) own culture to the learning" (Gallant, 2010, p. 2).

1.4 Lenguaje Latino

Latino is an open-source general-purpose programming language for Spanish-speaking people created in 2015 by Mexican software developer Primitivo Roman. Latino is directed to both non-experts —who want to reduce their learning curve— and experts who might just want to program using Spanish keywords. For example, the command `escribir`, which in its infinitive form translates to "to write", will print a given text.

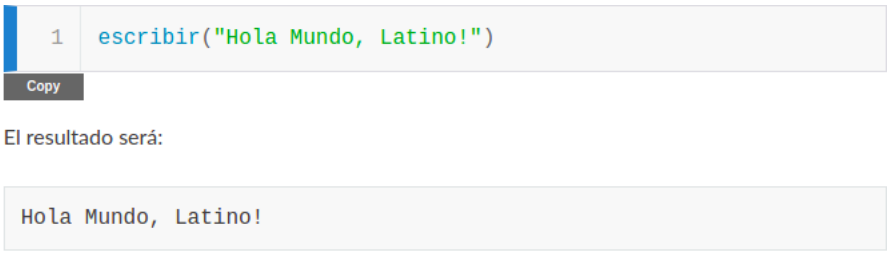


Figure 1.7: A Latino command to print a given text (Guerrero, 2021b)

Linguist Heriberto Avelino (2018) explains that Spanish, a language "spoken by approximately 405,638,110 speakers in the world", encompasses two main categories corresponding to the one spoken in Spain — known as "Peninsular Spanish" — and the one "spoken in the

Americas" —including the United States and Canada (p. 223). It also encompasses "some parts of Africa", where it is spoken natively too (p. 223). Because Latino's developer, Primitivo Roman, is located in Mexico City, one can infer that Latino draws from the Spanish of that city, where among many particularities, one can find "two major intonational patterns", one where the "peak of a pitch accent is shifted from the stressed syllable to the following syllables" and another "where the peak of the pitch accent is aligned with the stressed syllable". The latter being "emblematic of the working class speech variety of Mexico City Spanish" (p. 228).

Generally, written Spanish includes graphic accents both to provide guidance on intonation and to help differentiate between words with similar spelling. It is interesting, however, that in Latino, the keywords or commands do not have accents. Thus, making it difficult to correlate the written code with intonation. For example, the command indice for finding an element within a list lacks the accent in the letter "i" to form índice.

```
1  mensaje = ["Hola", "mundo", "Latino"]
2  escribir(lista.indice(mensaje, "latino"))
3  escribir(lista.encontrar(mensaje, "Latino"))
```

Figure 1.8: Latino code that generates a list and finds an element from that list (Guerrero, 2021a)

Perhaps the lack of accents is because of practical reasons, for example, so the user avoids spending time typing keyboard combinations. This practicality, however, makes one think that Latino is, perhaps inadvertently, using a so-called *neutral Spanish* or *standard Spanish*. This is a type of Spanish, connected to the historical development of the entertainment and communication industry in early 20th-century Latin America, where the vocabulary and the accents are hard to trace back to any particular country. This is audible in translated U.S. TV shows that aimed to reach a large number of Latin American countries (Chavarría, 1997). It is also found in the "adaptation of text manuals, informatics programs, [and] websites", where because of the practicality of being translated once, they "avoid terms that could be connected to with specific countries ... or linguistic regional phenomena" ("Español

estándar", nd, para. 15)

This practicality is also reinforced by the fact that the syntax of Latino seems not to change much from languages like Python or C++, for example. In other words, it seems that most of the effort has been put into translating the commands, missing explorations with other language constructs related to Spanish. And, while this might work for learners who want to learn programming, it might also be a missed opportunity to explore functionalities that come from the ways of knowing and being of Spanish speakers in Latin America. Furthermore, by just exploring the form and leaving the function untouched, Latino potentially validates normative values of programming.

Perhaps the latter becomes more obvious by looking at Latino's intention to enable the production of a diverse range of software applications, such as mathematical computations, web server development, database connectivity, and system scripting ("About Latino", 2022). Languages that have the characteristic of developing such a wide range of applications are called *general-purpose languages* —in contrast to languages with a specific orientation or domain. Arguably, the concept of general-purpose languages finds its roots in the concept of *general-purpose technology*. That is, technology capable of "[spreading out] to most sectors", being in constant improvement to "keep lowering the costs [for] its users", and "[making it] easier to invent and produce new products or processes" (Janovic and Rousseau, 2005, 3). Examples of these technologies are "[s]team, electricity, internal combustion, and information technology" (p. 1).

In evaluating Latino's status as a general-purpose technology, one can see clear intentions for it to spread to various sectors through the activities mentioned above. It also does this through its capacity for running on Windows, Apple, and Linux computers and interacting with industrial computer languages like Redis and MySQL ("About Latino", 2022). Latino's intentions to be in constant improvement are present in that it adheres to the open-source model by having a GitHub page where people can contribute and add to the language freely. Finally, Latino intends to make software production easier for Spanish speakers through its

commands and documentation where programming concepts are explained in Spanish.

```
//Latino: Serie numérica Fibonacci
funcion fib(n)
    a,b = 0,1
    mientras a < n
        a,b = b, a+b
        escribir(a)
    fin
fin

fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Figure 1.9: Latino code that generates the Fibonacci numeric series ("Latino v1.4.1", 2021)

One thing to consider, however, is to what extent users see the intentions described above as congruent. One can do this by looking at attitudes towards Latino in places like opinion forums. For example, in forobeta.com [4], under the title "Lenguaje de programación con sintaxis en Español [A programming language with Spanish syntax]"(2017), participant Son Lux states that "for a job, it would be difficult for a programmer to be asked to know Latino" (translation mine). However, they ask the Spanish-speaking community how feasible they think it is to learn this language and/or develop an application like a website. This post generated nine responses, from which two can be categorized as positive and the rest as negative. Positive answers included mentioning how interesting it would be to program in Spanish, even more, if the language has a high utility or capacity. Negative responses focused on a perceived impossibility of programming in something different than English. The latter, because "the community" is already accustomed to programming in English, and because information regarding programming concepts and procedures that are of good quality on the Internet are in English.

Information technologists Herrera Rivas and Roque Hernández (2019) concur that a large amount of digital information found in the Internet is in English as "the majority of Internet nodes and domains belong to the United States and to other English-speaking

[4]See full discussion here: https://forobeta.com/temas/lenguaje-de-programacion-con-sintaxis-en-espanol.594213/

countries" (p. 4, translation mine). In this regard, the ability to read texts in that language is key for non-English speakers within the current globalized world. In Latin America, this is categorized as a common barrier to overcome, adding to others such as access to the Internet itself. The latter makes one think that perhaps a bigger motivation for building Latino is to contribute to overcoming the so-called *digital divide*. The digital divide is defined by the OECD (2001) as "the gap between individuals, households, businesses and geographic areas at different socio-economic levels with regard both to their opportunities to access information and communication technologies (ICTs) and to their use of the Internet for a wide variety of activities" (p. 5). The importance of overcoming the digital divide is, for example, for people to be able to communicate with others, for them to be able to access information and to be able to enter the global economy. To achieve these goals, it is said that nations should focus on increasing communication infrastructures and providing their inhabitants with "computer/Internet literacy" skills (p. 6).

From all this, one can infer that Latino is motivated by the current global economic model, known as the *knowledge economy*. This is a term coined in the late 20th century by business educators describing how after World War II, the U.S. economy started shifting from "an economy of goods" into an economy where "ideas, concepts, and information" are "[applied] to productive work" (Drucker, 2017, pp. 263-264). This type of knowledge, now a global trend, differs from other knowledge, such as *intellectual knowledge*, in that it is "[t]he systematic and purposeful acquisition of information and its systematic application" (p. 267). The latter, so it aligns to the *knowledge industries*, like the ones focused on services, which insatiably demand "knowledge workers", particularly "computer programmers", "system engineers, system designers, information specialists, and health professionals" (pp. 266-267).

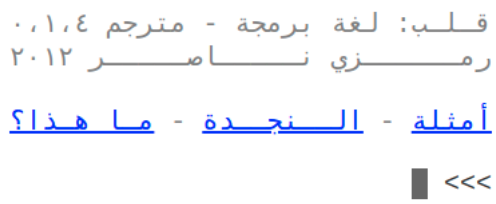


Figure 1.10: The welcome information displayed when starting the programming language Alb (Nasser, 2012a)

1.5 Alb

Alb is a programming language derived from Arabic that explores the "impact of human culture on computer science, the role of tradition in software engineering, and the connection between natural and computer languages" (Nasser, 2012b, para. 1). Alb, whose name translates to "heart", was created in 2012 by Lebanese-American computer scientist and game designer Ramsey Nasser with the purpose of illustrating the challenges of coding in languages other than English (Galperina, 2014).

```
("hello world" قول) <<<
hello world
hello world <==
<<<
```

Figure 1.11: A snippet of code that prints "hello world" (Nasser, 2012a)

The first thing that one can infer from the statement above is that Alb is not necessarily targeted at Arab speakers. Rather, its function is to expose an English-speaking audience —or more broadly speaking, a non-Arab speaking audience— to a critique of programming being too strongly tied to English logic as a natural language. The latter becomes clear when its creator mentions the following.

> *"All modern programming tools are based on the ASCII character set, which encodes Latin Characters and was originally based on the English Language. As a result, programming has become tied to a single written culture. It carries with it*

a cultural bias that favors those who grew up reading and writing in that cultural [context]" (Nasser, 2012a, para. 2).

An interesting aspect is that Nasser does not intend for English/non-Arab speakers to learn his computer language nor to start programming in Arabic. Rather, he invites them to try *reading* its code. This approach makes sense as reading code is tied not only to cultural studies as discussed in the introduction of this paper. Reading code is also tied with the more practical field of software development where a common task for programmers involves "fixing, inspecting, or improving code" and where skills of "reading, understanding, and modifying the original code" are key (Spinellis, 2003, 1). In this sense, the first practical limitation that Alb conveys is the difficulty for the *conventional* or *standard* programmer to perform the aforementioned tasks and thus of improving a given program or system. From a different perspective, this reading is about the possibility for the programmer's knowledge to deepen if borrowing from languages like Arabic. The latter has happened already with the Arabic numerals that were introduced to Europe in the 13th by Fibonacci, and that are not only used in programming but in Western sciences in general.

Reading code also involves socialization practices; for example, open-source software invites programmers to contribute, "read, critique, and improve" the code and, in exchange, learn from the code itself (Spinellis, 2003, p. 3). The latter is even seen as a literary practice where code is "[d]iscussed, internalized, generalized, and paraphrased" and where there are aesthetic judgments of high-quality vs low-quality code. Furthermore, reading code in such a way is key for programmers to "enrich [their] vocabulary, trigger [their] imagination, and expand [their] mind" by discovering "new architectural patterns, data structures, coding methods algorithms, style and documentation conventions, application programming interfaces (APIs), or even a new computer language" (p. 3).

Figure 1.12: A command to display the "about" section. It has a reversed question mark at the end (Nasser, 2012a)

Although Alb proves to be obscure for non-Arab speakers, aspects of its code can still be discussed in a literary way. For example, looking at the command in 1.12, one notices the use of parentheses which is a known symbol to computer users. From this, one can then start reflecting on the semantic meaning of punctuation marks like brackets and semicolons in modern programming and later start speculating how that meaning might change in Arabic. In that same example, the use of a reversed question mark makes one see parallelisms with symbols in other natural languages. For example, in Spanish, there is the use of an inverted question mark "¿" used in the opening of a written question. One can also discuss the aesthetics of Alb, for example, the diacritical marks and the cursive aspect of it, to make one realize the lack of these aspects in programming languages.

This constant invitation to reflection and speculation makes one infer that Alb is also imbued within the field of *speculative design*. This is a type of design that tries to move away from design as "problem solving" into design as an enabler of "discussion and [debates] about alternative ways of being" (Dunne and Raby, 2013, p. 2). The goal of speculative design is not to predict nor define the future, but rather to investigate "the idea of possible futures" (p. 2). The connection makes more sense as speculative design came out in the early 2000s —near to the time of the first exhibition of Alb —which was a time when the state of digital technologies and applications for the Arab world was very deficient. In terms of the Internet, for example, there were poor developments in morphological analyzers and language identification systems, which made it difficult "getting the right meanings" of Arabic texts to applications like text-to-speech translators and search engines (Albalooshi et al., 2011, p. 379).

Finally, the obscure quality experienced when reading Alb makes one wonder about its connection to the concept of esoteric knowledge. That is a knowledge whose quality is "being obscure" in contrast to "exoteric knowledge" which is accessible or public ("Esotericism", 2022, para. 1). Esoteric knowledge might also denote a type of knowledge with a testing purpose that will lead to "learning the higher truth" (para 1). It might also denote a certain

```
>>> (أمثلة)
سهل: مرحبا يا عالم
متوسط: عدد فيبوناتشي
متقدم: لعبة الحياة لكونواي
<==
<<<
```

Figure 1.13: A snippet of code that shows some program examples (Nasser, 2012a)

knowledge that by "nature" is "accessible only to those with the proper intellectual background" (para. 1). In programming, there is actually a type of languages called *esoteric* which include some or all of the characteristics mentioned above. Furthermore, the purpose of these languages is not necessarily to be executed, but rather to be a proof of concept or an art piece. For the communication culture scholar Geoff Cox, these languages are ones that "shift attention from command and control toward cultural expression and refusal" (Cox and McLean, 2012, p. 5).

1.6 Conclusion

Throughout this chapter, I investigated the programming languages Logo, Cornrow Curves, Rhythm Wheels, Lenguaje Latino, and Alb. Using interpretive and semiotic analysis, I outlined threads discussing the influence of *culture* in these languages. On a larger scale, the aim of this chapter was to open the possibility of a thesis suggesting that *culture* gets embedded in *all* computer languages. The latter might be obvious to some, however, my experience is that suspicion often emerges when one brings up these connections. For example, when people point out that computer languages should be *universal* after I comment on the *Spanish* computer language I built for this dissertation.

Another takeaway from this chapter is the various spheres intersecting in each of the discussed languages. For example, Logo, Cornrow Curves, and Rhythm Wheels were influenced by pedagogical theories such as Constructivism and Culturally Responsive Teaching,

respectively. In Lenguaje Latino pedagogy emerged as the invitation for Spanish-speaking users to learn programming concepts in their mother tongue. And, in Alb, this was precisely a reflection of how culture and tradition get embedded in these technologies.

The pedagogical sphere intersected with diverse representational dimensions. This is perhaps most evident in Cornrow Curves and Rhythm Wheels, motivated by the creation of meaningful connections between the user's culture and the scientific mainstream culture. In my discussion, however, I suggested the caveat that often, the appeal to the representational dimension is not the exploration of other lifeways but rather the incorporation of the user into the US corporate way of life.

When discussing the linguistic realm, this chapter uncovered the potential imagined communities of these languages. For example, in my discussion of Logo, I pointed out the usage of standard English over other linguistic variations, such as the New England and Boston accents. When discussing Lenguaje Latino, I highlighted the dual aspect of encouraging computational learning through one's mother tongue: it helps reduce the digital divide while using a standardized version of Spanish, representing just a few in Latin America.

Finally, standardized Spanish in Lenguaje Latino revealed a connection to concepts such as general-purpose technology. This connection also appeared in other languages discussed in this chapter, providing insight into the multiple infrastructures they belong to. For example, in the discussion of Alb, I identified the need for the users to subscribe to the Anglophone thinking structures to participate in the social and technological spaces where digital collaboration happens, mainly through the Internet.

Chapter 2

Seis8s, a live-coding language for electronic Latin dance music

This chapter discusses Seis8s, a web-based computer language that I developed that allows real-time interaction with digital audio and localized musical knowledge. Seis8s is particularly oriented towards Latin Dance Music, where commands and their resulting sounds revolve around instruments, such as `teclado` (i.e. a keyboard), `bajo` (i.e. a bass guitar), `güira` (i.e. a percussion whose sound resembles one of a shaker), and `congas` (i.e. drums characteristic of Cuban music). Seis8s commands also revolve around nouns that convey actions connected to those instruments, for example, `acompañamiento` (accompaniment), `ritmo` (rhythm), `punteo` (picking), and `tumbao` (i.e. rhythmic pattern).

Seis8s draws from live coding, a practice where audiovisual outputs are performed live using code and where sharing the performer's screen is strongly encouraged (Collins et al., 2003; Nilson, 2007). Seis8s is meant for the performer and the audience to experience the code in addition to the music, helping them connect with additional cultural layers of computer-music languages, like natural language. Through building Seis8s, I explore the following possibilities: 1) to derive a computer-music language from Spanish; 2) to appeal to an imagined community in/from Latin America; and 3) to explore the cultural, political, economic, and historical commonalities of that imagined community. Seis8s can be accessed

at https://seis8s.org/ and https://estuary.mcmaster.ca/.

```
tempo 100;
compás "partido";
acordes [re m, fa, la];

punteo [3a, 5a] [3, 4, 1 1.5 2 2.5] $ sample 3 $ cumbia acordeón;
acompañamiento (2 4) $ vol 0.75 $ cumbia teclado;
tumbao [1a 3a 4a] [1 3 4] $ sample 4 $ cumbia bajo;
ritmo [1 2 2.5 3 4 4.5] $ paneo 1 $ cumbia guira;
marcha (p t p a a) (1 2 3 4 4.5) $ paneo 0 $ cumbia congas;
```

Figure 2.1: A functioning snippet of code from Seis8s

2.1 Motivation

My motivation to develop Seis8s comes from my experience using and teaching with computer-music languages. For example, in 2015, working at an elementary school in Mexico City, I encouraged children to explore music creation through the well-known children-oriented music language Sonic Pi (Aaron, 2016). Sonic Pi was first implemented in 2013 with the aim of teaching computer programming and computer music to children in the UK. Sonic Pi developed into a larger ecosystem that, in addition to the software, it included lesson plans inviting the students to discover the "artistic aspects of the software" (p. 175).

However, in the process of my teaching, cultural mismatches came up. For example, Sonic Pi used alphabet letters to program music notes (i.e. C, D, E, F, G, A, B) as opposed to syllables (i.e. do, re, mi, fa, sol, la, si). Letters are often taught in Anglophone music, and syllables are often taught in Hispanophone music. Another mismatch was that although this language advertised the potential to create within various genres, it seemed to favor Electronic Dance Music through inbuilt preset sounds and synthesizers recalling this genre.

I have experienced similar things as an artist using these technologies. For example, although English is my second language, I remember having difficulty when trying to understand the meaning of some commands from the well-known computer-music language

SuperCollider. One command I remember was `yield`, which allows suspending a procedure or routine. The confusion lay in that I could not find a translation that made sense in relation to computing or sound. The dictionary describes it as a verb connected to processes of cultivation, production, and investment; and to the action of demanding and surrendering something (Merriam-Webster, nd). Years later, I could infer that `yield` in SuperCollider was potentially connected to the expression "to yield the right of way". This expression is related to the act of stopping or slowing down in driving intersections and is often said this way in Anglophone countries.

```
\begin{singlespacing}

(
r = Routine { arg inval;
    inval.postln; // Post the value passed in when started.
    5.do { arg i;
        inval = (i + 10).yield;
        inval.postln; // Post the value passed in when resumed.
    }
}
)
(
5.do {
    r.value("hello routine").postln; // Post the value that the Routine yields.
}
)
\end{singlespacing}
```

Figure 2.2: A SuperCollider code using the command `yield` ("Routine", n.d)

Seis8s, however, is not a project about translating commands from English into Spanish. Rather it is about exploring textual and aural meaning from computer-music languages and from a Latin American perspective. Early work on this includes my collaboration with the Mexican audiovisual collective RGGTRN, where between 2016-2019, I used SuperCollider to create pieces alluding to the popular music of this region. Chichiricha, for example, is a piece based on "Cuban descarga"[1], which allows the performer to program live a percussion solo using samples resembling the ones used by a timbalero. Another piece is Cacharpo, made in collaboration with media artist David Ogborn (Del Angel and Ogborn, 2017). This is an automated live coding system that, among many things, explores how melody and rhythm can evoke music styles related to Mexican cumbia and electronic cumbia.

[1]Cuban descarga is a mode of improvisation over rhythmic and harmonic ostinatos.

Early work also involved detaching from my personal experience and looking into other cultural dimensions by engaging with the community. For example, in 2018, RGGTRN and I co-organized a series of workshops where people in Colombia, Perú, and Ecuador came to design mini-computer music languages based on their own cultural context (Del Angel et al., 2019). In these workshops, two interesting exercises came out: *La Calle*, by sound artist Ivanka Cotrina, is a mini-language based on the slang from working-class neighbourhoods in Lima, Perú; and, *Sucixxx*, by DJs and performers Chakala, Maria Juana, Carolina Velasco, and Daniela Moreno, is a mini-language borrowing from trans-feminism and where the textual interface is a reappropriation of misogynistic reggaeton lyrics. The interesting aspect lies in that these exercises provided an example of "intelligible ideology critiques of both live coding practice and the specificity of the artist-programmer"[2] (p.1).

This way, Seis8s is a project that intends to add to current investigations on the semiotic aspects of computer-music languages and computer-music technologies in general (see Cox and McLean, 2012; McCartney, 1994; Rodgers, 2012; Rodgers, 2016; Sterne and Rodgers, 2011; Ogborn et al., 2015). The latter is by conceptually and practically exploring the creation of a Spanish-derived textual interface that connects meaning with functionality as well as with shared cultural dimensions of Spanish speakers in/from Hispanoamerica.

2.2 Background: A personal anecdote on postmodernity and live coding in Mexico

Between 2010 and 2013, I made an apprenticeship at the Mexican National Centre for the Arts (Cenart) in Mexico City. There, I was involved in an initiative to explore the artistic possibilities of Live Coding and Free-Libre/Open Source Software. I was one of many artists involved through teaching workshops and participating in audiovisual performances. Overall, this initiative helped Cenart train a community of audiovisual performers who have now

[2]The artist programmers are "end-user programmers, in that they create software not for others to use as tools, but as a means to realise their own work" (McLean, 2011, p. 14)

spread out to various parts of Mexico and abroad.

Cenart was founded in 1994 by the Mexican Council of Culture and Arts to teach art and culture and promote interdisciplinarity in this realm (López, 2020). Mexican historian Pilar Maseda (2005) explains that several sociopolitical aspects were happening in Mexico at the moment of CENART's foundation. These included shifts towards a more neoliberal and globalized life, influencing the arts in Mexico through the incorporation of postmodernist thought in governmental cultural policies.

There are at least two dichotomic understandings of postmodernism in Mexico and Latin America, as explained by Argentinian anthropologist Nestor Canclini (Garcia Canclini, 2005). The first one is "the rupture with what came before", and the second is "being the land of pastiche and bricolage, where many periods and aesthetics are cited ... in a unique way" (6). Now, the type of postmodernist thought pointed out by Pilar Maseda (2005) in the paragraph above is not the eclectic one. Rather, it was one oriented toward the "disappearance of the manifestations of the past", "especially those from the national artistic past" (para. 5). In Mexico, this postmodern perspective, paired with governmental elitism, resulted in cultural projects that were incoherent with the "cultural manifestations ... and the economic situation of the country" (para. 13).

For example, Maseda talks about the schools of classical, contemporary, and folkloric dance, where the first two were incorporated into CENART, and the last one was left out. This rupture with the national past and its popular culture matches another definition of postmodernity, now given by French Philosopher Jean-François Lyotard (1979), in which postmodernity is a condition of knowledge based on the "incredulity toward metanarratives" (xxiv). This incredulity, he continues, is a "product of progress in the sciences" where legitimacy is gained through technological efficacy (xxiv).

Not all places in CENART were scientifically nor technologically oriented; however, the facility I was working in, called *Centro Multimedia*, definitely was. At this facility, one could find different *ateliers*, each oriented toward very specific digital technologies. There was the

audio atelier, the *moving image atelier*, the *digital graphics atelier*, the *interactive systems atelier*, and the *virtual reality atelier*. This facility, altogether, had a strong commitment to researching the application of information technologies to artistic processes. And it also had a strong preference for Free-Libre/Open Source Software (FLOSS) tools.

During my stay at *Centro Multimedia*, I worked at the *audio atelier* [3]. One aspect of particular interest to discuss is this place's adoption of SuperCollider as one of its main tools to compose and teach music throughout the early 2000s. The adoption of this tool over others interests me for many reasons, starting with the fact that SuperCollider was a recent tool just made available for the major multiple operating systems. SuperCollider presented a new area of research that contrasted with software like Max/MSP, which dated back to the 1980s and already had a usage tradition within Mexico's experimental composers.

Another characteristic of SuperCollider was that it was open-source software, contrasting to Max/MSP, which was proprietary. Here, the open-source paradigm presented a hidden contradiction. On the one hand, it made the *audio atelier* aware of the potential for displacing the category of 'expert' and allowing the general public to legitimize scientific and artistic knowledge. On the other hand, this same openness left unchallenged the accumulation of knowledge that characterizes the "general paradigm of progress in science and technology" (Lyotard, 1979, 7).

Less contradictory, perhaps, was SuperCollider's heavy emphasis on the concept of *real-time* programming, a well-identified characteristic of the *postmodern society* where the computational "miniaturization and commercialization of machines [leads to] advancements ... in the circulation of sounds and visual images" (4). Real-time programming enabled performers to experience audio results almost immediately after execution. It also allowed the *audio atelier* to pursue aesthetic research through the practice of *live coding*, through weekly audiovisual performances where a *new* type of artist known as *live coder* became commonplace.

[3] An archived website of the audio atelier, as it was in 2013, can be accessed here, http://cmm.cenart.gob.mx/tallerdeaudio/

So far, I have discussed the technoscientific aesthetic of the *audio atelier*. But what about its musical orientation? An aspect to note was its strong alignment with the Western art music tradition through genres drawing from Electroacoustic, Noise and Serial Music. This alignment was so present that during my stay there, I believed this type of music was actually an aesthetic result of using SuperCollider and live coding. It was not until 2012 that *Algorave music* – a live coding Techno-oriented genre – was introduced to me by a live coding band from the UK at an international festival by the *audio atelier*.

The latter unveils not only a strong emphasis on the Western art music tradition but also a separation between *cultured* and *popular* arts. A separation perhaps connected to the cultural policy mentioned at the beginning of this section, which encompassed an outdated and elitist understanding of fine arts. And which Maseda (2005) exemplifies by how CENART, as an institution, incorporated certain schools like the National School of Classical Dance and left out others like the School of Design.

Maseda also points out how this cultural policy was disconnected from *regional art* as well, for example, by not incorporating the National School of Folkloric Dance nor the School of Crafts into Cenart. It is common sense to infer then that this cultural policy extended into *Centro Multimedia* and the *audio atelier* through research schemata that did not include *regional* electronic music formations happening decades ago[4]. For example, *Tecnocumbia*, *Nortech*, or *Latin rock*, all of which, by the way, heavily rely on synthesized sounds and thus on technoscientific experimentation.

By 2013, when I stopped frequenting the *audio atelier*, the computer-music language TidalCycles was becoming more common in various parts of Mexico City and Latin America. TidalCycles emphasizes patterns and repetition, making it suitable for creating Algorave and Techno music. It differs from SuperCollider, whose scope is less defined. My perception, however, is that in addition to Techno, the uses of TidalCycles by the people in this region are

[4]Terms such as *regional* and *national* still make sense in the globalized context of the Latin music industry. For example, Chilean ethnomusicologist Agurto (2022) mentions that pan-continental ideals such as "the formation of a common Latin American consciousness [have] historically been secondary to national ideologies and superseded by nationalist divisions" (p. xxiv)

still heavily aligned with the Western art music tradition, for example, through workshops on sound art with TidalCycles at institutions such as the *Mexican Center for Music and Sound Arts*. A few exceptions I have encountered from 2012 until 2023 include the live coders Rafrobeat (Colombia), and Sireñoras (Argentina), who perform electronic cumbia music with TidalCycles.

The point of this brief account is not to judge the artistic practice or musical taste of the people involved in the live coding scene in Mexico and Latin America but, perhaps, to allow for further reflection on why the strong emphasis on a lineage of Western art music and non-regional popular music. Or, the other way around: why the non-exploration of electronic music formations with a *national* or *regional* lineage? Furthermore, what were the implications of leaving these unexplored through programming languages and live coding?

On the one hand, being a member of the *audio atelier*, *Centro Multimedia*, and CENART allowed me to engage with Western art music and *Algorave music*. This was a rich experience that, at the time, complemented my recent undergraduate studies in music composition in popular music. On the other hand, it ingrained in me a dichotomic view between *cultured* and *popular* arts. And I developed a value system where being a technoscientific artist was more valuable than being a *plain* artist.

On a larger scale, the institutional non-exploration of the possible relationships between *national* and *regional* music through electronic means, like live coding, was a lost opportunity. For example, in the potential development of tools like the one I present in the following section. And, in terms of practices such as the ones that in recent years have given birth to music variations such as "electronic cumbia, cumbiatrónica ... nu-cumbia" or simply "digital cumbia", "referring to .. the innovative crossing between cumbia and electronic music" (Márquez, 2016, 55).

2.3 Introducing Seis8s

Seis8s is a computer language that allows real-time interaction with digital audio and localized musical knowledge, particularly towards Latin Dance Music. Seis8s is zero-installation and runs in the web-based platform for collaborative live coding, Estuary (Ogborn et al., 2017; Ogborn et al., 2022). It also runs as a web application on my personal website. Seis8s was implemented with Haskell, a strongly-typed, functional, general-purpose programming language that has been used in computer music (e.g., Hudak, 2008; Hudak et al., 2007; Quick and Hudak, 2013) and live coding projects (e.g., McLean, 2014; McLean and Wiggins, 2011; Murphy 2016a; Murphy 2016b). Building Seis8s using Haskell facilitated incorporating it into the Estuary platform – also written in Haskell – allowing Seis8s to be used collaboratively. Haskell also has many libraries for translating high-level code into lower-level effects and computations, allowing me to explore various paths.

As a web application, Seis8s uses GHCJS and the Reflex Platform to translate Haskell code into JavaScript code to allow user interactivity, for example, the generation of real-time sound in the browser. Seis8s uses the JavaScript library WebDirt (Ogborn and Beverley, 2016), which allows playing and modifying sampled sounds on the web and which was initially developed for Estuary.

2.3.1 Lexicon

Seis8s revolves around keywords or commands related to Latin dance music – also known as urban Latin music or Latin popular music. This is a 20th-century derivation of musics "based on Afro-Cuban rhythms, as developed and performed throughout the Hispanic Caribbean basis and its diaspora" and which are "designed for accompanying social dancing" (Manuel, 1998, p. 127). It includes musics like "Dominican merengue", "Puerto Rican bomba", New York Salsa, and "Colombian cumbia", to mention a few (p. 127).

The first category of commands from Seis8s revolves around the instrumentation of

these musics, which incorporate instruments from European cultures – e.g., "string, woodwind, brass, and keyboard melodic instruments"; African cultures – e.g., congas, güira, and marimba; and Indigenous cultures – e.g., flutes like the "Andean quena" (Béhague, n.d, para. 2, 8). At the time of writing, Seis8s has the following instruments: `acordeón`, `teclado` (keyboard), `bajo` (bass guitar), `güira` (i.e., a percussion scraper), `jamblock` (i.e., a wood-sound percussion), and `congas` (i.e., drums characteristic of Cuban music).

```
acordeón; teclado; bajo; güira; jamblock; congas
```

Figure 2.3: A functioning snippet of code that, when evaluated, plays the sound of the available instruments in Seis8s

The second category of commands makes reference to the styles mentioned above. At the time of writing, there is one preset style that is accessed through the command `cumbia`. This command gives access to a set of rhythmic and melodic patterns that correspond to the instruments above and which allude to Mexican types of cumbia, such as "cumbia sonidera", "tecnocumbia", and "digital cumbia". Future styles will include other types of cumbia as well, such as the incorporation of salsa and reggaeton styles.

```
cumbia teclado; cumbia bajo; cumbia güira; cumbia congas; cumbia jamblock
```

Figure 2.4: A functioning snippet of code where the style `cumbia` is applied to a set of instruments in Seis8s. When evaluated, this code plays the sound of basic rhythmic and melodic patterns alluding to Mexican cumbia

The third category of commands revolves around nouns that convey actions connected to those styles and instruments. These commands also relate to the terminology used colloquially by musicians playing some of these musics. At the time of writing, these include `punteo` (picking), `acompañamiento` (accompaniment), `tumbao` (i.e. rhythmic and melodic pattern for the keyboard, the bass, and the congas), `marcha` (i.e. rhythm for the congas), and `ritmo` (rhythm).

There are also less idiomatic commands such as `volumen` and `paneo` that modify, more generally, the quality of the sounds. Finally, Seis8s includes the command `sample`, from

```
marcha (p t p a a) (1 2 3 4 4.5) $ cumbia congas;
```

Figure 2.5: A functioning code that returns the `marcha` command for the congas in Seis8s. When evaluating this code, it plays a rhythmic pattern with the congas sound

which different sound variations of these instruments can be accessed. Future commands will include ones related to sound effects to be applied to the instruments.

```
sample 3 $ volumen 0.75 $ paneo 1 $ acordeón;
```

Figure 2.6: Function snippet of code that modifies the sound quality of Seis8s' instruments by selecting sample 3, setting the volume to 75%, and panning the sound to the right

Finally, the fourth category relates to global modifications to be applied to all of the instruments. At the moment of writing, these include the commands `acordes` (chords), `compás` (measure or bar), and `tempo`.

```
tempo 100;
acordes [re m, fa, la];
compás "partido";

punteo [3a, 5a] [3, 4, 1 1.5 2 2.5] $ sample 3 $ cumbia acordeón;
acompañamiento (2 4) $ vol 0.75 $ cumbia teclado;
tumbao [1a 3a 5a] [1 3 4] $ sample 4 $ cumbia bajo;
ritmo ([1 1.5 2 2.5 3 3.5 4 4.5]) $ paneo 1 $ cumbia guira;
marcha (p t p a a) (1 2 3 4 4.5) $ paneo 0 $ cumbia congas;
```

Figure 2.7: A snippet of code using the four categories of commands in Seis8s to create a basic cumbia rhythm at 100 beats per minute, using the chords Dm, F, and A

2.3.2 Syntax

The parsing library that Seis8s uses is called Haskellish (Ogborn, 2019), and it helps create small languages that have similarities with Haskell. In this way, Seis8s inherits, in a very general way, the Haskell way of forming valid instructions, particularly the application of functions done from right to left. It also inherits the use of parentheses and the "$" symbol, which are punctuation marks that separate the functions from each other – the "$" sign is

equivalent to parentheses (loosely speaking). Finally, Seis8s uses the semicolon to separate expressions – i.e. structures formed from concatenated commands.

In Seis8s, the basic unit is the instrument, to which commands can be added to the left to modify it, for example: `acompañamiento (2 4) $ teclado`. Using a parenthesis or a '$' sign, more commands can be added to continue the modification of the instrument, for example: `volumen 1 ((acompañamiento (2 4)) teclado)`, or, `volumen 1 $ acompañamiento (2 4) teclado`.

2.3.3 Semantics

Seis8s has two types of relations between its commands and the music. The first is tied to the global commands and the instruments as an ensemble, and the second is tied to the action commands and the individual instruments.

Relationships between global commands and the instruments as an ensemble

This relationship is established by Seis8s' instruments, each of which represents a layer to the overall music – which could be seen representing a composition or an ensemble. Some global aspects of this music can be (re)defined as follows.

tempo and compás

The command `tempo` establishes the speed at which the music – or ensemble – will play. This command is very common in computer-music languages. The tempo can be set in beats per minute. The command `compás`, which translates to "measure" or "bar", allows modifying the time signature in Seis8s' music. At the moment of writing, there are two available values for `compás`, namely, `4/4` and `2/2` – also available as `partido`. The default argument is `partido` which translates to "cut" or "cut time". This is a time signature usually used in music that is festive and rapid, such as marches, as well as some forms of Latin dance music, such as Salsa, Merengue, and Cumbia.

```
tempo 120;
compás "2/2";
```

Figure 2.8: Seis8s' `tempo` and `compás` commands

Armonía

Harmonies and melodies in Latin dance music are usually created with the twelve-tone system borrowed from Western European music. In Seis8s, the harmony of the ensemble can be changed with the command `armonía`, affecting all the instruments at once. Chord names – either in syllables or letters – can be input as parameters to this command. At the moment of writing chords with the following qualities are available: major, major7, minor, minor7, fifth, dominant, sus2 and sus4, augmented, diminished, diminished7, and semi-diminished.

```
armonía [do m, re sdim, sol M, fa m];
```

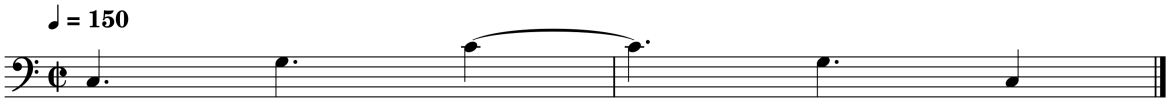
Figure 2.9: A chord progression of C minor, D semi-diminished, G major, and F minor written with syllables
.

Relationships between action commands and the individual instruments

The commands that convey actions to Seis8s' *virtual* instruments relate to actions usually done to their *real-world* equivalents. In Seis8s, this means that some `action` commands will work only over some instruments. For example, `tumbao` will have an effect over `teclado`, `bajo`, and `congas`, but not over `güira`, or `jamblock`. This is because, in Latin dance music, the rhythmic patterns of the güira and jamblock are not colloquially expressed as tumbaos. Rather, a tumbao makes reference to the melodic and rhythmic line that the keyboard, the conga, and/or the bass play within an ensemble – usually of salsa or cumbia music (Ochoa et al., 2017, p. 17).

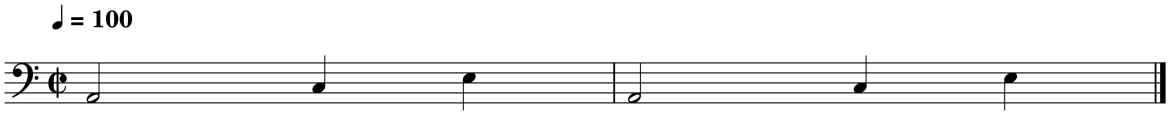
Bass tumbaos

The specificity of `action` commands, like `tumbao`, are also connected to the styles of music played. In the case of a salsa bass guitar, for example, the tumbao is often construed from the first and fifth intervals of the chord in turn. They are also played in a syncopated form (Mauleón, 1993, p. 106). And, in cumbia, the bass tumbao is construed from the chord's first, third, and fifth intervals. And, in its basic form, there is no syncopation.




```
tempo 150;
compás "partido";
acordes [do];
tumbao [1a 5a 8a, 5a  1a] [1 2.5 4, 2.5 4] $ sample 4 $ bajo;
```

Figure 2.10: A salsa tumbao written with traditional notation (top) and with Seis8s (bottom). Adapted from Mauleón (1993, p. 106)




```
tempo 100;
compás "partido";
acordes [la m];
tumbao [1a 3a 5a, 1a 3a 5a] [1 3 4, 1 3 4] $ bajo;
```

Figure 2.11: A cumbia tumbao written with traditional notation (top) and with Seis8s (bottom). Adapted from Martin (n.d)

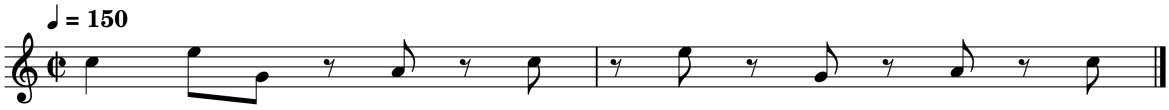
In Seis8s, these relationships are reflected through the tumbao presets, which can be used along with the style commands. At the moment of writing, four presets are available representing rhythmic and melodic variations of the basic cumbia tumbao.

```
acordes [re, la];
tumbao 1 $ cumbia bajo;
```

Figure 2.12: A snippet of code that returns the preset #1 of a cumbia bass tumbao in the chords D major and A major

Piano tumbaos

Tumbaos in the piano are also identified as *montunos*, describing a "syncopated piano vamp [that] provide[s] strong support for the melodic instruments and/or vocalists in an ensemble" (Mauleón, 1993, p. 118). Other musics like cumbia do not have distinct tumbaos or montunos as they often borrow them from Salsa.




```
tempo 150;
acordes [do];
compás "partido";
punteo [1a 3a (5a (-1)) (6a (-1)) 1a, 3a (5a (-1)) (6a (-1)) 1a]
       [1 2 2.5 3.5 4.5, 1.5 2.5 3.5 4.5] $ teclado;
```

Figure 2.13: A salsa tumbao written with traditional notation (above) and with Seis8s (below). Adapted from Martin (n.d)

Similarly to bass tumbaos, piano/keyboard tumbaos can be used along with the style commands. Four presets are available for the cumbia style at the moment of writing.

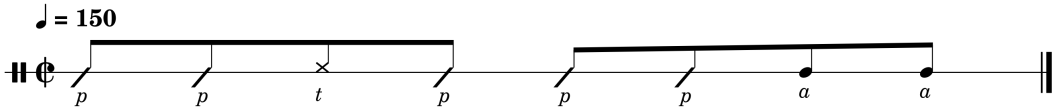
```
acordes [do];
tumbao 2 $ cumbia teclado;
```

Figure 2.14: A snippet of code the returns the preset #2 for a cumbia keyboard tumbao in the chord C major

Conga tumbaos

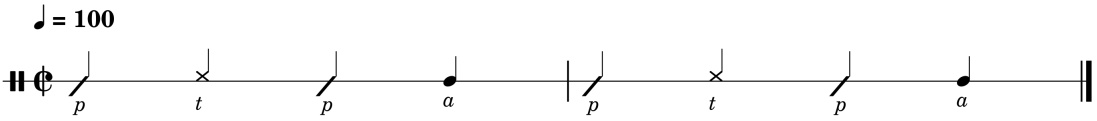
Tumbaos in the congas are also distinct in many of these musics. In Salsa music, for example, a tumbao is "an eighth-note, one measure pattern which accents beat 2 with a slap and beats 4 and 4+ with open tones; the other notes are produced by a 'heel-toe' pattern of the opposing

hand, also referred to as marcha (march)" (Mauleón, 1993, p. 66). In cumbia, the basic conga tumbao is a quarter note, one measure pattern which accents beats two and four with either a slap or an open tone. In Seis8s, the user can select how the congas are hit with the hands. Here, the parameters for `tumbao` are `p` which stands for palma (palm/heel-toe hit), `t` for tapado (muted/slap hit), and `a` for abierto (open hit).




```
tempo 150;
compás "partido";
tumbao [p p t p p p a a] [1 1.5 2 2.5 3 3.5 4 4.5] $ congas
```

Figure 2.15: A salsa tumbao in the congas in traditional notation (above) and in Seis8s (below). Adapted from Mauleón (1993, p. 66)




```
tempo 100;
compás "partido";
tumbao [p t p a, p t p a] [1 2 3 4, 1 2 3 4] $ congas
```

Figure 2.16: A cumbia tumbao in the congas written in traditional notation (above) and in Seis8s (below)

It is also possible to select which drum of the congas to be played.

```
tumbao [p (t a) p (t a)] [1 2 3 4] $ congas;
```

Figure 2.17: A conga pattern in Seis8s, where the first and third hits produce a sound of the quinto drum and where the second and fourth sounds are a tumbadora (t) drum

Finally, conga tumbaos can be used along with the style commands. Four presets are available for the cumbia style at the moment of writing.

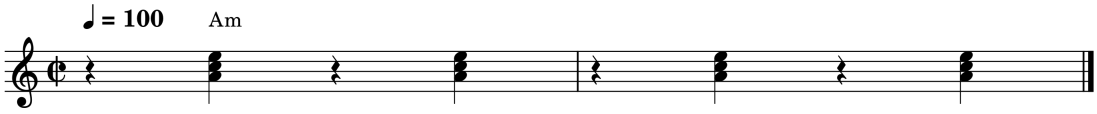
Acompañamientos

An acompañamiento which translates to *accompaniment* makes reference to blocks of notes or chords that provide harmonic and rhythmic support to melodic instruments and which are played "over the chord progression" (Mauleón, 1993, p. 147). Instruments that provide this type of accompaniment in Latin dance music are often plucked instruments and keyboards. At the moment of writing, accompaniments for the keyboard are available.


```
tempo 150;
compás "partido";

acordes 2 [(do maj7 (0 0.45)), (re m (0.45 1.25)), (sol dom7 (1.25 2))];
acompañamiento [1 2.5 4.5, 2.5] $ teclado;
```

Figure 2.18: A salsa accompaniment in the keyboard (top) and in Seis8s (bottom). Adapted from Mauleón (1993, p. 147)




```
tempo 100;
compás "partido";
acordes [la m];
acompañamiento (2 4) $ teclado;
```

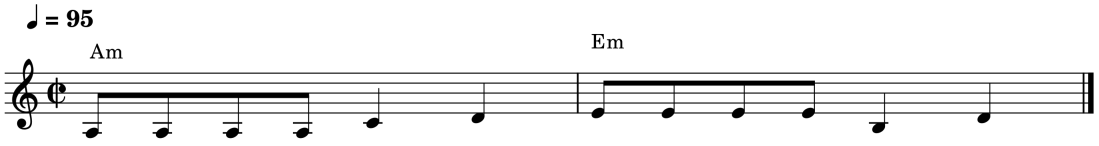
Figure 2.19: A cumbia accompaniment in the keyboard (top) and in Seis8s (bottom). Adapted from Avila and Madariaga (2009, p. 34)

.

Punteos

Punteos, which translates to *picking*, refers to the melody and/or melodic solos in Latin music. This technique is not particular to Latin music, as one can find them in rock and jazz

music for example. In genres like Cuban music, punteos often refer to the melodies made by the player of a plucked instrument such as the *tres cubano*. In other musics such as Mexican cumbia, punteo is also often used to describe the principal melodies made by the keyboard player. The scales used for punteos vary but one that is common in cumbia is the pentatonic scale.




```
tempo 95;
acordes [1a m, mi m];
compas "partido";

punteo [(1a (-1)) (1a (-1)) (1a (-1)) (1a (-1)) (2M (-1)) (4a (-1)), 1a 1a 1a 1a (5a (-1)) (7a (-1))]
[1 1.5 2 2.5 3 4, 1 1.5 2 2.5 3 4] $ cumbia acordeón;
```

Figure 2.20: A punteo fragment from the cumbia "Los Pajaritos" in traditional notation (top) and in Seis8s (bottom). Adapted from Guerrero (n.d)

.

2.4 Seis8s as text and cultural object

In this section, I would like to reflect on Seis8s' explorations stated in the introduction of this chapter, some of which could be seen as intended *properties* of the language. That is qualities that extend beyond functional ones [5] and that are, furthermore, *readable* or subject to interpretation. These include, for example, the use of Spanish constructs to appeal to an imagined community in/from Latin America with cultural, political, economic, and historical commonalities.

[5] In computing, there is a difference between functional and extra-functional qualities. Functional are things such as features, which is "a beneficial capability of a piece of software". Extra functional qualities are things like a property which is "an editable or read-only parameter associated with an application, component or class, or the value of such a parameter" ("Feature vs Property - What's the difference?", n.d).

Spanish constructs from Latin dance music

The Spanish language is used in almost two-thirds of the countries located in what is known as Latin America and the Caribbean. Thus, it is common sense to think that there would be an appeal of some kind for the various inhabitants of this region, particularly programmers and *live coders* of which there are many. There is, however, a lot of variation in Spanish in *Hispanomerica*, thus, it is worth inquiring more about the Spanish being used in Seis8s.

On one hand, Seis8s' linguistic aspect has been conceptualized mostly by the author of this dissertation, making it to infer that Seis8s' constructs are, however, drawn from Mexican Spanish. What could this mean? Perhaps it means that Seis8s' commands are closer to the *neutral* Spanish mentioned in the previous chapter. A type of Spanish where the intention is for its vocabulary and accent to be untraceable to a particular country, but which in reality has a noticeable influence from the Spanish spoken in Mexico City.

On the other hand, Seis8s commands such as `tumbao` and `punteo` borrow from musicians' lingo of globally distributed music like salsa and cumbia. How accurate is this terminology, though? Well, I learned some of this musical terminology during my undergraduate studies in Mexico. And more recently, I learned it from YouTube videos, and Salsa and Cumbia textbooks produced in places such as Colombia, Mexico, and the US. It is possible to then argue that the mediated aspect of my knowledge reflects in Seis8s a globalized, yet not necessarily accurate state of Latin dance music.

Legibility and pedagogy from live coding practice

The intention behind Seis8s' code is for it to be *read* by users, most of which are expected to be performers and audiences. Furthermore, Seis8s' code intends to invoke shared experiences in them, such as the one of speaking Spanish, as discussed above. This train of thought suggests a pedagogical expectation in Seis8s that aids in making legible cultural layers in music programming. However, two important conditions have to be met for this pedagogical approach. First, the performer has to share their screen with the audience. Second, the

audience has to pay attention to the screen.

Where do these conditions have to be met, though? Well, Seis8s draws from live coding practice where audiovisual outputs are performed live using code, and where sharing the performer's screen is strongly encouraged. Here, sharing the screen intends to be pedagogical in at least two ways. First, it looks to demystify programming practices by allowing the audience to see the code that takes place behind computer programs. Second, by demystifying the artist's mind by seeing the decisions they take during the performance.

It is possible, then, that Seis8s' legibility mostly makes sense among live coding communities as opposed to other computer music, where screen sharing is more uncommon. Even within live coding practice, it is currently a matter of debate whether the audience really reads, and furthermore, interprets the code during live performances. The latter is due to factors that make the code unreadable such as the dim lighting of the venues, bad contrast between the text and the background, and, the overall discreteness and abstractness of the code.

All this, particularly the last point, prompts that Seis8s' legibility seems dependent on the artist and the audience to be very specialized. They must be acquainted with Spanish and live coding practice and have a cognitive capacity to engage critically with information technologies. Nonetheless, it is worth remembering that in addition to live coding performance, other avenues for Seis8s' to become legible include workshops, written tutorials, and publications such as this dissertation. I will discuss some of these avenues in the following chapters.

Commonalities within the imagined community

At this point, it is worthwhile to continue pinpointing the intended users of Seis8s. So far, we know that a portion of this community is performers and audience members who attend Spanish-speaking live-coding spaces. The latter looks like a particular membership; however, the emphasis on the Spanish-speaking quality of the users strikes a potential connection with

the concept of *South-South cooperation*. This is the "cooperative exchange of experiences among countries that share historical realities and that have similar challenges" (United Nations, 2019, para. 11). The cooperative exchange intends to transform inhabitants' lives through "knowledge sharing, technology transfer, emergency response and livelihood recovery led by the South" (para. 19).

South-South cooperation contrasts with *North-South cooperation* where sharing practices intend to "make reference to the social, economic and political difference that exists between developed [and developing] countries" (para. 12). Seis8s then seems to reflect the former through its intention of fostering critical engagement with code in Latin America. And it is through this perspective that, perhaps, Seis8s' audience has the potential to extend into a wider public interested in the investigation of information technologies in and from Latin America.

In addition to Spanish-speaking users, Seis8s' imagined community emphasizes those who engage with live coding spaces. What does a live coding space imply, though? From my experience, it implies a certain separation from other electronic-music spaces. For example, the space of the *music producer*, who produces music structured around the capacity to record with digital workstations. In my experience, the latter contrasts with live coding's emphasis on a type of music production with less rigid, improvisational structures, that additionally intend to be ephemeral through the erasure of the program at the end of the performance.

But how are these live coding spaces constructed? Who are the people that compose them? In my experience, they are composed of musicians, sound engineers, media artists, visual artists, dancers, web developers, and programmers of all sorts. However, I have also encountered psychologists, journalists, historians, anthropologists, and sociologists who are interested in the digital cultures being formed through live coding practice. And it is, perhaps, this anthropological and sociological space that Seis8s points to, as I discuss below.

Commands as facilitators of dissent

In the background section of this chapter, I provided two dichotomic views of postmodernity by Argentinian anthropologist Nestor Canclini (2005). Now, I would like to conclude this chapter with a third, more negotiated definition:

> *"... we conceive of postmodernity not as a stage or tendency that replaces the modern world, but rather as a way of problematizing the equivocal links that the latter has formed with the traditions it tried to exclude or overcome in constituting itself. The postmodern relativization of all fundamentalism or evolutionism facilitates revision of the separation between the cultured, the popular, and the mass-based, upon which modernity still attempts to base itself, and elaboration of a more open way of thinking that includes the interactions and integration among levels, genres, and forms of collective sensibility" (9).*

When ideating Seis8s, I wanted to experiment by embedding certain music metaphors in the language, expecting users to ask about them and even challenge them. What I call metaphors can also be thought of as *boundary objects*. These "are a sort of arrangement that allows different groups to work together without consensus" (Leigh Star, 2010, 602). And, where the arrangement or *object* "derives from action, not from a sense of prefabricated stuff or 'thing'-ness" (603) [6].

In Seis8s, the *style commands* and particularly the `cumbia` command exemplify the idea of *boundary objects* as follows. Generally speaking, Cumbia is a term that references the mid-20th century Colombian Caribbean music that, at the time, reached most countries in the region and was adopted by their different cultures. Through this process, Cumbia developed

[6]The concept of *boundary object* was developed by American sociologist and information systems scholar Susan Leigh Star. A boundary object is the idea that people can interpret objects differently. This interpretive quality of objects is also called "interpretive flexibility" and is commonly found in sociology, history, and philosophy (Leigh Star, 2010, p. 602). Leigh Star addresses this concept from the sociology of computer science, where a boundary object is "something people ... act toward and with" (e.g., a program) (p. 603). That is to say, the object's materiality is "derived from action" (e.g., it is "embodied, voiced, printed, danced, and named" without the need of a previous agreement) (p. 603)

into many local variants to become a *generic* concept in everyday speech, encompassing a vast range of instrumentation, musical forms, lyrics, and political intentions.

This way, one can say that there is not a fixed essence of Cumbia, nor is it an irreducible phenomenon. And at the same time, it is recognized in the collective imagination as a result of migratory flows and lived experiences among the inhabitants of 20th-century Latin America. In the next chapter, I discuss some of these lived experiences through interviews with people from/in Latin America to explore how understandings of Cumbia and other Latin dance musical styles have shifted with changing political fortunes in Latin America.

2.5 Conclusion

This chapter discussed Seis8s, a web-based live-coding language that allows real-time interaction with digital audio and localized musical knowledge. Seis8s is mainly oriented toward Latin Dance Music, where commands and their resulting sounds revolve around instruments of such music. These included `acompañamiento` (accompaniment), `ritmo` (rhythm), `punteo` (picking), and `tumbao` (i.e., rhythmic pattern), among others.

Seis8s' motivation comes from my interest in addressing the cultural mismatches presented to me when creating and teaching music using computer-music languages. During my membership with the *audio atelier* at the Mexican National Centre for the Arts, I experienced the construction of this institution's artistic and techno-scientific identity through a strong emphasis on Western art music. This experience was rich but left me wondering what exploring *regional* popular music through live coding practice would have looked like.

Through the creation of Seis8s, I now have the opportunity to start exploring the aforementioned. Initial explorations include ideating *properties* in Seis8s that extend beyond functional ones and that, furthermore, are *readable* or subject to interpretation. These properties include Spanish constructs borrowed from Latin dance music and my lived experience as a Mexican citizen. These constructs, furthermore, could function as *boundary objects* that

help facilitate discussion and dissent about Latin dance music.

Finally, in this chapter, I tried to pinpoint the imagined community that Seis8s intends to appeal to. From a zoomed-out perspective, this community is formed by performers and audience members of Spanish-speaking live coding spaces. However, through a zoomed-in approach, one can see that Seis8s can potentially engage with a broader audience who is, for example, interested in studying digital cultures. Or is interested in simply belonging to the digital cultures that engaged with live coding.

The latter include musicians, sound engineers, media artists, visual artists, dancers, web developers, and programmers of all sorts. The former include psychologists, journalists, historians, anthropologists, and sociologists. In the next chapter, I discuss various perspectives on music and software by some of the aforementioned members of this imagined community.

Chapter 3

Seis8s in context: talking circles on music and software

This chapter reports on a series of online talking circles held from August to November 2020, where people from Latin America were invited to discuss software and music in connection with sociocultural aspects specific to them and their locality. In these meetings, I invited people to 1) discuss their perception of the interaction of gender, ethnicity, social class, and national identity with music and music software in Latin America. And 2) to experiment collectively with Seis8s to build community and generate ideas for its further development.

The final purpose of these circles was to generate discussion about the sociocultural circumstances intersecting with the development and use of Seis8s. This chapter starts with a short description of the circles, including where they took place, the number of people involved, and the methodology used for gathering and analyzing the participants' contributions. Then, I analyze the emerging themes through an interpretive method, putting participants' comments into conversation with critical theory.

The themes discussed in this chapter are categorized into two parts. Part 1 offers a discussion about the relationship between software and concepts such as *mestizaje*, *race*, *multiculturalism*, *pluriculturalism*, and *sexual orientation*. Part 2 of this chapter offers conversations around topics of cultural appropriation in Latin music, touching upon terms such

as *latinidad* and *sophistication* in listening practices. This section ends with a brief discussion on the sexual dimensions of genres such as Cumbia.

Broad questions guiding the discussion include: *What are some sensibilities, moral systems, belief systems, ways of knowing, and being of Latin American communities? What are the generalized difficulties and forms of resistance of these communities? How do these sensibilities, difficulties, and resistance intersect with music and software? How can knowledge from these sensibilities, difficulties, and resistance be used to generate new design principles in music software?*

3.1 Description of the circles

A talking circle is a dialogical practice where people can reflect and comment on what others have said in the group. This practice, which derives from the wisdom and tradition of Indigenous communities, fosters horizontal conversations where the participants themselves can organize and curate the themes to be discussed. Talking circles are used in places such as the United States, Mexico, Chile, and Costa Rica as tools for self-awareness, anti-oppression, and pedagogical practice (Malo de Molina, 2004; Arias, 2012; Moncada and Acebedo, 2017; OEI Chile; 2017).

The circles this chapter reports on happened in partnership with the Factory Media Centre (FMC), a non-profit and small gallery located in downtown Hamilton that provides physical space and technological resources to artists of the region. Participants were convened through the FMC's social networks and my personal Facebook and Twitter accounts using an enrolment poster in Spanish and another in English. Through these posters, people with no required expertise were invited to engage with either Latin dance music, electronic music, music software, and/or live coding as well as in issues of gender, ethnicity, social class, and/or national identity in Latin America.

The circles started in August and ended in November 2020, and because of COVID-19,

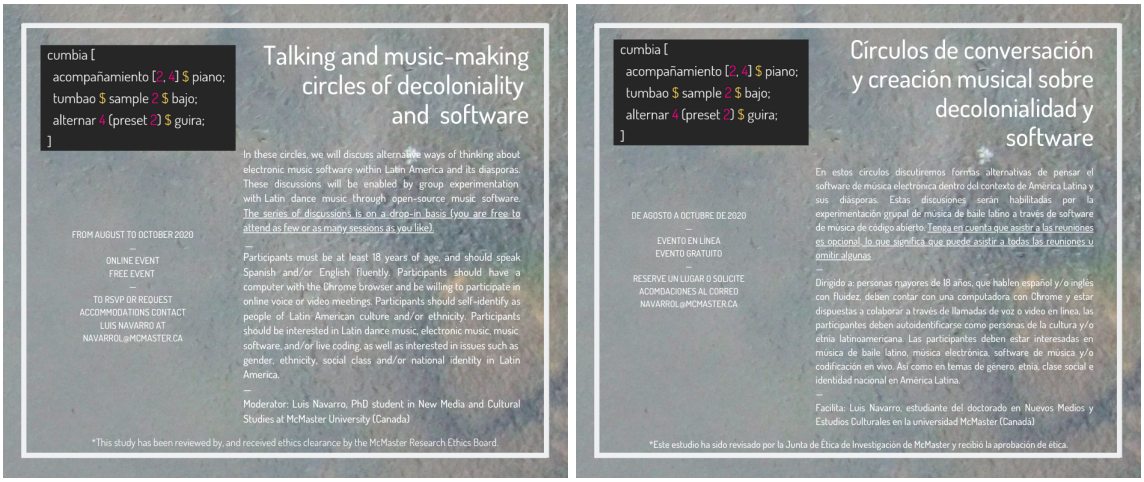


Figure 3.1: Recruitment posters

they were held online through the videoconferencing platform Webex. They happened on a drop-in basis, where people could attend as few or as many sessions as they liked. There were ten meetings of two hours and thirty minutes each.

Participants were mainly Spanish speakers who self-identified as being 18 and above. They were mainly from South America, but some lived abroad in places such as the United States, Canada, and Spain. Participants were divided into two groups, one meeting on Thursdays and another meeting on Saturdays. Based on the discussions, participants from both groups identified mostly as either men or women. Women were the majority in the group on Thursdays, whereas men and women were equally distributed in the Saturday group.

Most of the participants chose not to disclose their sexual identity. I perceived that, generally, the participants saw themselves as part of a community that was not economically marginal. Still, some participants did identify as marginalized and/or peripheral in terms of migration, citizenship, sexual orientation, and/or gender within the place they were from and/or they were currently living.

The Thursday group was intended to be for Latin American people living anywhere. This group welcomed a total of 23 people, of which about one-third attended all the sessions. In this group, people already had a great amount of experience with either music software, live coding practice, and/or music performance.

The Saturday group was intended for people located in Hamilton, Canada, and surrounding areas. By focusing on people living there, I intended to connect my research with the place I was living at the time, the location of my university, and thus where my knowledge was being produced. In the beginning, only four people, including myself, attended. After a couple of meetings, two additional participants transferred from the Thursday group because they could not continue attending on Thursday due to time conflicts.

Participants from this group were also musicians and sound artists; however, they did not have much experience with computer-music or live coding. This allowed me to facilitate explorations with Seis8s to generate shared musical moments, which led to follow-up discussions. In addition to the discussion, this group and I put together a musical performance at the FMC. This performance is described in the next chapter.

3.2 Methodology for gathering and analyzing participants' contributions

Before meeting with the participants, I worked closely with McMaster Research Ethics Board (MREB) to prepare an ethics protocol with procedures I would take to secure the participants' well-being and engagement. The first steps included defining the method, i.e., talking circles, and drafting the broad questions mentioned in this chapter's introduction. Another step was the creation of the recruiting poster shown above and the inviting of two initial co-moderators to lead the sessions. Finally, a necessary procedure was for participants' contributions to be de-identified to protect their privacy.

During the sessions, I shared materials such as texts, videos, and audio with participants to encourage conversation. These materials guided participants and me through issues related to identity, sexual orientation, language, capitalism, colonialism, extractivism, cultural appropriation, feminism, queerness, decoloniality, anti-coloniality, multiculturalism, pluriculturalism, philosophy, and the music market. A detailed description of the themes and

materials discussed per session is given in Appendices A.1 and A.2.

Additionally, I invited participants to share materials that we could all comment on (e.g. articles, videos, reviews, etc). Participants were also encouraged to share their artistic work with the group. In my opinion, the latter worked well as participants could take the role of both lead presenters and listeners. They also connected their thoughts to personal artifacts, such as their artworks, to talk about them in connection with the broader issues of interest.

Participants' contributions were gathered in the form of notes, audio recordings, and summaries generated. Some notes were taken in the form of real-time transcriptions, where my goal was to collect the exact words people were saying rather than focusing on their interpretation or analysis. My typing speed has limitations, and thus parts of these transcriptions might be incomplete or more inaccurate. On the other hand, the audio recordings were transcribed word by word and reflect more accurately what people said.

The summaries were generated the same week after each meeting to capture the overall themes and dynamics that happened. These summaries also served as a refresher of the state of the discussion, and because of this, they were shared with the participants before the next meetings. These also represent an initial analysis stage, which involved selecting the most important things people said. When making these summaries, I also identified keywords and short phrases that could describe the sub-themes of our discussions.

Additional keywords and phrases were included after transcribing the audio recordings. These keywords and phrases were matched up with similar ones to identify relations and broader categories. From these categories and related keywords and phrases, I went back to the summaries, notes, and transcriptions to pull out topical themes, which in this case, represent how participants perceived, felt, behaved, reacted, and generally mentioned in regards to the main themes, questions, and suggested materials.

Finally, from these topical themes, I identified the conceptual or overarching themes. Please see Appendix A.3 for these themes. Identifying these themes allowed me to better connect the participants' perspectives with the interaction of gender, ethnicity, social class,

and national identity with music and music software in Latin America. In the next section, I discuss some of these themes, weaving together participants' observations with scholarship on the topic. Please note that the translation of the participants' quotes is mine.

3.3 Part 1: Software in Latin America

3.3.1 What is software?

At our first meeting, one participant asked a key question that in my opinion helped establish common knowledge and common reflections. That question was "what is software?". Participants started with technical-oriented responses such as:

> "The abstract logical part [of a computer]".
>
> "It is what is inside the hardware, [which is] the physical part".
>
> "Software could be a connection of circuits".
>
> "In popular terms, software refers to a graphical interface – a program on a computer".

Then they moved to more conceptual definitions of it:

> "In Spanish Wikipedia, they define it as a masculine term".
>
> "The soul of things?"
>
> "[A] matrix of a series of multi-connections".

Participants also connected the idea of software with the one of a recipe or an algorithm, mentioning software is related to instructions to create something. Someone made an interesting distinction between algorithm and software, where the former is connected to labor. In contrast, the latter is more connected to mass production and to a capitalist system. Following this train of thought, it was said

> "... we all have certain algorithmic behaviors, we follow steps, we have a daily life. This is assembled to result in a larger gear that is capital".

Participants also posed questions related to software's political and conceptual aspects and the infrastructures behind it.

> "What are the political understandings behind software?"
>
> "Who creates software?"
>
> "Who distributes it?"
>
> "What constitutes the logic behind software?"

Finally, someone suggested being aware of the distinction between free and private software,

> "We are focusing a lot on software [in relation] to capitalism. What about software that doesn't fit in this box?"

To which someone answered,

> "Open source or free software is also synonymous with prestige. Free software doesn't escape these same capitalist structures. Good intentions can be corrupted". And someone made the follow-up question, "Is free software created in Universities exempt from the latter?"

This discussion about software, in addition to setting up common knowledge for all the participants, also showed that participants were potentially more interested in investigating it than other topics included in the poster description — such as electronic music, Latin dance music, and live coding. Participants' interest in discussing software could be connected to the difficulty of finding spaces to talk about the social aspects of software design. In my experience, it is more common to find spaces to talk about the social aspects of practices or spaces where software is used— for example, electronic music-making and performance.

This lack of space to discuss software design from a social perspective was also pointed out by one of the participants who talked about the need for software developers to enable discussions about software design in conjunction with the users. For example, participant J02 mentioned,

> "Spaces for designing together are important, but spaces like these circles call my attention more as there is a potentiality where many reflections can be shared."

3.3.2 Parallel coexistence, delinking digital technology

Mestizaje was a recurrent topic in both Thursday and Saturday meetings and was introduced by the following participant's questions,

> "How decolonial can one become if the tools that we use are produced by the colonial? And, where does mestizaje come in? Is mestizaje a contradiction when we work with technologies in Latin America?".

Anthropologist Peter Wade (2003) mentions that *mestizaje*, in its nationalist form, refers to an ideology where the foundations of the nation are thought to be a "mixture that occurred during the colonial era between African, Indigenous, and European people" (p. 275, translation mine). Wade describes how this national identity was sometimes embraced in response to nations such as the United States, "where racial mixing was taboo and racism against black and indigenous populations was notorious" (p. 275, translation mine). Is in this way that during the 20th century, "[c]elebrating mestizaje became an assertion of the supposed Latin American racial democracy" (p. 275, translation mine).

The phenomenon of *mestizaje* happened in different countries of Latin America in diverse ways but with the shared "emphasis [that was] placed on the Indigenous or the Afro" and with the constant that "the white always receive[d] attention" (p. 277, translation mine). Wade, coinciding with other various authors, views mestizaje in general terms "as a nationalist

ideology ... more or less disguised as whitening, both in physical and cultural terms" (p. 277, translation mine).

However, Wade considers,

> "... mestizaje is more complex [and] that multiple mestizajes can be thought of and ... in order to understand why the idea of mestizaje and being mestizo or the product of a mixing process is so deeply rooted in the populations and the imaginary of the nations in Latin America, it is necessary to go beyond the idea of mestizaje as a process of disguised exclusion" (p. 277, translation mine) [1].

Because of the latter, it seemed to me that it was important to uncover the participants' position toward mestizaje, so I decided to inquire more. First, participant J05 gave their point of view by making reference to the concept of *ch'ixi* or *cheje* from Bolivian sociologist Silvia Rivera Cusicanqui. In an interview shared by this participant, Rivera Cusicanqui defines it as:

> "The notion of ch'ixi is equivalent to ... [the] concept of "mottled society", which expresses the parallel coexistence of multiple cultural differences that do not extinguish but antagonize and complement each other" (TV UNAM, 2018).

This concept of cheje seemed to be familiar to other participants. For example, J05 described it as a "decolonized mestizaje ... as a color born from contradiction". The latter was complemented by participant J14 who mentioned,

> "... we could see history without so many veils, which are not completely black or white, but much more complex. [We should] understand the idea of mestizaje ... as the opposite of the pure, as contradictions".

[1] "Another dimension of the variation in ideologies about mestizaje is the diverse importance given to the indigenous and African contribution to thinking about the imaginary of the nation. In some countries, mestizaje has been thought with reference above all to the mixture between whites and indigenous people - for example, in Peru ... while in others the basic reference has been the mixture between whites and Africans - for example, Cuba" (Wade, 2003, p. 276, translation mine).

From my perspective, the reflections above regarding *mestizaje* pointed to participants' willingness to challenge nationalistic understandings of identity in Latin America. Moreover, they challenged these understandings with care by trying not to demonize their identity and avoid creating feelings of guilt within the group. They achieved this by focusing on ways of resignifying the ideologies that formed their identity rather than on suppressing their identity itself.

For example, participant J05 invited us to search for a resignification of this concept outside of the "white mestizo" scheme,

> "How to give a resignification, no longer thinking about the concept of the white mestizo, but closer to what we are trying to define?".

The term *white mestizo* struck me as novel as I have not heard it before. However, I think the term could be explained by Chicana Activist Gloria Anzaldúa (1987/2001). Anzaldúa critiques the 20th-century nationalistic ideologies in Mexico and the United States,

> "... commonly held beliefs of the white culture attack commonly held beliefs of the Mexican culture, and both attack commonly held beliefs of the indigenous culture ... [because of this, the] new mestiza copes by developing tolerance for contradictions, a tolerance for ambiguity. She learns to be Indian in Mexican culture, to be Mexican from an Anglo point of view" (94-95).

Delinking digital technologies

I would like to return to the questions mentioned by participant J12 at the beginning of this section where it was asked how mestizaje influences people living in Latin America when creating with technology. Here, I believe participant J12 was inviting us to delink from the idea of the use of digital technology —such as software— as an oppressive or a colonizing aspect. The latter, as this same participant suggested,

> "seeing 'Eurocentric' technologies as part of non-Western knowledge ... to begin rescuing the non-western from the western".

J12 furthered this by reflecting on ideas —or technologies— from non-Western cultures that have influenced Western thought such as "the non-Western mathematics ... that has allowed us to make computers ...".

My interpretation of J12's last comments is that *mestizaje*, seen as a series of contradictions, is not only particular to places such as Latin American countries. The latter, as contradictions, can also be found in places from Western countries within their tools and technologies. The latter is not only valid for digital technologies but also for ideas and ideologies. An example of this is the exploration of the mestiza identity in Donna Haraway's Cyborg Manifesto (1985/2016), through the concept of the "oppositional consciousness" by Chicana scholar Chela Sandoval (1981).

In her essay, Haraway looks at Sandoval's scholarship which describes a mode of political thinking and being that looks to *oppose* or transgress the "dominant social order" and the rigid categories it imposes (Sandoval, 1991/2013, 54,5). One of these modes is called the *differential mode*, and it is drawn upon Gloria Anzaldúa's 'Chicana mestizaje', which proposes a "form of consciousness made up of transversions and crossings ... between races, genders, sexes, cultures, languages, and nations (Sandoval, 1998, 352).

In this sense, I believe that some participants' perspective was that the usage of digital technologies in Latin America and their tensions towards being 'foreign', should not be connected to the false sense of purity that national understandings of *mestizaje* propose. Rather, it should be seen as part of the diverse cultural artifacts that coexist, complement, and antagonize within Latin American societies.

Software bricolage

Other themes that often emerged in our talking circles were about the asymmetric relations on top of which software is made. These topics came up first from a question by participant

J04 in regards to programming languages,

> "How to dialogue with these languages that are free software but are still directed and that in their essence have a lot of [the] first world?".

Five meetings after, participant J02 connected the term mestizaje to the term scarcity,

> "What do you think about technological mestizaje? Have you found it somewhere? What does it sound like to you? ... [In terms of mixing] old and new technologies so to speak... due to scarcity ...?

To which participant J09 responded with the term of precarity by saying,

> "... the word mestizaje is associated with the biological and [the] cultural ... if there is a cultural mestizaje then there is also a technological mestizaje. Although it can also be seen from the most negative point of view... I also know people who work from precariousness and in that, there is a possibility of developing with few elements [where] the answers do not depend on my ability but on my possibility to go to a store".

An interesting thing was that both questions and comments above were not directly connected to mestizaje as a contradiction but more to a mixing of biological, cultural, and technological materials and practices. The latter makes me think that mestizaje in this context might actually have a relation to the French word *bricolage*. *Bricolage* is a term originally formulated by French anthropologist Lévi-Strauss as "a survival of ancestral ways of thinking and doing which persists in certain everyday practices of our modern industrial civilization" (Johnson, 2012, p. 357). Lévi-Strauss mentions,

> "... in our own time the 'bricoleur' is still someone who works with his hands and uses devious means [des moyens détournés, indirect or roundabout means] compared to those of a craftsman [homme de l'art]" (as cited in Johnson, 2012, p. 358).

For Lévi-Strauss the *bricoleur* was not "exactly coincident with the contemporary definition of 'handyman' or 'handywoman', 'DIY man' or 'DIY enthusiast" nor was it with engineer (360). Furthermore, it is stated that what is produced by the bricoleur through bricolage was not "a replicable object" (361).

In this sense, words such as scarcity, precarity, mestizaje and bricolage, might point to a form or circumstance from and in which things can be produced under an unofficial and non-mass production ideology.

3.3.3 The concepts of race, multiculturalism, and pluriculturalism in software

From the responses shown in the previous section, I was able to feel a consensus where the participants wanted to detach *mestizaje* from the dominant national understandings of their respective countries. I also felt they were pushing for a resignification of the concept toward a less universalized cultural experience or even an ideal for people living in Latin America.

The latter is likely to be connected with shifts in public discourses after the 1990s when countries in Latin America started to recognize themselves as *pluricultural* and where they started to recognize that Indigenous peoples living there have the right to their self-determination (Jackson and Warren, 2005) [2].

Once the participants and I had commenced challenging national identities imposed by our nation-states, I wanted to know to what extent participants have reflected on the cultural differences and identities that other ethnicities in their countries have.

My interest extended to related concepts such as *race*, which was also new to me until

[2]For example, Mexican scholar Ortega Villaseñor (2010) describes Mexico's complex process of moving from "a monocultural state to a pluricultural one, that is, from a nation composed of a single ethnic group, a single language, a single right, to one multiethnic, multilingual and of multiple legal regimes" (224). The latter process saw its light for the first time in 1992 when published in the Mexican Constitution. According to Ortega, these policies were the product of the pressure coming from "Indigenous peoples of [Mexico and] the world, in their desire to preserve a freer world, fair and diversified culturally" (248). The latter, he mentions, happened with the aid of international organizations such as the United Nations as well as to an increase in international Indigenous and human rights.

recently. This concept and derivatives — such as 'White' and 'Black' — are often used in research within social sciences and humanities in countries such as Canada and the United States. It is also frequently used in the news, TV shows, and day-to-day conversations in that country.

On the other hand —and despite Mexican historian, Velázquez Gutiérrez's (2020) assertion that the category of *race* is very used among Mexican intellectuals—, my experience has been the opposite. That is not very used in the social sciences nor in the public sphere in countries such as Mexico, and despite *mestizaje* being a racial concept.

> me: "Is race a valid concept of analysis today?"
>
> J04: "... the researchers [mention that race] is not constituted by natural or biological concepts, but is constituted by social facts and is configured by contexts, and this can have effects of power relations and of subjection. This is in relation to blackness, and I have also heard the concept of Indianity, on the one hand. So, if it is valid or not? As a biological or natural fact, it is no longer valid, but it does have cultural effects that should not be ignored".

It was also mentioned that sometimes terms related to the concept of 'race' are appropriated by racialized groups as a process of empowerment,

> J09: "...these terms appear from an appropriation of meaning by the racialized groups themselves... One thing that resonates with me regarding this scientific qualification, is that it is not a valid concept for measuring something".

Likewise, it was pointed out that these same terms and the concept of 'race' become convenient for the market and that is why they are perpetuated,

> J09: "It resonates with me, first, from this pet word, where someone uses something that was scientifically proven in order to silence or impose a word. The other thing that resonates [with me] is that the market is the one that ends up

influencing who uses those terms and discourses. These are convenient for the market, and so reversing these ideas of difference could be dangerous for the system ... Science also works for the market".

Multiculturalism as technology

Within this same conversation, someone introduced for the first time the term *multiculturalism*, bringing to our attention its instrumental aspect,

> J04: "... this discourse of representation and multiculturalism is violent and ... it is a discourse that is a technology that is redefined and revalued from the nation".
>
> me: "Can you explain to us what is the problem with multiculturalism?".
>
> J04: "A little bit of the problem of assuming ourselves as mestizos or being sheltered under [that] governance scheme is that there are other groups ... who do not assume this [identity] ... who do not want to recognize themselves as mestizos [and] then they are no longer represented by these governments... So they are definitely not all represented. So if they move [from this representation], they [now] have a complex power relationship where a position of a police state is assumed against these Indigenous groups that do not want to identify with [what the] government proposes".

Three sessions later, we went back to discussing the concepts of multiculturalism and pluriculturalism. To have a better understanding of these terms in the context of Latin America, I suggested reading a text from Ecuadorian Cultural Studies researcher Fernando Garcés (2009). Garcés mentions that multiculturalism is "the simple acknowledgment of the existence of diverse cultures in a given space without making reference to their mutual relationships" (24).

Pluriculturalism is when "the cultural diversity of the countries is seen as a wealth that is incorporated into the white-mestizo political model and structure without questioning or restructuring it. This is the model ... most commonly applied in Latin America and also in the Andean region" (25).

After having read about these terms, we shared some more questions and reflections,

J02: "When is multiculturality confused with westernization?"

Participant J02 told us about central Mexico, where Indigenous Purépecha food sold in the streets and which usually does not have much sugar, now has more sugar so that the mestizo population likes it. For J02 this meant Westernization rather than multiculturalism.

I also offered some comments regarding Canada and Mexico, which are countries where I have lived. On the one hand, I mentioned that multiculturalism in Canada seems to be understood in terms of tolerance and increasing the country's labor force through immigration. On the other hand, immigration is way lower in Mexico, meaning multiculturalism there is connected to countryside migration. This type of migration is, nonetheless, connected to increasing the labor force as well.

In this respect, Canadian political science scholar Murphy (2013), describes how multicultural policies differ worldwide. For example, in Europe, these revolve around "circumstances of immigrants and refugees", whereas "in parts of Latin America [they] encompass the very different sorts of claims and circumstances of indigenous peoples" (30).

In the United States, he continues, "the term is generally not applied to indigenous peoples but it is applied to African Americans and policies geared towards issues of racial difference and racial disadvantage" (30). Finally, he concludes, "Canada's official multiculturalism casts its net even wider to include immigrants and refugees but also historic religious and linguistic minorities and ... Aboriginal peoples and the Quebecois" (30).

Going back to our group discussion participant J04 mentioned,

> "Under the discourse of multiculturalism ... there is an opportunity to recognize oneself within a race, but underneath there are some covert logics".

J04 continued reflecting on these terms,

> "I would like to make an attempt to turn things towards the concept of technology and I wanted to take up some words from [Paul] Preciado. Preciado is inspired by Haraway. [Preciado mentions] the footprint of man has been represented by the control or domination of power, [where the man] understands and categorizes [some cultures as] brute nature that must be domesticated".

Race, multiculturalism, and pluriculturalism in software

The use of *race* and multiculturalism as technologies of categorization connect with what critical studies scholar Tara McPherson (2013) mentions with particular attention to the US. Focusing on the 1960s, she points out the double standard where modes of racism passed from "overt to more covert" as a result of the civil rights movement fighting for Black people's rights (24).

According to her, it was possible that this double standard got embedded in the design philosophy of operating systems such as UNIX, privileging "the discrete, the local, and the specific" (25). Here, designers of UNIX reproduced particular wordings of problems and solutions drawn from discourses of multiculturalism, poststructuralism, and post-Fordism (29).

McPherson makes the latter more concrete by mentioning that the discourse of "neoliberal pluralism" from the 1960s was grounded on hiding the racist core of the US white society, similar to how the UNIX kernel (i.e. the core program that manages "'hardware memory, job execution, and time-sharing") is hidden beneath the shell (i.e., languages) (29). However, McPherson makes clear that the parallels between "racial and political formations" and UNIX design were not conscious ones but systemic ones (30).

McPherson's conversation on computer design is useful as it allows one to extrapolate it into Latin American societies and the software they have been producing – including Seis8s. For example, it allows one to ask in which ways has computer software in the region been inadvertently influenced by national ideologies such as mestizaje. Or, have their philosophies, metaphors, rules, and functions been influenced by the multiculturalism and pluriculturalism particular to this region?

3.3.4 On software orientations

On one meeting from Saturdays, I shared a fragment of the text "Becoming Straight" by Queer theorist Sarah Ahmed (2006). Ahmed begins with Simone de Beauvoir's quote "one is not born but becomes straight" (79). Ahmed then wonders "[w]hat does it mean to posit straightness as about becoming rather than being?" (79).

She mentions that sexual orientation has been framed as either a biological question or "a question of choice" (79). However, the latter and the former are the results of social constructions that "[do] not quite explain the ways in which sexual orientation can be felt as inherent and bodily or even as essential" (79). Because of this, Ahmed invites "to produce explanations of how orientations can operate ... as effects" and at the same time, be lived or "experienced as if they are originary" (80).

Ahmed also says that social construction is also informed by the space and the objects that surround us. To exemplify the latter, Ahmed quotes anthropologist Janet Carsten who narrates how the kitchen table in her home brings her back strong memories not only of cooking and eating but of "family discussions, community work, and ... games" (80). Here, the table allows a certain socialization that keeps people related by the "tangible" (81).

> "The table in its very function as a kinship object might enable forms of gathering that direct us in specific ways or that make some things possible and not others. Gatherings, in other words, are not neutral but directive. In gathering, we may be required to follow specific lines. If families and other social groups gather

> 'around' tables, what does this 'gathering' do? What directions do we take when we gather in this way, by gathering 'around' the table?" (81).

When discussing objects and spaces as orienting social relationships, participant S01 mentioned,

> "... That distinction between object and space seems appropriate to me... I find it appropriate to show that a social domain does not necessarily inform the body. It also happens vice versa, where bodies can take up space and form the social sphere".

In the same manner, Ahmed emphasizes "the role of repeated and habitual actions in shaping bodies and worlds" (2). That is, "[it] is not just that bodies are moved by the orientations they have; rather, the orientations [they] have toward others shape the contours of space by affecting relations of proximity and distance between bodies" (3).

S01 also mentioned,

> "... related to software: the design is not neutral. The very design of the table or of the bodies that occupy space already enables certain processes, right?"

Ahmed coincides with S01 in that space is defined by how our bodies reside there. For example, the rights that bodies have in a certain space define the space itself. Furthermore, how bodies experience "social differences" affects not only their interaction with the space but their interaction with other bodies sharing that same space (5).

Feminist activist Cynthia Cockburn (2004) exemplifies the latter when talking about the spaces of science and technology. Cockburn describes how the over-involvement of men in these fields gets to signify them having control over 'the future' (12). This over-involvement, she says, has made technological research have a

> "male orientation ... [that] has long obscured the significance of 'women's sphere' inventions, and this in turn has served to reinforce the cultural stereotype of technology as an activity appropriate for men" (15).

Social position and software

During our conversations, I questioned how my social position as a heterosexual male would have influenced the design of Seis8s. I asked the participants, how could we do a textual reading of Seis8s to discover the linguistic and non-linguistic signs that are normative in terms of sexuality, to modify them or mitigate them. Participant S01 responded,

> "Is it possible from the straight to design a space to liberate the bodies in a Queer way? I think that's the question right?"

To what participant J02 responded,

> "I think that even for people who identify as Queer it would be difficult because it means unlearning many things. I do believe that spaces are gendered in the sense that they enable certain people to have more power than others. So, I find it very difficult to think how to build something completely different".

J02's comments about unlearning things were connected to Ahmed's (2006) discussion of the familiar and how it is "shaped by actions that reach out toward objects that are already within reach" (7). Here, creating "new impressions" involves the "dynamic negotiation between what is familiar and unfamiliar" (7-8); and in this regard, it might that building new spaces, as J02's suggests, involves working with a mix of things we know (or not have unlearned yet) and things we do have unlearned,

> "Going back to ... the cultural and biological constructions of gender, about how we design certain spaces ... Programming languages is quite a masculine field. Although this was not always the case, we already know that the first people who

started working with programming languages and doing computing were women. So it was a female field".

In connection with J02's comments, we turned our attention to discussing ways to resist this male over-involvement in computing and programming languages. Participant S01 mentioned that a way to do so could be to

> "... reminisce how code design has a female history and perhaps highlight it as a [current] possibility".

In this same vein, J02 ended our conversation by mentioning how education is another way of resisting the patriarchal culture in computation,

> "For me, it has to do with education ... obviously [not by] looking for [a certain] gender to say that you are going to change something because you can be a woman or a man, [as both can] repeat the same capitalist structures ... I think this space where we are, where there are reflections on certain things, calls my attention more ... Many of [us] at some point will teach, so then there is a potentiality in which you [will] share many [of these] reflections. [And this, then becomes] a space where you can change things".

At the end of that session, someone mentioned the importance of asking about how gender and sexuality are represented in Seis8s. For example, participant S01 pointed out the heterosexual dimension of Latin music, particularly cumbia.

3.4 Part 2: Latin dance music

3.4.1 Cultural appropriation in Latin music

At the fourth meeting on Thursdays, I picked up on cultural appropriation. I suggested reading a pop culture article about cultural appropriation and whitening in Latino popular

music that addressed the Spanish singer Rosalía, who sings a mix of Flamenco and urban music like Reggaeton. The article discusses the multiple times Rosalía has been called or has called herself a *Latina* artist, including in the 2019 Grammys and in the article "20 Latino Artists making the world dance" from the magazine Vogue Mexico (Agrelo, 2019).

Participant J07 started by saying,

> "... the truth is that I have heard little about [Rosalía], I have heard about her through friends, and that she has been criticized for cultural appropriation. The article mentions that she won at the Grammys ... [but] that does not respond to a desire for it, but to the market that seeks to sell cultural things. When can we talk about appropriation and when not?".

Participant J05 built upon,

> "She is Catalan from Barcelona and does not have [Roma] roots and she was also accused of appropriating the [Roma] culture and the accent because it is not the way she normally speaks".

Participant J05 redirected their comments by mentioning that in Spain, Latin-American immigrant women have made backlashes against Rosalía as for example, "she has never been detained [or asked for] her [immigration] papers" in that country.

The latter points out that Rosalía potentially benefits from identities that are in a more asymmetric position in relation to the Spanish government. This participant concluded that one way to avoid cultural appropriation is to create around "shared oppressions" rather than upon "other" oppressions.

Farther into that conversation, participant J09 suggested that there was a type of appropriation that involves the transformation of oneself. This participant exemplified this by mentioning that if Rosalía had lived under the socio-economic conditions experienced by people living in Latin America, she might call herself Latina.

This was followed by participant J10 who mentioned,

> "I have heard Spanish people saying they name themselves as Latinas because their language comes from the Latin [language]".

The language of Latinidad

The comments from participants above pointed to important contradictions within terms such as Latin, Latino, Latina, and latinidad. Ethnic and Gender studies professor Juana María Rodríguez explains the "problematics involved in the construction of Latinidad" (2003, 9). She starts by saying,

> "Latinidad serves to define a particular geopolitical experience but it also contains within it the complexities and contradictions of immigration, (post)(neo)colonialism, race, color, legal status, class, nation, language, and the politics of location" (p. 9-10).

Rodríguez continues by asking,

> "So what constitutes latinidad? Who is Latina? Is latinidad in the blood, in a certain geographic space? Is it about language, history, and culture, or is it a certain set of experiences?" (p. 10).

Reflecting on these questions, she talks about the issues that imply defining latinidad through geographical spaces. One of the problems, she says, is that geography is "constructed by history and politics", and often marginalizes "Indigenous ethnic communities throughout the Américas that refuse to be subsumed by occupying nation states" (p. 10).

Something similar was commented on by participant J12,

> "... the term latinoamericano is problematic because if we speak about geography, then where do the native peoples who do not speak Spanish belong? Let's say, the Quechua and Aymara peoples ...".

J12 pointed to heritage and natural language as other contentious ways of defining latinidad. Rodríguez explains that if we try to see latinidad through the lens of a "shared linguistic 'Latin' heritage", then many people and regions will fail to fit into that scheme. This is because there are many non-Latin-derived languages used 'officially and unofficially' in Latin America, such as "Quechua, Aymara, Guaraní, French, Hindi, Creole, Papiamento, and English, to name a few" (p. 14).

Rodríguez continues, "Spain and Portugal also spread their colonial seeds elsewhere, in the Philippines, the Canary Islands, Cape Verde, Morocco, Macao, Mozambique, Guinea Bissau, and Angola, for example" (p. 14). Furthermore, if "[in] México, the mixture of Indigenous and Spanish blood is considered mestizo, ... in the Philippines this same mixture is considered Asian" (p. 14). In the same way, "[t]he cultural fusion of African and Portuguese brought about through enslavement may be considered Latino in Brazil, yet the same combination in Cape Verde is considered . . . something else" (p. 14).

To complicate things more, Rodríguez says,

> "Ironically, several regions of the Iberian Peninsula also make claims to being unwilling subjects of Spanish colonialism and conquest ... Populations in Galiza, Catalunya and Euskal Herria make varying claims for a cultural and linguistic identity separate from an imposed, culturally Castilian, Spanish nationalism, which has historically attempted to consolidate itself precisely through colonialism and conquest" (p. 15).

Cultural appropriation in Caribbean music

Resuming our conversation about cultural appropriation, participant J04 finished with a question that invited us to reflect on the limits of musical exchange,

> ".. one is always determined by the world of production and it is a very homogenizing way of operating. That on the one hand, but on the other, it is interesting

> to ask ourselves when it is hybridization and when it is appropriation ... what happens when there are others who assume and talk with the marimberos of the Pacific and then do some super productions and these 'maestros' are forgotten and at risk [because they are left in] poverty?".

J04's comment pointed to a real situation that has often happened with Caribbean music. Arboleda (2012), for example, mentions that during the 1940s Caribbean music from Colombia such as cumbia and porro, became popular in Mexico. Furthermore, its establishment there, through both Colombian and non-Colombian bands, helped to extend this music all over the Americas.

Figure 3.2: A picture of Mike Laure's album 'Cumbias picosas'

However, within this establishing process, a whitening process also happened through the "de-africanization of the melodies, distilling the 'peasant' performativity in the instrumentation and interpretation" (p. 56). The above can be exemplified through the Mexican musician Mike Laure who in the 1970s was known for mixing Rock n' roll with Caribbean music, creating the "first versions of 'Mexican cumbia'" (p. 57, translation mine). Arboleda explains,

> "Mike Laure was the first Mexican recording cumbia ... transforming it into

> what he or the Mexican record industry said was a more "adequate" sound for the population of those latitudes"(p. 57, translation mine).

Additionally, Laure never "gave credit to the [original] composers of this music, so his public believed these were originally composed by him" (p. 57, translation mine).

3.4.2 Sophisticated listening and instrument substitutions

Critiques about the ways that Caribbean music, particularly cumbia, gets legitimized in different regions in Latin America continued two sessions after. Here, participant J07 remembered Mexican musician Celso Piña — another exponent of cumbia during the 1970s onwards — when talking about the cumbia scene in Monterrey, Mexico.

> J07: "... in Monterrey, Colombian cumbia for many years ... until relatively recently ... was music for the lowest class ... And the case of Celso Piña seems very relevant to me, because ... one of the strongest moments for cumbia in Monterrey to be accepted or to begin to have that acceptance, was when Gabriel García Marquez came and asked Celso Piña to be the one entertaining his presentation or at least the dinner after his presentation".

J07 continued by explaining how after this event, "the intellectual elites from Monterrey started adopting cumbia", switching their perception of cumbia from a "study object" to a "study subject". J07 continued by explaining how this affected Monterrey's society in both good and not-so-good ways. On one hand, people there were able to openly say when they liked a cumbia song or band. On the other hand, there still was a reminiscence of discriminatory dynamics.

> J07: "... even though ... there is already a little more license and more fluency to be able to ... say 'these [sounds are] mine' [or] 'I like this', a curious dynamic continues. Cumbia is a success when it is in a large public space, where there

> is a certainty that you can listen to it without problems. That there is security. That it is a cultural event. But cumbia when it is produced or is heard in the places where it is actually generated, which are the 'barrios' ... is not as 'pretty' anymore".

J07 continued by comparing this type of legitimation to the idea of sophistication, particularly with Punk music in Mexico,

> "... the experience that I have had ... [is that] the first approach to punk ... is with bands that play a [very basic] rhythm. A very monotonous rhythm that is very easy to reproduce, and that anyone can play, but since it is so monotonous it's almost hypnotic. And at the beginning, when you start to listen to this type of music, well, it is good that you say that you like that because it is what you are getting to know. But eventually, it is not so cool that you say you like that because it is very 'primitive', very 'basic'.
>
> When you start to get a little more technique, ... when you start to sophisticate the rhythms and sounds a bit, it's like you're already talking that you have 'good taste', that you have a little more 'knowledge'. ... [You start accessing] other things a little more complex. And, that complexity in the rhythms and in the compositions also comes from the production centers of punk music, which would be the United States, Europe, England mainly, and Japan. But all this sound of the 'basic' and the 'primitive' is being pushed aside".

Responding to J07's comments, participant J04 returned to the topic of sophistication within the acts of music-making and listening,

> "... I found very interesting what [J07] said about that ... sound sophistication perhaps shows that Europe began to be constituted as the domain or as the one that dominated the rest of the world ... This makes me reflect on [the saying

> that]: "we have to sophisticate our music or our ways of listening". But this [sophistication] is in line ... with European heritages ...
>
> Well, it seems interesting to think about ourselves, ... how perhaps ... [we can] unmark ourselves or ... 'decolonize' ourselves from these colonized listenings".

Here the comments from J04 seemed to connect the idea of sophistication with modernization or with the idea of what something modern could be. J04 proposed to read an article from Ecuadorian Mayra Estévez (2015, 57), who investigates the historical and cultural factors that define sound, understood as "the articulation between sound/silence/sound". Estévez says that "sound as a form of 'creative expression' ... has been tied to the imposed exigencies of dominant discourses", such as the discourse of development. Estévez continues,

> "... within this framework, forms of knowing and power articulated to the idea of renewal of art through sound were produced; at the same time, the 'illusion' of development as the detriment of the local, was effectively conveyed" (p. 57).

For Estévez, "sound is conditioned to factors ... such as geography, climate, architecture" but also to "cultural conditions" such as politics and economy (p. 57). Furthermore, she posits power relations as elements that "condition the generation of the worlds of sound" (p. 58). For her, reflecting on the latter was missing within the artistic discourse of the "vanguards of the 20th century", in the same manner, that more obvious practices of development were put into practice critically in Latin America and other parts of the world.

Instrument substitutions

Towards the end of one of these meetings, J04 directed the following questions to Seis8s,

> "In regards to listening and the sophistication of sound, it also accounts for colonizations that we may even have naturalized ... So, thinking about the software ... well in the program that you showed us. It is interesting to think about the

> sophistication of the sounds, what does that imply? And what does this new dynamic imply with your program?"

In response to this important question, I said that my intention is not to convey that Seis8s produces an 'authentic' cumbia, but rather, that it produces music influenced by cumbia — and other musical genres.

I also understood this question as related to instrumentation, where for example, in Mexican cumbia, the congas substitute the more traditional drums from Colombian cumbia. And in this respect, my response to J04 also involved acknowledging that Seis8s' instrumentation responds to a 'whitened' and 'modern' version of cumbia and tropical music.

This type of substitution is not particular to cumbia or Seis8s but to tropical music in general. For example, during the 20th century, Vallenato, a musical expression of Colombian peasant Black people, was slowly transformed from the *traditional* to the *modern* by non-Black bourgeois composers. The latter through the substitution of instruments such as the "accordion, caja, and guacharaca" with "the guitar, maracas, and instruments", which were thought as "less stigmatized" (Blanco Arboleda, 2005, 176).

From a different perspective, I intended to reclaim some of these instruments and rhythms, and I put them at the center of Seis8s as a form of resistance to asymmetric types of abstraction in computing. For example, this asymmetric abstraction happens in drum machines by not giving a central place to specific instruments and rhythms. These instruments and rhythms include, for example, the congas and the Cuban clave, which are often seen as a "flavor" that could be added to rock or pop music.

3.4.3 Kumbia Queer, commands, lyrics, and dance

At another session, I picked up on the topic of Seis8s and the heterosexual dimension of cumbia that some participants had mentioned in some meetings before. I started by summarizing some strategies that participants have been mentioning in relation to making the language feel less heterosexual. Drawing from what participant J02 said about programming

languages and cumbia as spaces that have a gender, other participants suggested that the language continued to be designed in collaboration with other people.

Others suggested reflecting on who normally plays the instruments represented in Seis8s. That is to say, to reflect and ask what is the imaginary revolving around the instruments such as the conga, maracas, bass, etc. We asked, do I imagine myself playing, or a woman, or someone Queer?

As we were preparing for a performance, it was suggested to make the exercise of giving preference to women and Queer people from our group when picking up the instrument they want to experiment with during rehearsals. The latter, we said, trying to avoid an essentialization but to foster a reflection about the bodies that perform these instruments in this music.

Another suggestion was to use lyrics as a way to convey pro-women messages. I said, however, that Seis8s does not have a direct way to convey lyrics. I proposed then that a parallel approach might be achieved through the commands of Seis8s. For example, the command `tumbao` which is a masculine noun in Spanish, could have the alternative option `tumbaa` which might correspond to a feminized version of that noun. Or, we could have `tumbax`, which would correspond to a neutral version of that same noun and command [3].

The strategies above connect with feminist reflections around inclusive language. For example, the expression womxn, which through the substitution of the "e" for the "x", "creates a space for ... [all] women-identifying people", especially "when certain spaces align with white feminists and/or trans exclusionary radical feminist ... ideology" (Womxn's Center for Success, 2021, para. 14).

In places such as Mexico and Argentina, women also change the gender of day-to-day words such as "cuerpo", which is the masculine noun that translates to 'body'. Here, activist women refer to it as "cuerpa" to make the statement that their body is hers and no one else's (see Franulic, 2013; Vicente, F, 2018).

[3] We did not include this syntax in Seis8s, but it remains a future project.

More on playing with lyrics

To further the conversation and continue ideating about how to resist heteronormativity in cumbia and Latin Dance music, I suggested discussing the article "A queer look at urban popular music: Analysis from different musical proposals with a common queer sensitivity" by Spanish scholar Teresa López Castilla (2014).

In this text, Lopéz, who focuses on the field of Queer musicology, discusses the Canadian electro-punk artist Peaches and the Argentinian tropical punk band Kumbia Queers. López pays particular attention to how both artists "use performance as a political act, subverting and de-naturalizing gender through their songs, attitude, clothes, while touching upon other subjectivities" (p. 3, translation mine).

In terms of the lyrics, López highlights that in addition to being "intelligent, lascivious, and bold" they also convey an explicit Queer message (p. 3, translation mine). For example, in Peaches' album "Fatherfucker", the title of the album "is a form of making justice by changing the gender to the insult 'motherfucker'" (p. 3, translation mine). López also highlights how Peaches modulates her voice through technological means to make it sound ambiguous and to be able to perform or play different roles or genders within her music.

In the same vein, in Kumbia Queers' productions, they play with the Argentinian cumbia style *cumbia villera* that emerged in the late 1990s. This music, López explains, is a "re-action to romantic cumbia", where instead of being about love, "it talks about the life in the neighborhoods such as violence, issues with the police, alcohol, drugs, and sex" (p. 5, translation mine).

Here, *villera* is an appropriation or resignification of a pejorative term that originally made reference to a person living in a shanty town and who is considered to "like and deserve poverty" (Martín, 2012). In this regard, Castilla (2014) points to some similarities between the name villera and Queer in that both terms have been resignified to "return loaded with identity and pride of belonging" (p. 6, translation mine).

Is cumbia Queer in Canada?

Even though the conversation of dance did not come up much during our conversations, participant S01 did bring it up at the closing moment of one meeting in Hamilton,

> Is cumbia Queer in Canada?

This participant continued by mentioning it potentially is, as it responds to a type of sexuality different from Country or Rock. They also asked,

> How does cumbia affect bodies when they are dancing? For example, does their [lilting] movement blur their gender?

So far, I have not found much information with respect to cumbia scenes in Canada. However, S01's questions reminded me of an article discussing the relationship between the nation-state and the difficulty for bodies to dance in those spaces. Chilean sociologist Eileen Karmy Bolton (2013) describes, for example, how dance in Chile is "related to long-lasting processes, such as the formation of the Republic" (96)

More specifically, she explains, the development of "coercive processes of historical discipline (such as the ordering of the Nation-State) [resulted] in a kind of body atrophy when dancing" (96). Among others, these coercive processes included the "omission of indigenous and Afro-American heritage" in music (96). As well as the "proscription of carnival ... and popular festivities during the beginning of the Republic" (96).

In the mid-20th century, cumbia arrived in Chile. Due to the nation-state processes above, it developed an accent where the syncopated elements were minimized, making it easier to dance. Here, "the dance of the Chilean cumbia [developed as an expression of] a humorous, spontaneous and not very attractive modality of use of the body" (99). And where its "simplicity ... [gave] the possibility of dancing it freely and unattractively by bodies not used to moving their hips" (105).

In this regard, Chilean cumbia is perhaps also a place where gender blurs as it is also a type of public dance experienced "collectively through choreographies" oriented to entertaining rather than flirting or seducing. For example, through a *conga line* known as 'trenecito' where not only couples but anyone can join to "drag their feet to the rhythm" (102).

This discussion on dance was not comprehensive, however, it is perhaps a starting point for further exploration in regard to dance and choreography. And the ways it develops meaning beyond the traditionally male/leader-female/follower in Latin America and beyond such as in Canada, where this music has already arrived.

3.5 Conclusion

This chapter reported on a series of online talking circles, held from August to November 2020 where people from Latin America discussed software and music in connection to sociocultural aspects specific to them and their locality. The purpose of this chapter was to expand the understanding of some of the sociocultural circumstances intersecting with the creation and use of Seis8s.

Through these circles participants' opened up about their lived experiences to find meeting points and perspectives based on the collective. This way, participants were vital for the conversation to be as relevant and exciting as it was, allowing for the development of rich commentary beyond the individual, academic task of thesis writing.

The number of people involved in these circles – i.e., twenty-six participants – also provides insight into the need for spaces for collective discussion and creation around software and the arts, where artists and developers engage in conversations around the sociocultural aspects surrounding both their digital innovations and their target users. All this, with the aim of making computer technologies and digital art better directed and meaningful to people.

In this regard, participants' questions and contributions in part 1 of this chapter pro-

vided insight into some belief systems and ways of knowing and being of Latin American communities. For example, *mestizaje*, a belief system often tied to the nation-state's intention of unifying through homogenization. And, the *white-mestizo* cultural model that sees multiculturalism and pluriculturalism just in terms of tolerance and wealth.

These emerging themes provide value as they allow the formulation of interesting questions for future research. For example, in which ways has computer software in the region been advertently or inadvertently influenced by national ideologies such as mestizaje? Have their philosophies, metaphors, rules, and functions been influenced by the multiculturalism and pluriculturalism particular to this region? And, what are the implications of this?

Participants' contributions also provided insight into the generalized difficulties of Latin American communities. One difficulty highlighted was the shared complication of breaking the false sense of purity carried by everyday speech in Latin American societies. Another difficulty discussed was that the use of digital technologies is seen as contradictory because they are understood as foreign within these societies.

Upon reflection, participants offered ways of resisting the assumptions behind this rhetoric. For example, they discussed the concept of Cheje, which invites Latin American societies to be formed by the "parallel coexistence of multiple cultural differences that do not extinguish but antagonize and complement each other" (TV UNAM, 2018). Moreover, participants suggested that the Cheje concept can help think of digital technologies as artifacts part of such coexistence.

At the end of part 1, participants discussed gender and sexual orientation as key dimensions of culture, making us reflect on what our body is doing to the spaces from/where we design software. For example, they suggested programming languages as *spaces*, moreover, spaces with an overrepresentation of male bodies and a male-centered design perspective.

Here, participants offered a series of strategies to resist this male orientation. For example, they suggested documenting and sharing the histories of women programmers with the community at large to visualize these bodies while providing fresh perspectives on the matter.

Other suggestions included engaging with spaces where issues of gender can be discussed and understood, with the expectation to educate the general public.

In part 2, the topics of gender and space were also discussed, now from a Latin dance music perspective. Here, participants identified genres such as Cumbia as male-oriented and heteronormative. Because of this, participants and I looked at alternatives to resist through performance. For example, we discussed artists like Kumbia Queers who subvert gender roles through Cumbia villera, an Argentinian style that emerged in the 1990s as a reaction to romantic Cumbia, thus emphasizing other important dimensions of life.

The reflection above led us to discuss the importance of identifying if Seis8s' music responds to a range of bodies. We did not have an answer, but participants and I developed strategies to achieve this. We connected feminist reflections around inclusive language with the naming of commands. And where, for example, the command `tumbao`, which is a masculine noun in Spanish, could have the alternative option `tumbaa`, which might correspond to a feminized version of that noun.

Our discussions also took us to discuss ways of knowing and being now from the perspective of *latinidad*. Here, we discover the ubiquitousness of terms such as *Latino* and *Latina*. And we reflected on the different understandings of these terms and became aware of their complexity as they convey different meanings for different people and vary highly depending on the region or country where one is located.

Furthermore, we identified these terms as confusing and potential enablers of covert asymmetric appropriations of musics and cultures in this region. For example, during the 20th century in Mexico, cumbia went through a process of whitening by the record industry and the music elites.

Finally, participants questioned how *sophistication*, understood within a European ideal, was connected to live coding and Seis8s. And how this *sophistication* influenced our understanding of cumbia. Here, participants touched upon the long history of *sophistication* of cumbia and Latin Dance music in general, where the transformation of the *traditional* hap-

pened, among others, through the integration of instruments considered more *modern*.

Seis8s: applied conclusions

Since the beginning of my doctoral studies, I was very interested in discussing *mestizaje* in relation to software. This is as I had the hypothesis that my identity as *mestizo* had informed – consciously and unconsciously – the ideation and development of Seis8s. The statement that one's identity orients digital technologies might read as obvious. However, my experience is that research on that topic often focuses on the perspective and critique of *whiteness*.

Because of this, it was very interesting to see how often this concept came up throughout the three months of the circles. Furthermore, I was struck when participants proposed discussing it since the first day, particularly as I had decided beforehand not to take the conversation that way, to avoid directing participants to my personal interests. And because I was not expecting them to identify as *mestizos*.

This way, the discussion of *mestizaje* opened my understanding of this concept through other concepts such as *Cheje*, where I was able to detach from a false sense of *purity* of Latin American societies and digital technologies as *foreign*. Detaching myself from these assumptions was essential to overcome my internal conflicts and fears about combining computation with music forms like cumbia.

Another aspect of the circles that I enjoyed very much was when we discussed forms of resistance through performance. For example, I enjoyed the discussion about 'cumbia villera' as a form of displacing normative forms of romance in songwriting confirms the existence of such diverse sub-genres. Through this discussion, I learned that Latin dance music, overall, is irreducible to rigid categorizations, either from the music industry or from the preconception of *latinidad*.

The discussion above on displacing normativity in Latin music reminded me of genres like 'Salsa consciente' (conscious Salsa), which "evokes the idea of class consciousness in the Marxist sense ... and/or an ethical conscience that rejects consumerist individualism in

favor of social solidarity" (Espinoza Agurto, 2014, ix). It also reminded me of the genre of 'Salsa dura' (hard Salsa), which draws upon the jazz genre hard bop by focusing on the improvisatory aspects of the music as opposed to the lyrics.

These genres then provided me with further material to reflect upon and potentially direct me toward the music Seis8s is emphasizing. This made me think, what if Seis8s is enabling a type of music emphasizing resistance to asymmetric types of computer-music abstraction by putting Afro-Latin urban instruments and rhythms at the center rather than as a 'flavor'?

To continue this incipient conversation, in the next and final chapter, I describe a series of live coding performances and exhibitions using Seis8s, where I tried to engage the Latin American community in reflections on the options and restrictions of computer music languages. And to promote critiques and appropriations challenging universal understandings of bodies, culture, politics, and economies of these technological and artistic milieus.

Chapter 4

Seis8s in practice: public performances and exhibitions

Since the summer of 2020, Seis8s has been used in nine collective performances and five individual performances and has been submitted to two conferences as online installations. This chapter reports on these performances and presentations, discussing them individually. Each discussion starts with a contextualization of the place and event where they happened. This is followed by short vignettes narrating the performances.

Additionally, each section ends with a brief discussion identifying and reflecting upon foreseen and unforeseen project objectives. On the one hand, a foreseen objective was to engage the Latin American community worldwide with live coding performance and for them to reflect upon the options and restrictions of computer music languages. This critical objective intended to promote critiques and appropriations challenging universal understandings of bodies, culture, politics, and economies of these technological and artistic milieus. On the other hand, unforeseen objectives included users' engagement in learning and producing electronic Latin dance music.

Furthermore, reflecting upon these objectives revealed important steps that I had to consciously and unconsciously follow to achieve them. Examples of this include achieving stable functionality and usability of the software as well as providing documentation for the

users. Current and further directions to achieving these objectives better are also discussed.

4.1 Seis8s usage aesthetics: Live Coding and Networked Music performance

As mentioned in chapter 3, Seis8s heavily draws from Live Coding practice (Collins et al., 2003; Nilson, 2007). This influence is notable in the aesthetics of Seis8s; for example, the economic aspect of its expressions allows spending less time typing. It is also noticeable in the emphasis on the performer showing their code by sharing the screen with the audience. It is important to be aware of these two characteristics to understand better why the performances described in the following sections look the way they do.

Another aspect influencing the aesthetic experience, particularly within collaborative performance, is the use of Seis8s through the Estuary platform. Estuary is a web-based software that enables musical collaboration with other users through the Internet (Ogborn et al., 2017). From a user interface perspective, by using Estuary, the users – i.e., performers and audience – will experience other users writing and editing code. They might also experience one or multiple syntaxes coming from Seis8s and from other languages available in Estuary. This way, the users will also experience both sound and visual outputs in the form of music, image, and/or video simultaneously.

The performances described in the following section happened online due to the global pandemic of COVID-19. In the Spring of 2020, COVID-19 arrived in the city of Hamilton, Canada, where I was developing this software. Worldwide, communities of artists, including the ones belonging to live coding and networked music, shifted from an emphasis on in-person concerts to virtual/online ones. I had planned in-person musical presentations using Seis8s in Canada and Colombia; however, the pandemic made it impossible to do them. Future work includes using Seis8s in in-person environments.

4.2 Performances and Installations with Seis8s

4.2.1 *D'Binis at Campamento Extendido <impendingvoid> by Posternura Records*

Seis8s' first live performance happened on Friday, July 3rd, 2020, at 21.00-00.00 hrs UTC-4[1]. *Campamento de Verano Cyberpunk* is a recurrent event where artists meet to showcase artwork related to technology and culture. It is organized by *Posternura Records*, a Chilean independent record label. Live coding participants at this event played a variety of music styles, including algorithmic soundscapes by Mexican Hernani Villaseñor and Techno music by Chilean Christian Oyarzún.

Seis8s' performance took the form of an online audiovisual performance presented by Grupo D'Binis, a duo composed of the visual artist Jessica Rodríguez and myself. The performance happened on Estuary, where I performed music with Seis8s, and Jessica created visuals with CineCer0. The link to the performance is https://youtu.be/8UA8QdVat34?t=1355.

Figure 4.1: Poster for the online event "Tokata Campamento: impending void" by Posternura Records and Campamento de Verano Cyberpunk (Chile, July 3, 2020)

At this performance, I played a cumbia-like improvisation as this was the music style I had been practicing up to this point with Seis8s.

The performance started with the woody sound of the jam block while a disem-

[1]UTC stands for Coordinated Universal Time, and it "is the primary time standard by which the world regulates clocks and time" ("Coordinated Universal Time", 2022, para. 1). UTC-4 is one of many time zones which "are expressed using positive or negative offsets from UTC" (para. 7). UTC zones can easily be translated to local time using the Internet, for example by googling them. During the COVID-19 emergency, online live coding events online started coordinating using the UTC zones to avoid time misunderstandings.

bodied voice – coming from the command `altavoz` *– greeted: "¡Arriba Tokata Campamento! [Come on Tokata Campamento!]". I introduced a hi-hat sound while continuing typing to produce a conga sound (24:13), followed by a bass guitar playing a basic cumbia pattern with a C major chord and after a couple of minutes also D minor chord (24:44).*

The disembodied voice continues now saying the duo's name: "Grupo D'Binis" (26:37). A couple of minutes after, the güira comes along with a new chord sequence of D min and C# min that creates a heavier texture (28:44). Suddenly, a crash! Something made Seis8s and Estuary crash altogether – up to this point in its development, Seis8s had many hidden bugs, mostly related to how it combined the lists of notes and rhythmic values. From my experience, software crashes are common in live coding practice, and sometimes that even adds some rush to the performance. For me, the crash made the performance memorable (34:36).

After a necessary reload and a couple of minutes lost, the performance continues almost from the same place – as I pasted back the code that I had copied before reloading. Now a glissando sound comes in and out at no specific time. I type a new chord and slowly increase the tempo to 190 beats per minute.

A plucked-string-like sound is now playing to produce a simple melody with the unison and the octaves from each chord (39:47). The accompanying keyboard becomes more noticeable when I try a new sample. A long-gone hi-hat sound comes back, bringing an air of refreshment into the improvisation. I try to vary the music more by changing the chords. Still, my inexperience with Seis8s makes me program something different, making the chords disappear and leaving the percussion alone for a couple of moments (46:09).

It's time for the conga solo! I start with a list of rhythmic values arranged randomly by hand (46:48). The disembodied voice comes back to announce "Grupo D'Binis" and to greet "Arriba tokata campamento!". It is time to go (49:51). A

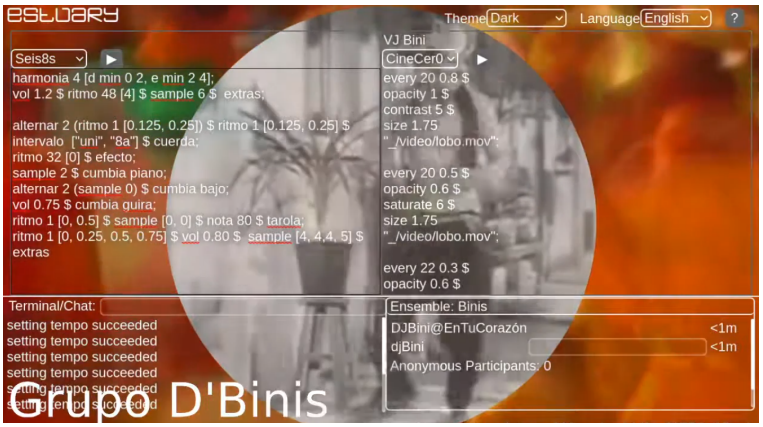


Figure 4.2: Screenshot of D'Binis performance using Estuary at Tokata Campamento (Chile/Canada, July 3, 2020)

> *simple melody is playing while I simplify the code and the music in preparation to end our 20-minute performance. Only the bass and the conga are left before slowly fading away.*

An aspect I was uncertain about was if I was going to be able to keep the music dynamic and changing for the 20 minutes that we had programmed. The latter was because of the little practice I had up to this point with Seis8s. After watching this performance again, I felt that enough musical variation had happened, and it was fun to hear it again. I noticed that I achieved a lot of dynamism by adding new chords, deleting older ones, and speeding up Estuary's tempo.

At this point in the development of Seis8s, a key goal was to put out the idea of this language by engaging the audience in two ways: with its Spanish-derived commands and its cumbia-like style of electronic music that could appeal to them differently than other more pervasive styles such as techno. At this point, showing the audience that Seis8s was usable was essential for me too. Showcasing Seis8s usability encompassed demonstrating that Seis8s had many commands and options for their modification. It also encompassed conveying a certain amount of reliability by avoiding too many crashes during my performances.

4.2.2 *RGGTRN at Digitropics*

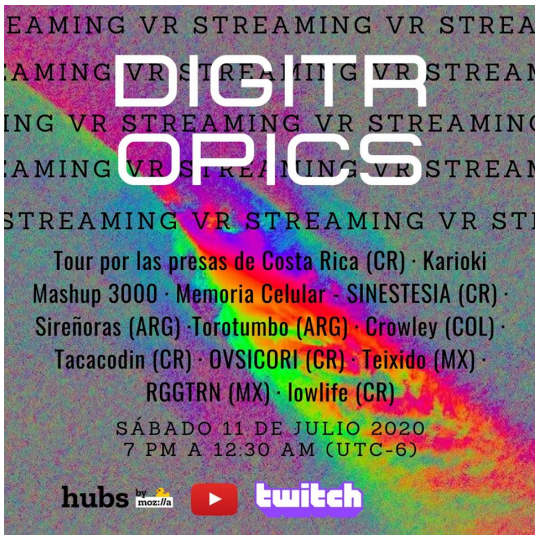


Figure 4.3: Poster for the online event "DigiTropics" by Tacacoding (Costa Rica, July 11, 2020)

Seis8s' second live performance happened at the event *DigiTropics by Tacacoding* on July 11th, 2020 at 19.00.00 hrs UTC-6. Tacacoding is a live coding collective located in Costa Rica that frequently organizes live coding concerts. This particular edition had a lineup of live coding artists performing Latin dance music such as Argentinian-based *Torotumbo* and *Sireñoras*.

Seis8s' presentation at this event took the form of an online audiovisual performance presented by RGGTRN, a quartet composed by the media artists Emilio Ocelotl and Marianne Teixido as well as of Jessica Rodríguez, and myself. The link to the performance is https://youtu.be/McnXHFJYBQM.

Jessica and I started off the performance using Estuary. Emilio and Marianne were in charge of the second act using SuperCollider. From our side, Jessica and I planned to improvise around videos of Walter Mercado, a *hispanohablante* TV celebrity known for reading horoscopes.

> *Using Estuary's CineCer0, Jessica starts by playing videos of Walter Mercado. I focus again on creating cumbia-like music. I start with a disembodied voice, now coming from a text-to-speech website I had open.*
>
> *The voice greets "Tacacoding a través de Digitropics y RGGTRN Internacional traen para ustedes Walter Mercado, el homenaje [Tacacoding through Digitropics and RGGTRN International bring to you Walter Mercado, the tribute]". After, Jessica played a video from a TV announcement where Walter Mercado promoted his horoscope services.*
>
> *A conga sound starts, evolving from no apparent rhythmic pattern into a basic cumbia* marcha. *A jam block and a kick drum come in. There is no specific*

command to play the kick at this time, so I use the command `extras`*, which contains a bunch of additional samples (1:37).*

Figure 4.4: RGGTRN internacional at "DigiTropics" (Costa Rica/México/Canada, July 11, 2020)

An alarm-type sound plays for a couple of seconds while a cumbia bass fades in slowly. Better control over Seis8s' functions is noticeable this time! I start adding new chords to vary the hypnotic aspect of cumbia while Jessica continues throwing new videos. A new video now looping coincided with the introduction of a melody using a very synthetic sound coming from the command `teclado` *(5:46).*

The melody commences varying while the video is still looping. Suddenly, the chords stop. There is a big silence. This time this is not a mistake but a way to produce silence on harmonic instruments like the keyboard. This silence also gives space for a new video where Walter Mercado is speaking. The chords reactivate their previous pattern, and there is no silence anymore (8:13).

A plucked-string-like comes in while I slowly accelerate the tempo on Estuary's terminal. The music is faster now (9:55). The cumbia-like texture now resembles hypnotic circus music! There are three Walter Mercado videos playing at the same time. Is this part the climax of the performance? I think so, as more videos are added (10:47).

I lower the overall volume of the music, while the disembodied voice comes back to greet the organizers again, namely Tacacoding*. Walter Mercado's voice gets in*

the mix when he says "El amor es la razón de vivir" and "I am Walter Mercado". 15 minutes have passed, and it is time to switch to Marianne and Emilio (11:23). I prepare the streaming so they come in smoothly while I lower the volume of my music. The music disappears almost completely (14:22). Only some congas and a güira are left while Emilio and Marianne enter with a reggaeton-industrial-like rhythm ...

I believe that RGGTRN's performance, particularly the section performed with Seis8s, fits well within the expectations and ideas behind this event. This event has been one of the few ones in which there is an emphasis on electronic Latin dance music. For example, the name of the event, that is *Digitropics*, seemed to convey the idea of digital "tropical music". Additionally, live coding cumbia bands such as *Sireñoras* and *Torotumbo* were present performing at this event.

4.2.3 *Luis Navarro Del Angel at* Galvanized Suns *by Diasporic Futurisms*

Seis8s' fourth presentation happened on October 15th, 2020, during a three-day online event called Galvanized Suns, organized by the curatorial team *Diasporic Futurisms*, composed by Toronto-based artists Adrienne Matheuszik and Vanessa Godden. This event featured artworks re-imagining "pasts, present and futures through narratives of diaspora outside of Western hegemony by artists, designers, performers, and writers who contribute to the Caribbean and Latin American diaspora culture in Canada" ("Galvanized Suns Archive", 2020). Artists participating included Montreal-based live coder Illest Preacha. This event was organized in partnership with the Toronto community-building platforms *Subtle Technologies* and was supported by the Canada Council for the Arts: Digital Originals Grant. This solo performance happened in Estuary, and the link to the recording is the following: https://youtu.be/b3qQDQ2N2Ko.

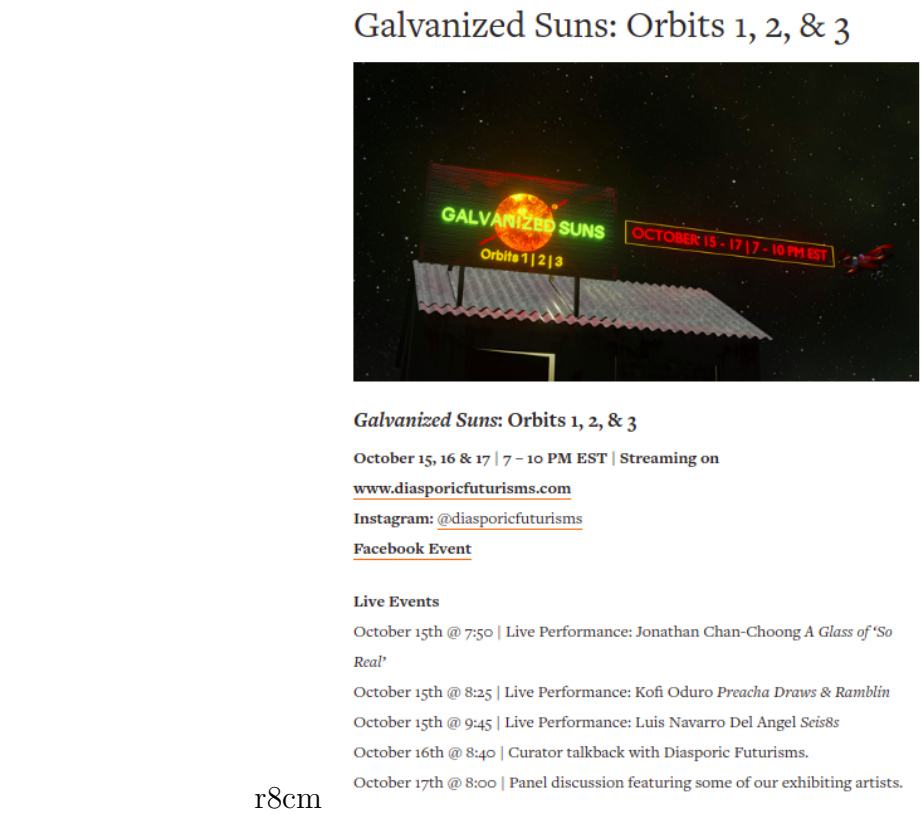


Figure 4.5: Flyer of the online event "Galvanized Suns by Diasporic Futurisms (Hamilton, Canada, Oct 15th, 2020)

I start the performance with a preset cumbia rhythm for the bass moving through the D min and G minor chords. After that, I introduce a jam block cumbia and a conga preset as well as a keyboard accompaniment (0:40). To create variation, I use the function `alternar` *to switch between accompaniments in beats 2 and 4 to beats 1, 2, and 4 to create variation. I start playing with the* tumbadora *drum part of the conga set. The* tumbadora *provided a bassy sound with every eight beats, creating some more variation to the music (1:32).*

I play a melody using two commands because the `punteo` *command was not yet conceived. With the command* `intervalo`*, I accessed the 5th and 3rd interval of each chord, and with the* `ritmo` *command, I programmed the 16th beat pattern (5:38). I duplicate the cumbia sound one octave above to reinforce the melody (6:46). After a couple of moments, I erase the melody and start lowering the intensity of the sounds, opening space for a conga solo (8:12).*

Figure 4.6: A Seis8s' syntax error during a conga solo at Galvanized Suns

The conga solo starts (9:43). The main drum for this solo was the quinto*, which is high-pitched. I start playing with 16th notes. I then add another pattern that now includes two* tumbadora *open bassy sounds (10:00). I continue playing with 16th-beat rhythms executing after each small change to add to the idea of the liveness of the solo (10:48). However, a syntax error that I had not noticed earlier deters the conga from responding to the changes and, thus from arriving at a climax. There is nothing to do about it at this point as the fifteen minutes the organizers gave me to play are coming to an end (13:23).*

I decide to finish the conga solo and move on to prepare to end the performance. I introduce all the instruments again to bring back the festive feeling (15:12). The melody also came back, as well as a disembodied voice saying "¡sensacional!" (i.e. sensational!) (16:09). I start to lower the volume of all the sounds, and one by one, they disappear slowly while the disembodied voice continues saying "¡sensacional!". I erase one instrument after the other until only the bass guitar is left to fade out slowly.

This performance was one of the few that did not include visuals because I was rethinking the relationship between the visuals and Seis8s. This state of mind partly developed after perceiving that this event was more serious than the others described. This event had a

curatorial aspect, which felt like I was entering a gallery space, and thus, I felt my performance with Seis8s had to comply with this space.

In previous performances, the visuals were intended to add a campy element. An example of these visuals included video excerpts from the horoscope celebrity Walter Mercado described above. However, at Galvanized Suns, I thought this type of exaggerated visuals would not align with the gallery-wise quality I had presumed of this event. Ultimately, Galvanized Suns resulted in the opposite of what I thought, with people dancing and enjoying the music. Furthermore, the campy visuals would have encouraged people to dance and enjoy even more.

4.2.4 *Tania Alejandra, Gabriel G. aka alom, Jessica Rodríguez, Andres Miramontes, and Luis N. Del Angel: Rehearsal for a Performance at the Factory Media Centre*

Figure 4.7: Poster for the online event "Talking and Music Circles on Decoloniality and Software (Hamilton, Canada, Oct 24, 2020)

This was a rehearsal in preparation for an online performance and artist talk at the Factory Media Centre (FMC) located in Hamilton, Canada. This rehearsal was on Saturday, October 24, 2020, around noon EST. It happened at one meetup for the *Talking and Music-Making Circles on Decoloniality and Software*, described in the previous chapter. The performers were Tania Alejandra, Gabriel G. aka alom, Jessica Rodríguez, Andres Miramontes, and myself [2]. The link to the performance is https://youtu.be/jrCh43IFcac.

This rehearsal starts with güira and a plucked-string-like sound that I have used

[2]These participants agreed to be identified for this performance

in previous performances. Actually, at this point, Seis8s had only two plucked sounds. This was a long one that was moving across the different chords. The plucked string lowers one octave while one of the participants slowly introduces a conga, trying out different "hand hits" (0:41).

Tania is in the process of coding an accompaniment keyboard, while the bass guitar, played by Jessica, has been sounding for a while. This music does not resemble the cumbia music previously done with Seis8s. The cadence feels more like Cuban music (3:48). This cadence is accentuated by Tania's accompanying keyboard now playing quarters on each of the four beats (4:01).

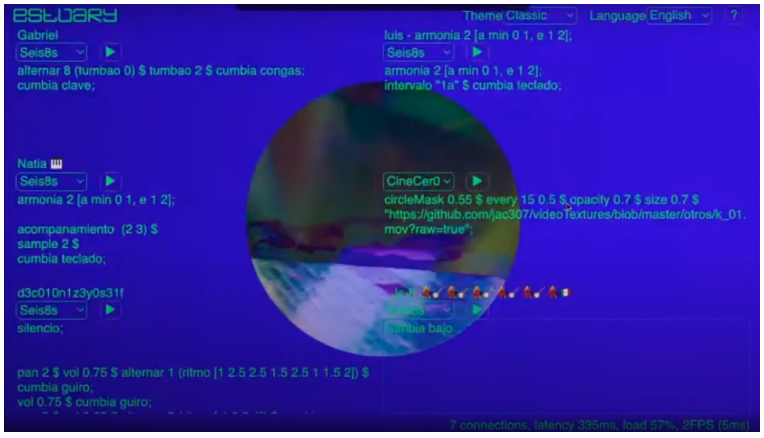


Figure 4.8: Performance at the event "Talking and Music Circles on Decoloniality and Software (Hamilton, Canada, Nov 5, 2020)

The plucked string continues with a predictable yet comforting melody that is now followed by a high-pitched, more synthetic sound. A 3/2 clave comes in, played by Gabriel (5:02). I had forgotten there was a clave in Seis8s. The congas take more presence; they vary each cycle. An accordion came in a moment ago, and it is playing only one note with each chord change (6:50).

The performance seems to stall for a moment, but then people take out some of the instruments, making it simpler to listen (9:30). The long-gone accompaniment of the keyboards comes back with a reverberation that makes it sound more organic. I am not sure where the reverberation comes from, though, as Seis8s does not

have these effects yet (10:09).

I suggest preparing ourselves to finish the rehearsal. The music still sounds like Cuban music, with a slow cadence similar to Cuban Son (12:40). People start fading out their instruments until only a constant quarter-note keyboard accompaniment fades out moments later.

I think this was a more memorable performance than the one we presented one week after at the FMC. This is because the music we played in the rehearsal sounded within a completely different style than my previous collaborative performances. It was also memorable because I felt everyone was very relaxed and having fun. They also were able to follow my instructions easily while keeping their flow and spontaneity.

The performance we gave at FMC was also very good; however, I felt that the participants and I were nervous. We also ran into some technical issues where my computer, from where I was streaming, halted a bit, hindering the performance flow. Some issues with visuals happened as well, as they were not rendering properly.

4.2.5 *Cybernetic Orchestra at Estuary's 5th Anniversary by the NIL*

This performance happened as part of a seven-hour online concert celebrating *Estuary's 5th birthday*. Members participating with the Cybernetic Orchestra were Liam Foley, Alejandro Franco, Alex MacLean, Sam McBride, Samantha McEwan, David Ogborn, Jessica Rodriguez, Gordon Simmons,

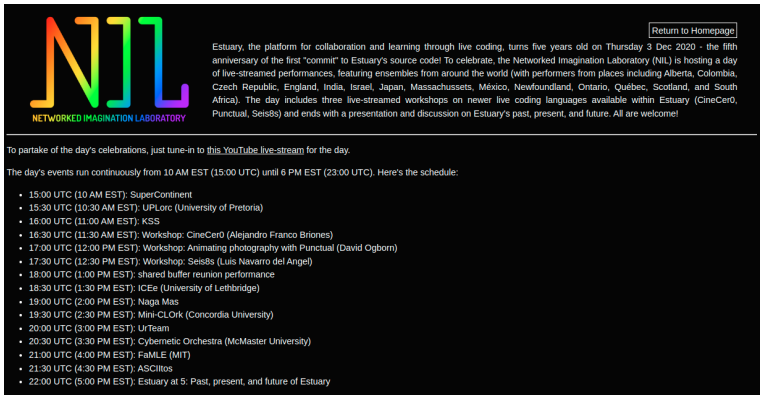


Figure 4.9: Lineup of the *Estuary 5th Anniversary* (Hamilton, Canada, Dec 3, 2020)

Stephen Surlin, Alejandro Tamayo, Andrew Veinot, and myself. This performance was on Thursday, December 3rd, 2020, at 3:30 PM EST. The concert was streamed on Youtube and can be accessed through this link: https://youtu.be/fd83R6gTgxY?t=18618.

In this performance, the visuals were brought back with the use of *CineCer0*. Another language called *MiniTidal* was used too. This is a language for making sound and music patterns. An interesting aspect was that for this performance, the orchestra used the Estuary roulette widget that simulates the action of lining up to form a virtual *conga line* (see Ogborn et al., 2017). This way, the twelve of us were able to modify code by lining up and taking turns on the three different text editors.

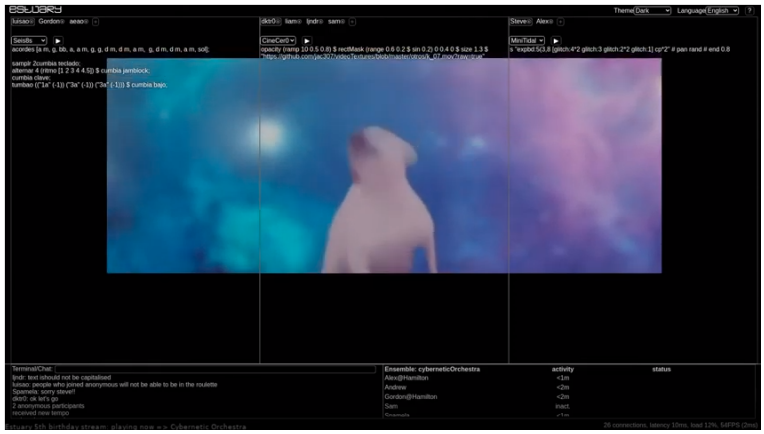


Figure 4.10: Three text editors in Estuary, each with *virtual* conga lines (Hamilton, Canada, Dec 3, 2020)

> *The orchestra starts with a cumbia bass pattern moving throughout a harmonic progression composed by A minor, G, A minor, D minor, A minor, and G. At the same time, Alejandro codes an image of a dog dancing in space. While this playful image is displaying, a jam block sound and a clave join in (5:11:07). On the Minitidal editor, a sound called* glitch *starts playing. In that same editor, more orchestra members start lining up! The same happens in Seis8s' editor, where Gordon makes a quick change (5:11:51).*
>
> *David is now lined up to change some parameters of Seis8s, while a text with the generic line "lorem ipsum" appears on top of the dancing dog (5:15:02). Some more percussive sounds get added through Minitidal; they move from the left to the right speaker. The perceived randomness of Minitidal rhythmic patterns makes the music dynamic and more entertaining. A melody is coming from Seis8s, which has*

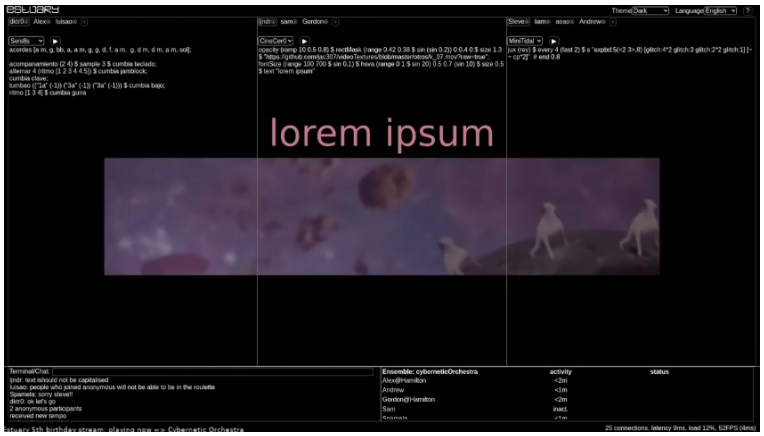


Figure 4.11: A dancing dog in space and a *lorem ipsum* text (Hamilton, Canada, Dec 3, 2020)

been sounding for some time now. It is not the product of the punteo *command but from the* acompañamiento *command. Now, I do add another melody using the* punteo *command to create a melodic counterpoint (5:16:11).*

The counterpointing melodies progress through the same chords established at the start but are now accompanied by a text saying "cu cu cu cumbia!!!". It is interesting that this text, as well as the "Lorum Ipsum" text, substitutes the disembodied voice heard in other performances. An audience member who previously commented on the chat now comments on the melody: "totally bobbing my head over here" (5:21:03). Orchestra member Jessica replies to the participant about how one time we played Cuban son with Seis8s.

David suggests starting to deconstruct to produce a graceful exit. David, Liam, and Alejandro proceed to do the same for Minitidal while Alex focuses on Cinecer0 visuals. Steve and I focus on Seis8s by simplifying the code and just erasing it at times (5:23:25). We are almost finished when the text "Happy Birthday Estuary displays on the screen". As in other previous Seis8s' performances, the bass is the last to fade out. The orchestra achieves a smooth piece with minimal syntax errors.

An exciting aspect to reflect upon is that the orchestra members preserved the idea of

producing cumbia-style music in contrast to the performance in the section above, where the ensemble participants played Cuban-style music.

4.2.6 *The ASCIItos: Barush, Nuriban and Gabriel G at Estuary's 5th Anniversary by the NIL*

This performance also happened at the concert for Estuary's 5th anniversary. This performance is particularly important because it was the Seis8s' first performance in which I did not participate. Rather, participants of this performance were the Mexican musicians Barush Fernández, Nuriban, and Gabriel G., aka alom. They used Minitidal and another language hosted in Estuary, Punctual, for synthesizing sound and visuals for this performance. The link to the performance is https://youtu.be/fd83R6gTgxY?t=22312.

> *Their performance starts with a drone sound coming from an accordion coded in Seis8s. They are using the parameters to lower octaves using the command* `punteo` *(6:12:35). Almost right away, a drum kick starts while more notes are added to the droney accordion (6:13:21). At the top-left corner text editor, another person struggles to resolve a syntax error that deters Punctual from rendering the visuals. The music continues moving dynamically, however.*
>
> *The rhythm of Minitidal's kick transforms into something calm and smooth before starting to be accompanied by some conga sounds coming from Seis8s (6:15:51). Finally, Punctual's syntax error gets resolved, making reactive, colorful vertical lines appear (6:16:35). More hits are being added to the conga to produce a faster rhythmic pattern. The sound of a square wave coming from Punctual is added (6:17:05). It moves from low to high pitches while the multiple-color vertical lines continue reacting accordingly.*
>
> *Suddenly, the kick drum and the congas align, producing a more techno rhythm, while the square wave seems to be getting to a climax by moving rapidly through*

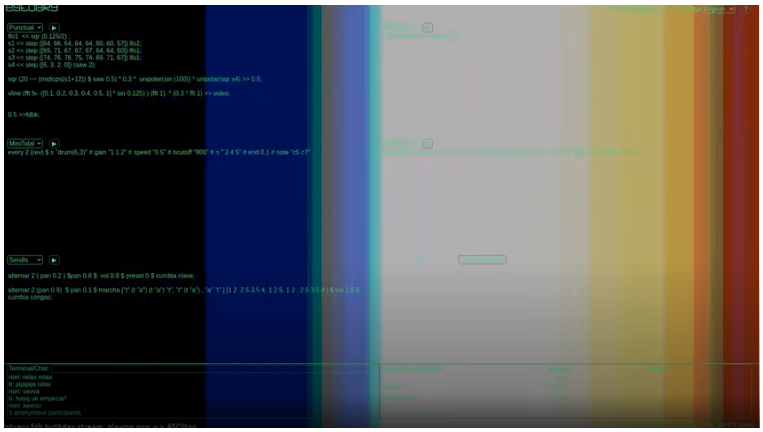


Figure 4.12: Asciitos performing at Estuary's 5th birthday (Hamilton, Canada, Dec 3, 2020)

mid-range pitches (6:20:15). The climax reaches, and the performers now enter a new section using low-pitched drone sounds and a steady but calm kick drum (6:20:43). A harsh melody seems to be processed by a cutoff and vowel filters in Minitidal while the congas and the kick drum continue steadily.

The performers decide to deconstruct, and someone simplifies Seis8s' code, leaving a clave emphasizing each quarter note. Conga sounds also play at times. The visuals are stacked in the upper-right corner. But they start to fade out when someone types `silence` *on Minitidal (6:22:27). Now, a smoother melody is playing on Punctual, still using a square wave. This melody has two parts, one lower and calm and another high-pitched and faster. This makes one realize the performance is not actually ending but has just transitioned to a hypnotic, relaxed moment.*

Some more movement is created in Punctual's visuals while the congas and the melody continue. Drums are coded in Minitidal. Suddenly, a harmonic change happens in Punctual's melody (6:26:01). It becomes mysterious and appealing. After a couple of moments, the melody volume goes down, almost imperceptible, but it is still there (6:26:31). Seis8s' congas continue steadily but in an interesting way because they move through an intricate five-measure rhythmic pattern (6:27:03).

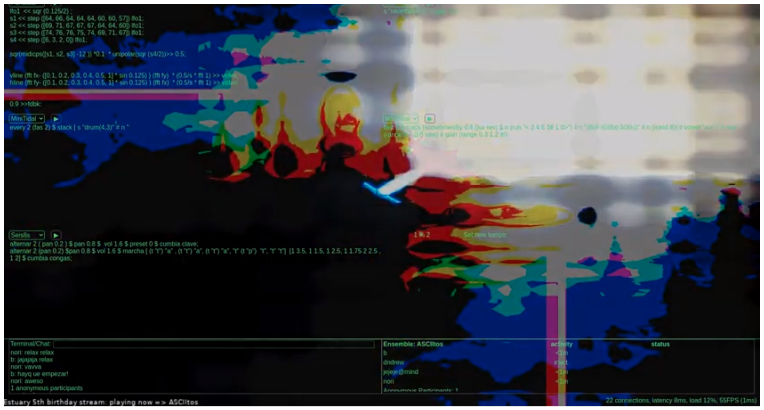


Figure 4.13: Asciitos performing at Estuary's 5th birthday (Hamilton, Canada, Dec 3, 2020)

Figure 4.14: Asciitos performing at Estuary's 5th birthday (Hamilton, Canada, Dec 3, 2020)

Estuary has a chat widget where one of the performers announces that the end of the performance is coming soon. They say "let's make some noise!" (6:33:10). The music becomes techno, moving faster now, perhaps in preparation for a grand finale. The congas and the kick drum are steady and very present. Punctual's melody is dynamically changing too (6:33:28). The visuals are now pure blue, coming and going from right to left. Suddenly, the rhythmic steadiness breaks. It becomes more random and slow. It is seductive. Percussive sounds not heard before are coded while the melody evolves into a high-pitched area that gives the feeling of an upcoming explosion (6:34:18).

Through the chat, they greet their city of origin, Morelia, Mexico (6:38:28). Punctual's code gets erased, and the visuals go with them. Finally, Minitidal's code disappears, cutting off its percussive sounds abruptly.

Asciitos' exploration of the aesthetics of drone music through Seis8s showed an advance in terms of the known goals of this project, namely the technological adoption and re-purposing of Seis8 by its users. Furthermore, it showed the potential for users to break the constraints of Seis8s' ideology. This ideology was being conveyed through presets referencing Latin music and commands, such as the `cumbia` command, required to initialize the instruments. Asciitos' performance, two sections above, showed that asking the user always to use these initializing commands was restrictive. Following a more recent revision to the language, the `cumbia` command is now optional, allowing users to think about instruments and musical patterns outside this genre more freely.

4.2.7 *Grupo D'Binis at PUMPUMYACHKAN Festival 16th by Asimtria*

This was the third online performance by Grupo D'Binis – composed by visual artist Jessica Rodríguez and myself. This performance happened on May 7, 2021, during the PUMPUMYACHKAN Festival 16th edition, organized by the artistic platform Asimtria, located in Ayacucho, Perú. The link to the video is https://youtu.be/LkZvv30bU4g?t=4115.

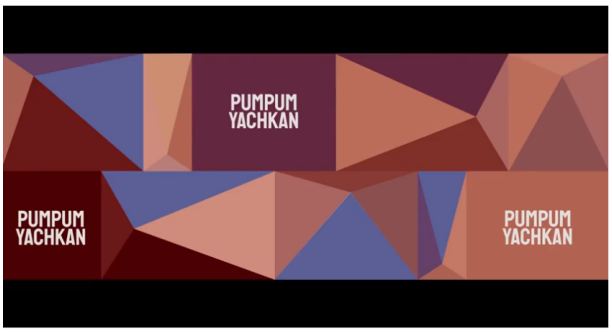


Figure 4.15: Poster of PUMPUMYACHKAN Festival 16th by Asimtria (Ayacucho, Perú, May 7, 2021)

This performance marked an important moment within the usage of Seis8s. First, in this performance Jessica and I played two compositions that had a defined structure, rather than relying purely on improvisation as in our previous performances. Secondly, we enriched the use of Seis8s not only through visuals but also through a physical keyboard/synthesizer that Jessica used to play melodies. In addition to this, I used a MIDI controller connected to the digital audio workstation *Reaper*, to launch pre-recorded greetings, sound effects, and to play some melodies as well.

> *Our performance starts with a disembodied voice triggered from Reaper by with my MIDI controller. It says "Pumpumyachkan Festival Asimtria 16 trae para ustedes a Grupo D'Binis. Con sus anfitrionas de confianza Dj Bini y Vj Bini. Presentando el estreno mundial de la Cumbia Michoacana y la Cumbia de la Lluvia 2.0. Vámonos con ellas! [Pumpumyachkan Festival Asimtria 16 brings Grupo D'Binis to you. With your hostesses Dj Bini and Vj Bini. Presenting the world premiere of Cumbia Michoacana and Cumbia de la Lluvia 2.0. I'll leave you to them".]*
>
> *A lower-voiced effect starts, paired with the congas (1:09:27). A disembodied voice announces again "Grupo D'Binis, con sus anfitrionas de confianza Dj Bini and Vj Bini [Grupo D'Binis, with your hostesses Dj Bini and Vj Bini]". A keyboard starts a cumbia-like accompaniment just before Jessica starts playing the main melody*

of La Cumbia Michoacana (1:10:13). After one pass from Jessica, I double the melody with my midi controller (1:11:06).

Figure 4.16: D'Binis La Cumbia Michoacana at PUMPUMYACHKAN Festival (Perú/Canada, May 7, 2021)

Then both move on to what seems to be a chorus section. We do not repeat the melody right away, but rather only the bass is left sounding only (1:12:25). A güira and the disembodied voice that announces the name of our band come again (1:12:53).

Jessica plays the melody again with the synthesizer; I double the melody so there are two layers (1:13:55). We move again to the chorus, which now repeats and repeats. At the same time, a video of Walter Mercado plays in a loop.

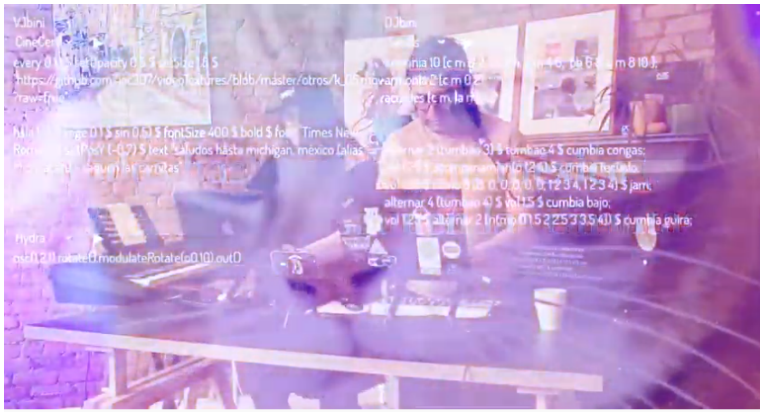


Figure 4.17: D'Binis performing with physical keyboards/synthesizers at PUMPUMYACHKAN Festival (Perú/Canada, May 7, 2021)

I prepare Seis8s' code to transition to the next composition, "La Cumbia de la LLuvia". The tempo increases while Walter Mercado is looping (1:16:40). The

keyboard accompaniment coming from Seis8s becomes more frequent, increasing the intensity of the performance (1:17:10).

Jessica makes a glissando on the synthesizer to give some variation while there is a hypnotic rhythm coming from the bass, the keyboard, and the güira. Jessica repeats. The disembodied voice announces the band again "Grupo D'Binis, con sus anfitriones de confianza Dj Bini y Vj Bini".

I prepare to start playing the melody created for this composition. I select the proper sound in the midi controller, and I start. It is a high-pitched short melody that threads with a secondary, longer melody played by Jessica. Both melodies complement. We repeat (1:21:17).

It is time to end our 15-minute performance. Jessica and I stop playing the physical synthesizers giving space to a melody I launch from Seis8s. The music lowers down while I present the band through a microphone I had set up: "Grupo D'Binis es en los teclados Vj Binis, en los visuales Vj Binis. Su servidor, Dj Bini en el teclado segundo y el código de Seis8s. ¡Gracias! [Grupo D'Binis is on the keyboards Vj Binis, on the visuals Vj Binis. Dj Bini on the second keyboard and code for Seis8s. Yours truly, thank you!]" (1:26:40).

4.2.8 *Seis8s as an online Installation at International Conference on Live Coding*

From December 15 to 17, 2021, I presented Seis8s as an online installation at the International Conference on live coding, streamed from Chile. This was the first time Seis8s was introduced to the public as a standalone website (or web application) rather than as a language hosted in Estuary. Peer reviewers of the conference provided important feedback on technical and conceptual issues with the application that influenced the user experience.

Figure 4.18: Seis8s at the International Conference on Live Coding 2021 (Chile, Dec 15-17, 2022)

One important insight that this feedback provided was how Seis8s was now perceived as both an artwork and a tool. The latter could be linked to the tangibility that websites provide. That is to say, a computer language by itself could be perceived as more ephemeral and malleable, similar to the idea of the *artwork*, which allows multiple interpretations. On the other hand, a website is often more static and rigid and could be compared more to the idea of the *tool*, which has a more defined function.

Seis8s being perceived as both an artwork (or a language) and a tool (or a website) can be positive for the user experience. This is because this separation provides more clarity about the graphical interface's affordances versus the language's affordances. Furthermore, it also allows the user to understand whether an issue is coming from the interface or from the language.

Figure 4.19: Website of the International Conference on Live Coding 2021 (Chile, Dec 15-17, 2022)

In fact, the reviewers pointed out a lack of intuitiveness connected to perceived latency in the music changes, based on a delay between the moment they pressed the play button and the moment the code was taking effect.

On a different note, reviewers provided insights in regards to the lack of discussion about the conceptual aspect of Seis8s. At the time, this was true because no website sections discussed its theoretical background. Furthermore, this lack of conceptual discussion made it difficult for the users to pinpoint what were the aspects challenging the norms within computer-music platforms. For example, alluding to Latin American music becomes simplistic by not providing more context about its

importance in relation to the cultures producing them and/or in relation to computer music in general.

In this regard, the lack of discussion and documentation made it unclear how Seis8s' was producing critique or how it was a product of critique. This also suggests that new media art and technology need to be contextualized when it intends to convey critical intentions. Otherwise, the work stays too open to interpretation and speculation that could be counterproductive.

4.2.9 *Genuary 2022 Seis8s In Space* by Illest Preacha

Genuary 2022 Seis8s In Space is a 55 seconds performance by Montreal-based new media artist Illest Preacha. This performance happened on January 6, 2022, as part of the event *Genuary 2022*. People were invited to create one small artistic piece per day using creative coding software at this event. Illest Preacha performed using Seis8s and the online software Livecodelab for creating visuals. The link to the performance is the following: https://youtu.be/zgbEE75fdRE.

Figure 4.20: Genuary 2022 Seis8s In Space (Canada, January 6, 2022)

> *Illest Preacha's performance starts with a code snippet from Seis8s. The performance starts with energetic music too. There is a prominent bass that jumps from note to note. A keyboard sometimes responds by filling in the gaps, coinciding with the bass. It switches from two different chords, namely Dm and F. In parallel, abstract round shapes lay behind Seis8s' code. They are white, yellow, green, and red. The congas and the güira have been sounding since the beginning. They have a steady pattern, but Illest Preacha transforms them at times to create*

movement. At 41 seconds, the shapes transform into straighter lines just before fading out at 55 seconds.

It is important to mention that Illest Preacha's performance had a post-production stage where he modified his recorded performance by accelerating the speed ratio. This modification post-performance points to a novel and innovative type of appropriation of Seis8s. This was innovative in that by speeding up the recording, a performance that originally might have taken minutes now becomes abbreviated to less than one minute. This shortening of time contrasts with all of the other performances in this chapter, whose length was 15 minutes or more.

A second innovative aspect was that the fast speed of the recording not only made the pitches higher but also made all of the cumbia presets that Illest Preacha was using unrecognizable. This modulation allowed Illest Preacha to move away from Seis8s' prescribed music style.

4.2.10 *Seis8s as an online Installation at the New Interfaces for Musical Expression Conference*

Figure 4.21: NIME conference poster (New Zeland, June 28-July 1, 2022)

From June 28 to July 1st, Seis8s' standalone was exhibited as an online installation at the 22nd New Interfaces for Musical Expression conference (NIME), streamed from New Zealand by the University of Auckland. This NIME edition had the sub-theme of "Decolonising Musical Interfaces".

As with the ICLC conference, important feedback was provided by peer reviewers. A critical insight was that Seis8s was described as a *musical-loop software with a text interface*. This description is interesting as it provides innovative perspectives on

Seis8s' code. For example, naming Seis8s' code an "interface" positions it as the element that enables direct interaction with the music to be produced.

The idea of the "interface" allows one to see the artificiality of code, thus opening the potential for re-imagining it – for example, through the use of natural languages such as Spanish. Furthermore, reviewers' feedback revealed that looking at Seis8s' code as artificial allowed them to appreciate the intent of confronting and potentially re-configuring the default user – e.g., the English speaker who performs jazz or rock music.

Other important feedback was in regards to issues with the graphical user interface that could affect the user experience. For example, reviewers highlighted the lack of visual cues to help the user know whether the code has been properly executed and/or when the code is producing an error. This feedback was very important and guided me to include error messages showing if there is a syntax error in the code.

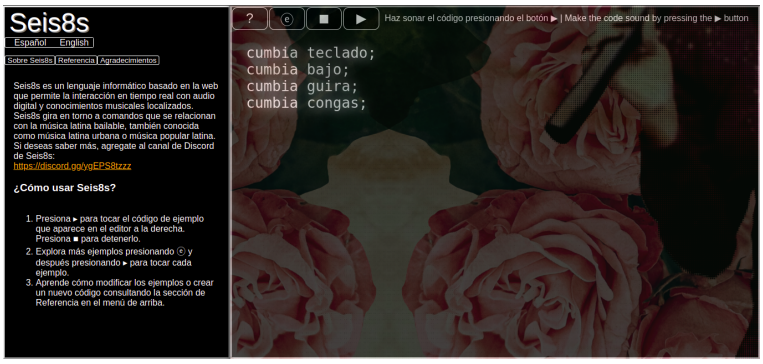


Figure 4.22: Genuary 2022 Seis8s In Space (New Zeland, June 28-July 1, 2022)

A perceived issue with the lack of Seis8s' documentation was highlighted too. Seis8s' website lacks information describing the possibilities and motivations of the software. This was true at the time, as no initial directions were given to the users for them to know how to use the language. Based on this, I now have included an *about* section explaining what Seis8s is. I have also incorporated clear steps for users to try the initial examples provided in the platform.

Finally, this feedback revealed a desire for users to have background information about the Latin rhythms proposed in Seis8s. This made me recenter my reflections on one of Seis8s' intentions: to disseminate and educate about Latin American music. This goes hand-to-hand with reflecting on usability. That is, if users know contextual information about Latin

American music, they can make sense of Seis8s' commands. Making sense of its commands will allow them to use them better, appropriate them better, and create critique with them.

Currently, there is a reference section on Seis8s' website describing the instruments, their commands, and their parameters. Future work includes creating tutorials and how-to guides to contextualize these rhythms.

4.3 Conclusion and further work

This chapter reported known uses of Seis8s, focusing on collaborative and individual performances as well as online installations. For this chapter, my account of these uses included short contextualization about the events where these happened, followed by vignettes of the performances. Short discussions were weaved into the accounts touching upon the goals of engaging a Latin American community and inviting them to produce critique through the appropriation of the software.

An important aspect identified was the steps taken to achieve these goals. For example, the first step was showing the language even if it was in its initial stages, just with the purpose of *putting out there* the idea of a Latin-music Spanish-inflected computer language. This step also involved making Seis8s *legible* through clear commands that consumed few parameters. Another way to make it legible was through a quick start that helped convey Seis8s functionality.

I started this chapter by identifying key aspects influencing the uses of Seis8s, for example, the emphasis on the interactive experience that modifying and displaying the code during the performance allows the users. This was also true for Seis8s in the context of online installations. I also identified networked music as an important influence for these uses, especially when performing collaboratively through Estuary. Finally, a relevant aspect influencing Seis8s' uses was the COVID-19 health emergency that encouraged performances and conferences to happen in the virtual space.

In this chapter, I also talked about various people collaborating in the performances. For example, from the first performance of Seis8s, visual artist Jessica Rodríguez collaborated by coding visuals to accompany Seis8s' music. Other collaborators of these performances included my dissertation supervisor David Ogborn and the members of the Cybernetic Orchestra as well as the participants of the talking circles. These collaborations were crucial for trying Seis8s in a shared space and opened the potential for future analysis of individual roles in collective performance.

The description of the individual performances brought our attention to the appropriations of Seis8s. In the performance of ASCIITOS – Barush, Nuriban, and Gabriel G – unexpected drone sounds were produced by lowering the pitch of Seis8s' accordion sample. These were accompanied by constant drum kicks from TidalCycles and abstract visuals from Punctual. The same was true for Montreal-based artist Illest Preacha who detached from Seis8s' cumbia presets to create a less specific music style that merged with complex visuals. Furthermore, Illest Preacha modified its performance after being recorded to produce a sensation different from the one Seis8s allows in real-time.

Seis8s, in the context of online installations, also provided important insights, particularly on the user experience when interacting through a graphical interface. Here, users revealed missing aspects of Seis8s as a web application outside Estuary. These included the need for visual cues to show when there is an error in the code syntax and the need to incorporate buttons to allow stopping the sound without the need to erase the code. These features have now been incorporated into Seis8s standalone, making it easier for users to interact and engage with the software.

Within the same train of thought, the ICLC and NIME conference reviewers pointed out the need for discussion documents for users to connect critique with Seis8s. Similarly, reviewers' feedback helped me bring to the front Seis8s' *education goal* by pointing out the need for better and more comprehensive documentation. In this respect, future work includes creating explanatory documents and tutorials discussing more specific aspects of the music

styles supported by Seis8s, such as cumbia and Salsa.

Additional plans include offering workshops for the general audience to learn about cumbia and other Latin music. Additional plans also include collectively reflecting on other secondary goals, such as Seis8s as a tool to teach people how to program. Perhaps here is precisely where Seis8s can offer perspectives of music software that emphasize resistance to asymmetric types of computer-music abstraction, where Afro-Latin instruments and rhythms are at the center of both the performance and the technology.

Chapter 5

General conclusions

In this dissertation, I speculated about culture, social spheres, and programming with a particular focus on Latin America. The specific purpose of my speculation was to gain insight into how computer platforms can be (re)thought and (re)designed around the consciousness and struggles of Latin American communities. The final result of this speculation took the form of Seis8s, a computer language inflected by Spanish and Latin Dance Music.

To arrive at this, nonetheless, I first asked the following questions to identify current debates in the communities and the fields I intended to explore: *What are some specific modes of consciousness –i.e., sensibilities, moral systems, belief systems, modes of knowing and being– of inhabitants of Latin America? What are their struggles –i.e., generalized difficulties and their modes of resistance? In which ways do these modes of consciousness and struggles intersect with music software? How can music software be (re)thought and (re)designed around this consciousness and struggles? What would be the implications of this (re)design?*

5.1 General findings and main takeaways

To gain insight into these questions, this dissertation foregrounded various cultural analyses that allowed me to arrive at the following conclusions. Generally speaking, I argued that computer languages are influenced by social spheres (e.g., science and arts), economic models

(e.g., knowledge economy), communication systems (e.g., natural language), and infrastructures (e.g., software collaboration protocols). And even though the languages discussed in this dissertation could be perceived as *niche* ones, I contend that the influence of *culture* in more industrial languages can also be easily perceived. Take, for example, the computer language Python where *readability* is one of its core philosophies. This *readability* is achieved through style practices for code, such as using lowercase letters when naming modules and functions (van Rossum, 2001).

Another important realization of this research was the possibility of gaining insight into the *modes of consciousness* of Latin American peoples. Of course, this research only allowed for showing just but a glimpse of such an aspect. Nonetheless, I found it quite interesting how participants' responses provided insight into some of their belief systems and ways of knowing, for example, mestizaje and latinidad. Their answers also allowed insight into shared struggles, including the difficulty of breaking the white-mestizo ideology. And they suggested modes of resistance, such as the invitation to reflect on the irreducibility of their identity and the potential for their societies to coexist with others.

Finally, a first-class component of the doctoral work was coding Seis8s, as can be seen in the section A.4, which contains hundreds of lines of code. The first version of Seis8s took me eight months to develop, and it took me between two and three additional months to develop two subsequent versions. During these months, my supervisor Dr. David Ogborn met with me every other week to discuss and help me understand the technical and conceptual requirements of the project. All the aforementioned draws attention to the substantial actions taken to produce this doctoral work, and it also helps to see the value of work other than writing within doctoral programs such as this one.

The coding skills I gained during the process of creating Seis8s are one of the most valuable personal takeaways because I can now teach people interested in building their own computer-music languages. However, gaining these coding skills was challenging. For example, I had to spend several months learning Haskell and the Reflex Platform, and it

took me most of my Ph.D. to fine-tune my coding skills to code Seis8s. To aid in my learning process, I organized various workshops, including the ones in South America mentioned in Chapter 2. I invited people to imagine mini computer-music languages I would code with them during the workshops. This approach proved helpful in increasing my coding skills and allowed me to reflect more profoundly on the conceptual requirements of my own computer language, Seis8s. It also allowed the participants of the workshops to see the potential of building their own computer-music languages.

Once I had created Seis8s, its socialization through live coding performance allowed me to reflect upon its different meanings and purpose. For me, Seis8s currently means the possibility of emphasizing resistance to asymmetric types of computer-music abstraction by bringing Afro-Latin instruments and rhythms to the center of the technology and the performance. The latter could lead to further inquiry leading to novel improvisatory and theatrical strategies aligning with critical computing, such as live coding and conscious music[1], such as cumbia villera and hard Salsa.

5.2 Reflections on the research process

Reflecting back on my research process, one of the elements I found most interesting was the possibility for me to carry out such a diverse range of research methodologies, traversing from theoretical approaches such as Critical Code Studies to hands-on ones by building Seis8s to participatory ones through the conversation circles, to live-coding performance. Using these different methodologies helped me address the topics of this dissertation from various and distinct angles. And furthermore, it helped me gain valuable experience in interdisciplinary research.

These multiple approaches also enabled the general public to engage early on with my

[1]Conscious Latin dance music refers to "texts and musical markers that poetically express political, historical, and class awareness of the shared Latino/Latin American existences, identities, and experiences" (Agurto, 2022, p. xix). This music reflects on issues of migration, poverty, and racism while also creating a sense of belonging and collectiveness through festive rhythms and melodies.

research on various levels. For example, some participants from the conversation circles also had the chance to perform live with Seis8s. In other cases, I engaged with the audience through conference presentations in Canada, Perú, and Mexico about the theoretical components of this research. The latter, although not discussed in this dissertation, helped me to organize my thoughts, contributing to the *legibility* of the overall project.

One of the flaws of this research, however, was precisely in regards to the *legibility* of the project. As mentioned in Chapter 4, ICLC and NIME reviewers suggested the need for documents such as articles or posts for users to connect Seis8s' music-making with critique about these milieus' culture, politics, and economies. This is an area for future work where I foresee myself publishing the chapters of this dissertation in journals and making shorter versions of them available through Seis8s' website. This way, they could be consumed with the music-making tutorials, allowing for a more rounded understanding of the project.

Another aspect that I look forward to continuing working on in terms of the *legibility* of the project is to focus on finding ways to produce more structured performances with Seis8s. For example, by having regular rehearsals solo and in groups. Or through the incorporation of physical instruments into the performance. This aspiration will help find novel ways of using it and increase the outreach potential of Seis8s.

5.3 Contributions to computer programming and live coding practice

From my perspective, the aforementioned issues with respect to *legibility* and even user *adoption* come from Seis8s potentially being perceived as unnecessary, silly, or even *exotic*. Nonetheless, as design scholars Dunne and Raby (2013) mention, speculative practices are "by necessity provocative, intentionally simplified, and fictional" (3). And, as an audience, it is important to understand that the true goal of this type of speculation is often to enable discussions on pressing topics in a given area. Furthermore, speculation serves "to create

spaces for discussion and debate about alternative ways of being, and to inspire and encourage people's imaginations to flow freely" (2). In summary, it is "a catalyst for collectively redefining our relationship to reality" (2).

I argue that the research presented in this dissertation contributes to the field of computer programming by enabling discussion on questions of ideology and software in Latin America. For example, *in which ways have multiculturalism and pluriculturalism influenced computer software in this region? Furthermore, what have been the implications of this?*

The latter questions, nonetheless, were a collective result of the conversation circles reported in this dissertation. These questions show the potential for computer development to be informed by ethnographic and participatory research findings. Furthermore, this joint effort points to the emerging field of *computational ethnography*, which among others, involves investigating how non-virtual worlds influence virtual ones (Brooker, 2022, 37).

Research creation, such as that related to live coding and New Interfaces for Musical Expression, can also benefit from research methodologies like *computational musicology*. This is an "interdisciplinary research area that requires the contribution of questions, methods, and insights from both musicology and computer science" (Volk et al., 2011).

Finally, another contribution this research provides is that it explores the possibility of complementing live coding practice with the field of Critical Code Studies. The intersection of these two areas of study opens the potential for finding new significance to the practice of sharing the screen with the audience. For example, it could invite the audience to *read* the multiple codes displayed by the software.

References

Aaron, S. (2016). Sonic pi–performance in education, technology and art. *International Journal of Performance Arts and Digital Media*, 12(2):171–178.

"About Latino" (2022). In Lenguajelatino.org/. https://manual.lenguajelatino.org/en/1.4.0/About-Latino.html.

Adler, J. (2016). Intermediate zones of culture. reflections on the (historiographical) problem of contemporaneity. *Andamios*, 13(31):295–311.

Agrelo, J. (2019). Rosalía and the blurry borders of what it means to be a latin artist. In https://www.motherjones.com. Retrieved May 11, 2023 from https://www.motherjones.com/media/2019/10/rosalia-and-the-blurry-borders-of-what-it-means-to-be-a-latin-artist/.

Agurto, A. E. (2022). *Salsa Consciente: Politics, Poetics, and Latinidad in the Meta-Barrio*. Michigan State University Press.

Ahmed, S. (2006). Queer phenomenology. In *Queer Phenomenology*. Duke University Press.

Albalooshi, N., Mohamed, N., and Al-Jaroodi, J. (2011). The challenges of arabic language use on the internet. In *2011 International Conference for Internet Technology and Secured Transactions*, pages 378–382. IEEE.

Ali, S. M. (2016). A brief introduction to decolonial computing. *XRDS: Crossroads, The ACM Magazine for Students*, 22(4):16–21.

Allen, M. (2013). What was web 2.0? versions as the dominant mode of internet history. *New Media & Society*, 15(2):260–275.

Anzaldúa, G. (2009/1987). La conciencia de la mestiza. *Feminisms redux: An anthology of literary theory and criticism*, 303.

Anzaldúa, G. ([1987] 2001). La conciencia de la mestiza [the conscience of the mestiza. In Bhavnani, K., editor, *Feminism and 'race'*, pages 93–107. Oxford University Press, Oxford.

"Architecture of Denmark" (2022). In Wikipedia. https://en.wikipedia.org/w/index.php?title=Architecture_of_Denmark&oldid=1070209899.

Arias, M. (2012). El círculo de conversación como estrategia didáctica: Una experiencia para reflexionar y aplicar en educación superior. *Revista Electrónica Educare*, 16(2):9–24.

Avelino, H. (2018). Mexico city spanish. *Journal of the International Phonetic Association*, 48(2):223–230.

Avila, A. S. F. and Madariaga, A. C. L. (2009). la cumbia, repertorio escolar para el conjunto instrumental y vocal desde séptimo basico a cuarto medio. Master's thesis, Universidad Tecnologica de Chile-INACAP, Chile.

Bard Wigdor, G. and Artazo, G. (2017). Pensamiento feminista Latinoamericano: Reflexiones sobre la colonialidad del saber/poder y la sexualidad [Latin American feminist thought: Reflections on the coloniality of knowledge/power and sexuality]. *Cultura y representaciones sociales*, 11(22):193–219.

Begel, A. (1996). Logoblocks: A graphical programming language for interacting with the world. *Epistemology and Learning Group MIT Media Laboratory, MIT, Boston, MA*, pages 62–64.

Blanco Arboleda, D. (2005). la música de la costa atlántica colombiana transculturalidad e identidades en méxico y latinoamérica. *Revista colombiana de antropología*, 41:171–203.

Bourdieu, P. (1984). A social critique of the judgement of taste. *Translated by R. Nice. Cambridge, Massachusetts, Harvard University Press*.

Brooker, P. (2022). Computational ethnography: A view from sociology. *Big Data & Society*, 9(1):20539517211069892.

Béhague, G. (n.d). Latin American Music, Characteristic instruments. In https://www.britannica.com/. Retrieved April 17, 2022, from https://www.britannica.com/art/Latin-American-music/Characteristic-instruments.

Calefato, P. (2008). On myths and fashion: Barthes and cultural studies. *Sign Systems Studies*, 36(1):71–81.

Castilla, M. T. L. (2014). Una mirada queer hacia la música popular urbana: análisis desde diferentes propuestas musicales con una sensibilidad queer común [a queer look at urban popular music: analysis from different musical proposals with a common queer sensitivity]. *Quadrivium*, (5):23.

Chavarría, M. P. (1997). Dimensiones Internacionales de la Comunicación [International dimensions of communication]. *Razón y Palabra 7 (2)*.

Chun, W. H. K. (2011). *Programmed visions: Software and memory*. Mit Press.

Collins, N., McLean, A., Rohrhuber, J., and Ward, A. (2003). Live coding in laptop performance. *Organised sound*, 8(3):321–330.

Collins, P. H. (2015). Intersectionality's definitional dilemmas. *Annual review of sociology*, 41(1):1–20.

Colman, A. M. (2015). declarative knowledge. In A Dictionary of Psychology. Oxford University Press. Retrieved 23 May. 2023, from https://www-oxfordreference-com.libaccess.lib.mcmaster.ca/view/10.1093/acref/9780199657681.001.0001/acref-9780199657681-e-2104.

"Coordinated Universal Time" (2022). In Wikipedia. Retrieved 23 May. 2023, from https://www-oxfordreference-com.libaccess.lib.mcmaster.ca/view/10.1093/acref/9780199657681.001.0001/acref-9780199657681-e-2104.

Cox, G. and McLean, A. (2012). *Speaking code: Coding as aesthetic and political expression*. MIT Press.

Da Costa, B. (2008). Reaching the limit. *Tactical biopolitics: Art, activism, and technoscience*, pages 365–385.

Del Angel, L. N. and Ogborn, D. (2017). Cacharpo: Co-performing cumbia sonidera with deep abstractions. In *Proceedings of the International Conference on Live Coding (ICLC)*.

Del Angel, L. N., Teixido, M., Ocelotl, E., Cotrina, I., and Ogborn, D. (2019). Bellacode: localized textual interfaces for live coding music. In *International Conference on Live Coding*, volume 4.

Di Próspero, C. E. (2016). El live coding: subjetividades y sociabilidad en nuevas prácticas artístico técnicas.

Drucker, J. (2013). Performative materiality and theoretical approaches to interface. *DHQ: Digital Humanities Quarterly*, 7(1).

Drucker, P. (1969/2017). The knowledge economy. In *The age of discontinuity: Guidelines to our changing society*, chapter 12, pages 263–286. Routledge.

Dunne, A. and Raby, F. (2013). Beyond radical design? In *Speculative everything: design, fiction, and social dreaming*, pages 1–10. MIT press.

Durán Salvador, W. O. (2009). *El lenguaje de programación y compilador del Instituto de Cálculo en 1965 [The Compiler and Programming Language of the Institute of Calculus in 1965]*. Ediciones del domo.

Eglash, R., Babbitt, W., Bennett, A., Bennett, K., Callahan, B., Davis, J., Drazan, J., Hathaway, C., Hughes, D., Krishnamoorthy, M., et al. (2017). Culturally situated design tools: Generative justice as a foundation for stem diversity. In *Moving students of color from consumers to producers of technology*, pages 132–151. IGI Global.

Eglash, R., Bennett, A., O'donnell, C., Jennings, S., and Cintorino, M. (2006). Culturally situated design tools: Ethnocomputing from field site to classroom. *American anthropologist*, 108(2):347–362.

Elliott, P. W. (2013). *Participatory action research: Challenges, complications, and opportunities*. Centre for the Study of Co-operatives, University of Saskatchewan.

Elsaesser, T. (2006). Early film history and multi-media: An archaeology of possible futures? *New media, old media: A history and theory reader*, pages 13–26.

Escobar, A. (1995). *Encountering development: the making and unmaking of the Third World*. Princeton studies in culture/power/history Encountering development. Princeton University Press.

"Esotericism" (2022). New world encyclopedia. https://www.newworldencyclopedia.org/p/index.php?title=Esotericism&oldid=1065483.

"Español estándar" (n.d). In Everybodywiki. https://es.everybodywiki.com/Espa%C3%B1ol_est%C3%A1ndar#cite_note-1.

Espinoza Agurto, A. (2014). Una sola casa: Salsa consciente and the poetics of the meta-barrio.

Eubanks, V. (2018). *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press.

"Feature vs Property - What's the difference?" (n.d). wikidiff.com. https://wikidiff.com/feature/property.

Franulic, A. (2013). Consideraciones sobre la práctica feminista de cambio lingüístico... o los destellos de insolencia [Consideraciones sobre la práctica feminista de cambio lingüístico... o los destellos de insolencia]. In andreafranulic.cl. Retrieved Jan 18, 2022 from https://andreafranulic.cl/lenguaje/consideraciones-sobre-la-practica-feminista-de-cambio-linguistico-o-los-destellos-de-

Gallant, D. J. (2010). Science, technology, engineering, and mathematics (STEM) education [White Paper]. Technical report, Ohio State University.

Galloway, A. R. (2004). Social realism in gaming. *Game studies*, 4(1):2004.

Galperina, M. (2014). Artist's notebook: Ramsey nasser. *http://animalnewyork.com/artists-notebook-ramsey-nasser/*.

"Galvanized Suns Archive" (2020). In diasporicfuturisms.com. https://www.diasporicfuturisms.com/galvanized-suns.html.

Garcés, F. (2009). De la interculturalidad como armónica relación de diversos, a una interculturalidad politizada [from interculturality as a harmonious relationship of diverse to a politicized interculturality. *Interculturalidad crítica y descolonización: fundamentos para el debate*, pages 21–50.

Garcia Canclini, N. (2005). *Hybrid Cultures Strategies for Entering and Leaving Modernity*. University of Minnesota Press, Minneapolis.

Goldenberg, E. P. and Feurzeig, W. (1987). *Exploring language with Logo*. Mit Press.

González, J. P. (2009). Musicología y américa latina: una relación posible. *Revista argentina de Musicología*, 10:43–72.

Grosfoguel, R. (2011). Decolonizing post-colonial studies and paradigms of political-economy: Transmodernity, decolonial thinking, and global coloniality. *Transmodernity: journal of peripheral cultural production of the luso-hispanic world*, 1(1).

Guerrero, C. (n.d). Cumbia de los pajaritos. Available at https://fr.scribd.com/document/335080889/Cumbia-de-Los-Pajaritos.

Guerrero, M. (2021a). Lib «lista». In *Manual Lenguaje Latino*. https://manual.lenguajelatino.org/es/stable/.

Guerrero, M. (2021b). Mi primer programa. In *Manual Lenguaje Latino*. https://manual.lenguajelatino.org/es/stable/.

Gustein, E., Lipman, P., Hernandez, P., and de los Reyes, R. (1997). Culturally relevant mathematics teaching in a mexican american context. *Journal for Research in Mathematics Education*, 28(6):709–737.

Gutiérrez, M. E. V. (2020). Racismo y afrodescendientes en méxico: cinco reflexiones para la 'deconstrucción' de las nociones de raza y mestizaje [racism and afro-descendants in mexico: Five reflections for the 'deconstruction' of the notions of race and mestizaje]. *Boletín de Antropología. Universidad de Antioquia, Medellín*, 35(59):17–34.

Hall, S. (1999). Encoding, decoding. In During, S., editor, *The cultural studies reader*, chapter 36, pages 507–519. Routledge.

Hall, S. (2000/1973). Encoding/decoding. *Media studies: A reader*, pages 51–61.

Hall, S. (2015). Cultural identity and diaspora. In *Colonial discourse and post-colonial theory*, pages 392–403. Routledge.

Hall, S. (2017/1958). A sense of classlessness. In *Selected Political Writings*, pages 28–46. Duke University Press.

Haraway, D. ([1985] 2016). A cyborg manifesto: Science, technology, and socialist-feminism in the late 20th century. In *Manifestly haraway (Vol. 37)*, pages 4–90. U of Minnesota Press.

Herrera Rivas, H. and Roque Hernández, R. V. (2019). Digital divide, english language and their link with spanish reading comprehension. *RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, 10(19).

Hopper, G. (1981). *History of programming languages I*, chapter Keynote address, pages 7–20. Academic Press.

Hudak, P. (2008). The haskell school of music. *Yale University*.

Hudak, P., Hughes, J., Peyton Jones, S., and Wadler, P. (2007). A history of haskell: being lazy with class. In *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, pages 12–1.

Jackson, J. E. and Warren, K. B. (2005). Indigenous movements in latin america, 1992–2004: Controversies, ironies, new directions. *Annu. Rev. Anthropol.*, 34:549–573.

Janovic, B. and Rousseau, P. (2005). General purpose technologies. *Nber working Paper Series*.

Karmy Bolton, E. (2013). También yo tengo mi cumbia, pero mi cumbia es chilena: apuntes para una reconstrucción sociohistórica de la cumbia chilena desde el cuerpo [i also have my cumbia, but my cumbia is chilean: notes for a sociohistorical reconstruction of chilean cumbia from the body]. *Resonancias, revista de investigación musical*, pages 93–110.

Ladson-Billings, G. (1995). Toward a theory of culturally relevant pedagogy. *American educational research journal*, 32(3):465–491.

"Latino v1.4.1" (2021). In https://www.lenguajelatino.org/. https://www.lenguajelatino.org/.

Leigh Star, S. (2010). This is not a boundary object: Reflections on the origin of a concept. *Science, technology, & human values*, 35(5):601–617.

Lialina, O. and Espenschied, D. (2015). Do you believe in users?/turing complete user. *Mass Effect: Art and the Internet in the Twenty-First Century*, pages 1–14.

Lyons, M. and Fels, S. (2014). A nime primer. NIME.

Lyotard, J. (1979). The postmodern condition: A report on knowledge (vol. 10; translated from the 1979 les editions de minuit in 1984 to english by g. bennington & b. massumi). *Minneapolis, MN: University of Minnesota Press. Retrieved from https://monoskop. org/images/e/e0/Lyotard_Jean-Francois_The_Postmodern_Condition_A_Report_on_Knowledge. pdf. Accessed on March*, 8:2019.

López, A. I. (2020). Cenart, un espacio para las artes en la ciudad de méxico. https://www.admagazine.com/lugares/cenart-un-espacio-para-las-artes-en-la-cdmx-20200223-6499-articulos#:~:text=El%20Centro%20Nacional%20de%20las%20Artes%20fue%20creado%20en%201994,arte%2C%20la%20cultura%20y%20la.

Malo de Molina, M. (2004). La encuesta y la coinvestigación obreras, autoconciencia [the worker survey and research, self-awareness]. *Nociones Comunes. Experiencias y Ensayos entre*.

Manuel, P. (1998). The course of performance: Studies in the world of musical improvisation. In Nettl, B. and Russell, M., editors, *The Course of Performance: Studies in the World of Musical Improvisation*, chapter 6, pages 127–48. University of Chicago Press, Chicago.

Marino, M. (2018). Reading culture through code. In *The Routledge Companion to Media Studies and Digital Humanities*, pages 472–482. Routledge.

Marino, M. (2020a). *Critical code studies*. MIT Press.

Marino, M. (2020b). Flow-matic. In *Critical code studies: initial methods*, Software studies, pages 129–160. The MIT Press, Cambridge, Massachusetts.

Márquez, I. (2016). Cumbia digital: Tradición y postmodernidad. *Revista musical chilena*, 70(226):53–67.

Martín, E. (2012). Cumbia villera and the end of the culture of work in argentina in the 90s. *Youth Identities and Argentine Popular Music: Beyond Tango*, pages 59–81.

Martin, E. (n.d). How to play cumbia bass. https://www.debajoelectrico.com/en/how-to-play-cumbia-bass/.

Maseda, P. (2005). El centro nacional de las artes y la política cultural. *Discurso Visual-Revista Digital - Cenidiap*, January-March(3).

Mauleón, R. (1993). *The Salsa Guidebook for Piano Ensemble*. Sher Music Co.

McCartney, A. (1994). *Creating worlds for my music to exist: How women composers of electroacoustic music make place for their voices*. York University.

McLean, A. (2011). *Artist-Programmers and Programming Languages for the Arts*. PhD thesis, Department of Computing, Goldsmiths, University of London.

McLean, A. (2014). Making programming languages to dance to: live coding with tidal. In *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, pages 63–70.

McLean, A. and Wiggins, G. A. (2011). Texture: Visual notation for live coding of pattern. In *Proceedings of the International Computer Music Conference 2011*. University of Huddersfield, UK.

McPherson, T. (2013). Us operating systems at mid-century: The intertwining of race and unix. In *Race after the Internet*, pages 21–37. Routledge.

McPherson, T. (2018). *Feminist in a Software Lab: Difference+ Design*, volume 6. Harvard University Press.

Merriam-Webster (n.d). Yield. In Merriam-Webster.com dictionary. Retrieved April 14, 2022, from https://www.merriam-webster.com/dictionary/yield.

Moncada, Z. M. and Acebedo, F. B. (2017). Círculos de paz y convivencia en los centros educativos. *Integración Académica en Psicología*.

Murphy, M. (2013). Chapter 3: A typology of multicultural policies. In *Multiculturalism: A critical introduction*, pages 30–45. Routledge.

Nasser, R. (2012a). Alb. Retrieved March 28, 2022 from http://nas.sr/---/.

Nasser, R. (2012b). The Alb Programming Language. GitHub.com. Retrieved Jan 31st, 2022 from https://github.com/nasser/---.

Nilson, C. (2007). Live coding practice. In *Proceedings of the 7th international conference on New interfaces for musical expression*, pages 112–117.

Noble, S. U. (2018). Algorithms of oppression. In *Algorithms of oppression*. New York University Press.

Ochoa, J. S., Pérez, C. J., and Ochoa, F. (2017). El libro de las cumbias colombianas. *Medellin: Universidad de Antioquia*.

OECD (2001). Understanding Digital Divide. OECD Publications. Retrieved March 26, 2022 from https://www.oecd.org/digital/ieconomy/1888451.pdf.

OEI Chile (2017). Círculos de conversación [video]. Retrieved Feb 18, 2023 from https://youtu.be/YC352JrWxUM.

"Off the Floor and Onto the Screen" (2022). In https://el.media.mit.edu/. https://el.media.mit.edu/logo-foundation/what_is_logo/logo_primer.html.

Ogborn, D. (2019). haskellish: For parsing Haskell-ish languages. In https://hackage.haskell.org/. Retrieved April 16, 2022, from https://hackage.haskell.org/package/haskellish.

Ogborn, D., Beverley, J., Brown-Hernandez, N., Briones, F., A., Gray, B., MacLean, A., Del Angel, L., Oduro, K., Park, S., Roberts, A., Rodríguez, J., Sicchio, K., Stewart, A., Testa, C., and Tsabary, E. (2022). Estuary 0.3: Collaborative audio-visual live coding with a multilingual browser-based platform. In *7th International Web Audio Conference, WAC 2022*, volume 7th. DOI: 10.5281/zenodo.6767377.

Ogborn, D. and Beverley, J. (2016). WebDirt. In https://github.com/. Retrieved April 16, 2022, from https://github.com/dktr0/WebDirt.

Ogborn, D., Beverley, J., del Angel, L. N., Tsabary, E., McLean, A., and Betancur, C. (2017). Estuary: Browser-based collaborative projectional live coding of musical patterns. In *International Conference on Live Coding (ICLC)*, volume 2017.

Ogborn, D., Tsabary, E., Jarvis, I., Cárdenas, A., and McLean, A. (2015). Extramuros: making music in a browser-based, language-neutral collaborative live coding environment. In *Proceedings of the First International Conference on Live Coding*, pages 163–69.

Papert, S. (1980). Turtle geometry: a mathematics made for learning. *Mindstorms: Children, computers and powerful ideas The Harvester Press Ltd.*

"Peronism". (2022). In Wikipedia. Retrieved May 28, 2023 from https://en.wikipedia.org/wiki/Peronism.

Quick, D. and Hudak, P. (2013). Grammar-based automated music composition in haskell. In *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*, pages 59–70.

Quijano, A. (2006). The" indigenous movement" and the pending questions in latin america. *Argumentos (México, DF)*, 19(50):51–77.

Rizvi, F. (2015). Stuart hall on racism and the importance of diasporic thinking. *Discourse: Studies in the Cultural Politics of Education*, 36(2):264–274.

Rodgers, T. (2012). Toward a feminist historiography of electronic music. *The Sound Studies Reader*, pages 475–489.

Rodgers, T. (2016). Toward a feminist epistemology of sound. *Engaging the World: Thinking after Irigaray*, pages 195–214.

Rodriguez, J. M. (2003). *Queer latinidad identity practices, discursive spaces*. Sexual cultures. New York University Press, New York.

"Routine" (n.d). In supercollider 3.12.2 online manual. Retrieved April 15, 2022, from https://doc.sccode.org/Classes/Routine.html.

Sandoval, C. ([1991] 2013). *Methodology of the Oppressed*, volume 18:Theory out of Bounds. U of Minnesota Press.

Sandoval, C. (1998). Mestizaje as method: Feminists-of-color challenge the canon. In Trujillo, C., editor, *Living Chicana Theory*, pages 352–370. Third Woman Press, Berkeley, CA.

Shifres, F. and Rosaba-Coto, G. (2017). Hacia una educación musical decolonial en y desde latinoamérica / towards a decolonial music education in latin america. *Revista Internacional de Educación Musical*, 5:85–91.

Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M. L., Minsky, M., Papert, A., and Silverman, B. (2020). History of logo. *Proceedings of the ACM on Programming Languages*, 4(HOPL):1–66.

Solomon, C. J. (1978). Teaching young children to program in a logo turtle computer culture. *ACM Sigcue Outlook*, 12(3):20–29.

Sperry Rand Corporation (1957). *Introducing a New Language for Automatic Programming*. Sperry Rand Corporation-Univac Data Processing Division.

Spinellis, D. (2003). Introduction. In *Code reading: the open source perspective*, chapter 1, pages 1–18. Addison-Wesley Professional.

Sterne, J. and Rodgers, T. (2011). The poetics of signal processing. *differences*, 22(2-3):31–53.

Szurmuk, M., Irwin, R. M., et al. (2009). *Diccionario de estudios culturales latinoamericanos*. Siglo XXI.

Teitelbaum, M. (2016). Falling behind? boom, bust, and the global race for scientific talent provider-name: Princeton university press, princeton, nj. *Economics Books*.

Trujillo, M. E. et al. (2015). Mis "manos sonoras" devoran la histérica garganta del mundo: sonoridades y colonialidad del poder. *Calle 14 revista de investigación en el campo del arte*, 10(15):54–73.

TV UNAM (2018). Utopía ch'ixi" con Silvia Rivera Cusicanqui [ch'ixi utopia with Silvia Rivera Cusicanqui]. Retrieved Feb 18, 2023 from https://www.youtube.com/watch?v=pHJkCqe2gAk.

United Nations (2019). La cooperación sur-sur, ¿qué es y por qué importa? [south-south cooperation, what is it and why does it matter?]. Retrieved Feb 8, 2023 from https://news.un.org/es/story/2019/03/1453001.

van Rossum, G. (2001). Pep 8 – style guide for python code. In https://peps.python.org. Retrieved April 14, 2023 from https://peps.python.org/pep-0008/.

Vicente, F (2018). Cuerpas y úteras [bodies and uterus]. In womxnscenter.uci.edu. Retrieved Jan 18, 2022 from https://feminarian.es/2018/02/15/cuerpas-y-uteras/.

Volk, A., Wiering, F., and van Kranenburg, P. (2011). Unfolding the potential of computational musicology. In *Proceedings of the 13th International Conference on Informatics and Semiotics in Organisations*, pages 137–144.

Wade, P. (2003). Repensando el mestizaje [Rethinking mestizaje]. *Revista colombiana de antropología*, 39:273–296.

Williams, R. (1977). *Marxism and literature*, volume 392. Oxford Paperbacks.

Womxn's Center for Success (2021). Why Womxn with a 'X' ? In womxnscenter.uci.edu. Retrieved February 19, 2023, from https://womxnscenter.uci.edu/why-womxn-with-a-x/.

Appendix A

Appendices: Talking circles, list of materials and themes

A.1 Thursday sessions

Session 1 - Thursday, Aug 27, 2020

During this session, I introduced and contextualized the circles within my doctoral studies. The participants introduced themselves and were invited to generate questions and themes to be discussed in the upcoming sessions. Questions generated by them included: What is software? How decolonial can something be if the tools are produced by the colonial? Where does mestizaje come in? How to dialogue with languages that are free software but are directed to the first world?

Session 2 - Thursday, Sep 3, 2020

During this session, we discussed how we, the participants, use inclusive language within our day-to-day interactions within and outside Latin America. We commenced reflecting on some of the questions posed last week. I suggested potential lines of work and production for our circles, including making write-ups, blog posts, live performances, and public talks. I

suggested the following readings to enable discussion: A Spanish translation of "The Master's Tools Will Never Dismantle the Master's House" (Lorde, 1979/1984/2003); and "Software Libre, entre la soberanía, la descolonización y las proyecciones tecnológicas [Free Software, between sovereignty, decolonization and technological projections]" (Terceros, 2015).

Session 3 - Thursday, Sep 10, 2020

This session was facilitated by two participants who talked about potential ways of resignifying the concept of mestizaje. We used as a framework the Cheje concept by Bolivian feminist sociologist Silvia Rivera Cusicanqui. We learn about this concept through the video "Utopía ch'ixi" con Silvia Rivera Cusicanqui [ch'ixi utopia with Silvia Rivera Cusicanqui]" (UNAM, 2018). This video was proposed by one of the facilitators. We connected this discussion with the artistic practice of both facilitators who work with Andean textiles and sound.

Session 4 - Thursday, Sep 17, 2020

This session was also facilitated by two participants. The first participant talked about the hegemonic structures and the tensions within their practice of new media conservation and restoration. The second participant helped us to reflect on the concept of Latinidad within the music and the arts. These discussions were enabled by the following materials selected by me and the participants and which include magazine articles and music and art videos: "Rosalía and the Blurry Borders of What it Means to Be a Latin Artist" (Argelo, 2019); "Renata Flores - Trap + Quechua - Tijeras ft. Kayfex" (Flores, 2018); "BARTOLINA XIXA - Ramita Seca, La colonialidad permanente" (Xixa, 2019); "Lido Pimienta - Nada — A COLORS SHOW" (Colors, 2020); "Things Change: Conservation and Display of Time-based Media Art" (Genus, 2018); "Tony Conrad- Keeping an artwork alive" (Tate, 2020).

Session 6 - Thursday, Oct 8, 2020

This session was facilitated by me, where I suggested continuing reflecting on what is Latinidad or what represents to be Latino, Latina, Latinx. We reflected on Queerness and Latinidad and we tried to connect these concepts to queer computing. Our discussion was enabled by the following readings selected by me: "Que(e)rying Latinidad" (Rodríguez, 2003); and "Critical unmaking: toward a queer computation" (Gaboury, 2018).

Session 5 - Thursday, Sep 24, 2020

I facilitated this session, where we explored Seis8s collectively by following a small tutorial developed by me for this purpose. The participants played with Seis8s using the Estuary platform which enables collective music-making. In the second hour, we reflected on the concepts of colonialism, decoloniality, decolonization, anti-colonialism. We connected these concepts with the history of Cumbia. Our discussion was enabled by the following videos: "¿Qué es decolonialidad? [What is decoloniality?]" (Ochy Curiel, 2020); and "La cumbia como matriz sonora de Latinoamérica [Cumbia as the sound matrix of Latin America]" (Darío Arboleda, 2020).

Session 7 - Thursday, Oct 15, 2020

Two participants facilitated this session, where we discussed the topics of creativity and programming. The first participant presented their work about electronic literature, programming languages, and speech. The second participant facilitated a discussion about body-machine relationships and affective relationships within language and programming from a psychological perspective. The participants selected the following material that enabled this conversation: "Lectura expandida: una actividad viso/motora y significativa en la realidad textual [Expanded literature: a visual / motor, significant activity in textual re-

ality]" (Rodríguez, Rodríguez, Brianza, ?); and "Educación emocional en Japon [Emotional education in Japan]" (COPERAT, 2015).

Session 8 - Thursday, Oct 22, 2020

A participant and I facilitated this session. We discussed the concepts of multiculturalism, pluriculturalism, and interculturality within our places of living. We reflected on how they affect the language that we use daily. During the second hour, a participant facilitated an overview of the Andean culture and sensibilities within Bolivia's music, the arts, and technology. I selected the following reading that enabled the first hour of the discussion: "Multi, pluri e interculturalidad: ¿políticas de la diferenciación colonial? [Multi, pluri and interculturality: politics of colonial differentiation?]" (Garcés, 2009). The other facilitator selected the following Bolivian movie, movie analyses, and news article to enable the second hour: "La nación clandestina [The clandestine nation]" (Sanjinés, 1989); "Cine portable – La Nación Clandestina: la mejor película boliviana de la historia [Cine portable - La Nación Clandestina: the best Bolivian film in history]" (Salvaje, 2018); "El barroco andino en La nación clandestina: una lectura desde la 'puesta en escena de la identidad mestiza' [The Andean Baroque in The clandestine nation: a reading from the 'staging of mestizo identity']" (Romero, 2017); "Por qué el reloj del Congreso de Bolivia gira al revés [Why does the clock of the Bolivian Congress turn backwards]" (BBC, 2014).

Session 9 - Thursday, Oct 22, 2020

We discussed social class, music, and software by critiquing the intentions behind the project "One Laptop per Child". I suggested the following movie to enable the discussion: "Web" (Kleimann, 2014). We also talked briefly about how spirituality might look when talking about software and technology. A participant suggested the following reading and videos to enable the discussion: "Mis manos sonoras devoran la histérica garganta del mundo [My sonorous hands devour the hysterical throat of the world]" (Mayra Estévez, 2015); and

"Bárbara Santos: La curación como tecnología [Bárbara Santos: Healing as technology]" (Santos, 2020).

Session 10 - Thursday, Nov 10, 2020

This was a farewell meeting, where some people shared their contact information to keep in touch with others, and we discussed potential future projects including the residency at Platohedro and a potential talk at the Universidad Nacional de Colombia.

A.2 Saturday sessions

Session 1 - Saturday, Aug 29, 2020

During this session, I explained the purpose of the circles and contextualized them within my doctoral project. The participants also introduced themselves and generated questions and themes to be discussed in the upcoming sessions.

Session 2 - Saturday, Sep 5, 2020

I facilitated this session, where we discussed the similarities and differences between using inclusive language in Canada and Mexico. We briefly discussed how we perceive our identities fit within Canadian society. Finally, we talked about the gentrification of cumbia. I selected the following reading that enabled our discussion: "La música de la costa atlántica colombiana transculturalidad e identidades en México y Latinoamérica [The music of the Colombian Atlantic coast transculturality and identities in Mexico and Latin America]" (Arboleda, 2005).

Session 3 - Saturday, Sep 12, 2020

I facilitated this session where we experimented with Seis8s. We then reflected on the idea or concept of Latinidad focusing on its limits of representation within mass media music. I selected the following magazine articles to enable the discussion: "Shakira tiene a la champeta de moda en el mundo: el momento dorado que vive el ritmo" (El País, 2020); and "What J-Lo and Shakira missed in their Super Bowl halftime show" (Rivera-Rideau, 2020).

Session 4 - Saturday, Sep 19, 2020

We continued discussing Latinidad and the categories of Latino, Latina, Latinx. We tried to connect these concepts to Queer theory. A participant suggested the following talk to enable the discussion: "Donna Haraway and Brigitte Baptiste at Bogota" (Idartes, 2019).

Session 5 - Saturday, Sep 26, 2020

We spent the entire session playing with Seis8. Participants commented on Seis8s and suggested some functions to incorporate into the language.

Session 6 - Saturday, Oct 10, 2020

A participant facilitated a meeting where we discussed music institutions in Mexico, such as the Mexican centre for Music and sound art (CMMAS), and festivals, such as Visiones Sonoras. We could not talk about the texts shared, but we started discussing colonization within the listening practice. The latter as no one read the suggested texts. The texts suggested were Becoming Straight". (Ahmed, 2006); and "La cumbia como matriz sonora de Latinoamérica [Cumbia as the sound matrix of Latin America]" (Darío Arboleda, 2020). A participant and I co-facilitated an experimenting session with Seis8s for about thirty minutes.

Session 7 - Saturday, Oct 17, 2020

I facilitated this session, where we discussed the heterosexual dimension of things, including programming languages and cumbia. I selected the following reading and video talks to enable the discussion: “Becoming Straight”. (Ahmed, 2006); and “La cumbia como matriz sonora de Latinoamérica [Cumbia as the sound matrix of Latin America]” (Darío Arboleda, 2020).

Session 8 - Saturday, Oct 24, 2020

We discussed an upcoming public artist talk and performance at the Factory Media Centre, to which I invited them. A participant and I co-facilitated another tutorial about Seis8s.We connected our performance with the following case study of queer cumbia, which I suggested to read and watch: “Una mirada queer hacia la música popular urbana: análisis desde diferentes propuestas musicales con una sensibilidad queer común [A queer look at urban popular music: analysis from different musical proposals with a common queer sensibility]” Castilla, 2014); and “Chica de Calendario” (Kumbia Queers, 2011).

Session 9 - Saturday, Oct 24, 2020

We rehearsed with Seis8s for a public performance and artist talk at the Factory Media Centre. I shared the following video for participants to find motivation for our public performance and artist talk: ”Salsa cimarrona y sus pregones anti/descoloniales [Salsa cimarrona and their anti/decolonial pregones]” (Glefas Latam, 2020).

A.3 Categories and overarching themes

Appendix II

nº	Categories identified	Level of importance (primary, secondary, tertiary)	Related keywords and phrases	# of related keywords and phrases	Topical themes	Overarching or conceptual themes
1	Language		Political programming languages, Language as a tool and programming languages, Inclusive language within our meetings, Languages as wound/injury, Language rupture, Musical Languages, Sonic languages, Prosthetic language, As a representation and as a detonator (what does it reaches to name?) and its affective relationship with language and tongue, Languages as container, Language incalculability and irreproducibility due to the randomness of its social development, Esolangs, Spanish, Portuguese, Quechua, Discover the linguistic and non-linguistic signs that are normative (e.g., heterosexual, western, capitalist), textual reading, Linguistic signs, name if it is decolonial, colonial or	31	***Participants used:*** - inclusive language within our meetings ***Participants perceived:*** - Language as a tool - Language as wound/injury - Language as a representation and as a detonator - Language is affective - Language as a container - Language is incalculable and irreproducible - Language develops randomly - Speaking certain languages is a result of survival - Words as a space to create and defend ourselves - English as a colonial language - Spanish as a colonial language - Quechua offers other ways of making sense of the world. ***Participants asked:*** *- How to dialogue with these languages that are free software but are still directed to the first world and that in their essence have a lot of this first world?*	- Inclusive language. - Language - Characteristics of language - Colonial languages - Language as worldview

			anticolonial, tongue, speech patterns, orality, survival (sobrevivencia or supervivencia), essence, The word and its existence as an image, aesthetic-graphic respresentation, expanded literature, electronic literature, the words as a space to create and defend ourselves			
2	Hegemonic structures		Violent processes of unification and segregation in the academy, social position, strong male presence in curatorial and conservation practice, Violence, Dominant "whiteness" in drag culture, Coloniality, UNAM, CMMAS, The market, The market and the record labels, Masculine place, the visible and the invisible, privilege, mechanisms of oppression, Warrior empire, Political divisions, migration, liberatory issue, Aymara culture, socialization, capital, accumulation, straight bodies, imposition of a masculine culture, male culture, sexism, misogyny,	30	***Participants denounced:*** - Violent processes of unification and segregation in the academy - A strong male presence in the curatorial and conservation practice - A dominant "whiteness" in drag culture - Sexism, misogyny, and objectification in the video game industry	- Hegemonic processes within the Academy - Dominant cultures within artistic practices

			and objectification in the video game industry, eugenic control, mestizaje, atrocities in the name of love			
3	Identity		Conscience as Latin Americans, Invalidation of individual and collective identity, Latino identity, Fluid identity, Queer identity, Identity politics, mestizaje, "white" people, Erasure of categories such as women or Canadian, The concept of race, Latin, Latina, Latinx, Coexistence of multiple cultural differences, phenotype, Limits of self-recognition, social position, accents, self-identification, Privilege, Indigenous, Cultural descent, mixtures, mix, Body, sexuality, migration, race,	28	**Participants described themselves as:** - Colonized people **Participants felt:** - There is some invalidation of individual and collective identity of people from Latin America. - To be from Latin America means to have lived through its precariousness **Participants consider:** - The concept of race is sometimes purposedly used by racialized communities as a political statement - People should be free to self-recognition - Self-recognition have limits - Mestizaje was synonym of citizenship in Bolivia	- The limits of self-identification - Latin America and precarity - Latin America and colonization
4	Programming		Political programming languages, Language as a tool and programming languages, Creativity and programming code as a code of conduct, Spiritual or affective programming, Genetic	21	***Participants perceived:*** - Programming languages have the essence of the first world - Code as a code of conduct - programming languages have a heterosexual dimension	- Programming languages represent assymetrical relations

			programming and vincular, Affective processes that cannot be programmed but can be influenced, Module, Esolangs, Users, live coding, non-programmable, expanded literature, electronic literature, the heterosexual dimension of things, programming languages and cumbia, sexism, misogyny, and objectification in the video game industry, knowing how to make a programming language is vital to resist the monoculture of computer science and software design, mnemonics		***Participants asked:*** *- How to dialogue with these languages that are free software but are still directed to the first world and that in their essence have a lot of this first world?*	
5	Latin rhythms		Champeta, Cumbia, Romantic salsa, Extend the Latin rhythms to other territories: Carlos Vives and the gentrification of cumbia, Tuning, Music and stereotypes, Reggaetón and clave, The market and the record labels, Willy Colón, Celso Piña, Mixtures, precarity, precariousness, migration, flow, the music of the Colombian Pacific,	20	**Participants perceived:** - Cumbia has a heterosexual dimension - Cumbia is meant to extend to other territories - The tuning in cumbia is foreign to jazz musicians - Cumbia might be queer in Canada because it potentially responds to a sexuality other than Country or Rock - Cumbia is seen as something acceptable when it is displayed	- Hegemonic dimensions of Cumbia - Sexuality in Cumbia - Aspects of cumbia that are foreign in Canada

			Cumbia and tropical music, queer cumbia		in large spaces and in a more curated and 'cultural' form. - Cumbia in Monterrey was went from an object of study to a subject of study.	
6	Latin America/Abya Ayala		Global South, Governments in Latin America, Identities and music in Latin America, Extractivism and music in Latin America, Conscience as Latin Americans, Colombia, Central America, Monterrey, Abya Ayala, Ahistorical society, The Caribbean, Spanish, Portuguese, mix, mixtures, hybridization, precariousness, Aymara culture	18	**Participants mentioned:** - Another name for Latin America is Abya Ayala - Colombia is an ahistorical society - An Andeacentrism is present in Bolvia - The use of Latin America as a term that may even exclude native peoples **Participants perceived:** - Spanish language rooted to Latin America	- Languages rooted to Latin America - 'Proper' ways to name Latin America - Oppressive particularities of Colombia and the Andes
7	Relationships		Community relationships, Computer-body relationships, Collective work, the visible and the invisible, mixtures, warrior empire, precariousness, links between nature, technology, and body, shared individual questioning, socialization, listening, sensitive gaze, design spaces together, trust, deep-rooted and generational wound, social practices	17	**Participants perceived:** - Issues about colonization and oppression are sometimes not discussed in elementary school in Mexico **Participants mentioned:** - In Spain, there is not much consciousness about colonization of the Americas - In Spain, October 12 is the day of Hispanidad. **Participants described:**	- Elementary school as socialization - National festivities/holidays as socialization - Talking circles as a socialization space

					- Our discussion circles as a place for shared individual questioning.	
8	Mestizaje		Gendered dimensions of mestizaje, Critique of mestizaje, Resignification of the concept of mestizaje outside the "white mestizo" scheme, Mixed race and/or immigrants, Exclusion of mestizx experiences within anti-racism workshops in Canada, White mestizo, Cheje, hybridization, Resignification, Biological and cultural mestizaje, mix, mixtures, Hybridization, technological mestizaje, precariousness, citizen, eugenic control	17	**Participants perceived:** - Mestizaje should be resignified outside of the "white mestizo" scheme - There is an exclusion of the mestizx experience within anti-racism workshops in Canada - There exists technological mestizaje **Participants mentioned:** - Mestizaje is synonym of citizenship in Bolivia	- Potential resignifications of mestizaje - Potential understandings of mestizaje
9	Queer theory		Post-punk and queer themes, Gay people, Dominant "whiteness" in drag culture, Hegemonic narrative of queer computing, worldview, the street in relation to nature and, sexuality, queer phenomenology, orientations informed by social practices, queer cumbia, spaces have	13	***Participants consider:*** - There is a hegemonic narrative of queer computing - Spaces have gender - Cumbia might be queer in Canada because it potentially responds to a sexuality other than Country or Rock	- Gender and sexuality of computer languages - Gender and sexuality of cumbia in Canada

			gender, cultural and biological constructions of gender, social construct of sexuality, queer sensibility			
10	Multiculturalism		Representation and colonial multiculturalism, Multicultural exchange, Violence, Pluriculturalism, Racial tolerance, mix, mixture, Political divisions, migration, biological category, multiculturality, pluriculturality, interculturality and software	13	**Participants considered:** - Multiculturalism could be confused with Westernization - Multiculturalism in Mexico propose an asymmetric relationship between Indigenous and mestizo people - Multiculturalism in Canada is understood as tolerance **Participants defined:** - Multiculturalism as a space where multiple cultures exist - Pluriculturalism as a space where cultures and their geographies stand out - Interculturality can be positive, where there are different ways of thinking and existing - Interculturality can be negative if it is crossed by politicized instances. - There is a utopian interculturality where there are complex relationships and horizontal and multidirectional negotiations.	- Traits of multiculturalism in Mexico and the Americas - Differences between multiculturalism, pluriculturalism and interculturality - Multiculturalims within nation states

					- Unlike interculturality, multiculturalism and pluriculturalism are currently visible in nation-states.	
11	Software		Free software, paid software, software developers, software documentation and manuals as archival form, algorithm, user, software design as non-neutral, software as a platform in which bodies occupy space, sexism, misogyny, and objectification in the video game industry, knowing how to make a programming language is vital to resist the monoculture of computer science and software design, multiculturality, pluriculturality, interculturality and software	12	***Participants perceive:*** - Software is algorithms - Software design is not neutral - Software is a platform in which bodies occupy space - There is sexism, misogyny, and objectification in the videogame industry - That knowing how to make a programming language is vital to resist the monoculture of computer science and software design - Programming languages is software directed to the first world ***Participants asked:*** *- What is software?* *- How to dialogue with these languages that are free software but are still directed to the first world, and that in their essence have a lot of this world?*	- Monoculture of software design - Effects of the monoculture of software
12	Resignification		Hybridization, Anti-colonialism, decoloniality, Mixtures, Precarity, Liberatory issue, creation	10	**Participants considered:** - There is a potential for resignification of the concept of	- Mestizaje as a "white mestizo" scheme

			processes outside the domestication of recurring or traditional methods, resistance, opening to other perspectives (miradas), harmonize rather than have specific knowledge, representation, female culture, prosecute or direct our intentions		mestizaje outside of the “white mestizo” scheme - Is not necessarily to ask whether technology can be decolonial, or how decolonial it can be, but rather the important thing is to find ways of resignification - The appropriation of cumbia by the "Cholombianos" in Monterrey, Mexico is more honest because it does not respond to a commercial market	- Ways of thinking the resignification of technology - Honest appropriation
13	Spirituality		Spirituality in music, Guadalupe virgin, The present, Aymara vision of the future, the Ayara vision of time and space, to salvage a more respectful worldview, essence, harmonize rather than have specific knowledge, moral sphere, the layer as one’s harmonization	10	**Participants consider:** - Harmonizing is more important that having specific knowledge **Participants mentioned:** - In Andean culture, the future is also the past	- Ways of understanding spirituality within knowledge and technolgy
14	Ideology		Mixtures, mestizaje, first world, Worldview, Precarity, National holidays, capital, accumulation, dominant ideologies, human construction, object of consciousness	10		

15	Practice of conservation and restoration		The museum tradition, new codes and ethics within the practice of conservation and restoration, Planned obsolescence and future migrations, Curators, Participatory conservation, Anachronism of the museum, Library processes, Conservation and orality, Participatory voices, Strong male presence in curatorial and conservation practice, Violence	10	**Participants denounced:** - Curatorial and conservation practices have a strong male presence	
16	Body		Subject, individual, user, subjectivity, be a colonizer, cognitive and visual motor processes, software as a platform in which bodies occupy space, straight bodies, cultural and biological constructions of gender, beings undifferentiated to nature	10	**Participants considered:** - Software as a platform in which bodies occupy space	- Ways of being a body
17	Art/Artists		Philosophy, Khipus, Worldview, Precarity, expanded literature, electronic literature, Sound and visuals	7		South American art

18	Appropriation		Musical appropriation, Cultural appropriation, Reverse appropriation, Negative appropriation vs exchange, Cumbia experiences in Monterrey: potentially an honest appropriation, dismantling vs appropriating, The street in relation to nature and sexuality, moral sphere	6	**Participants consider:** - Something has the potential to stop being a negative appropriation if the person who appropriates it, returns something in exchange to the community from which it was taken, or if a said person goes to live in the same situation as the person or community from who takes such cultural material. - It is utopian that it can be eliminated completely, it would have collateral effects **Participants asked:** - Is appropriation the same as dismantling?	- What is appropriation? - Ways of mitigating appropriation - Appropriation vs dismantling
19	Dialogue		Decoloniality, Anticolonialism and decoloniality as propaganda, feminism, Limits of dialogue and debate, Decolonization, Anti-colonialism, Delinking, unlearning, spaces of reflection	6	**Participants invited to:** - Create “dialogical relationships”.	Forms of dialogical relationships
20	Music		Post-punk and queer themes, The market and the record labels, worldview, The music of the Colombian Pacific, pro-feminist lyrics	5	**Participants considered:** - The way of listening and the sophistication of the sound accounts for some colonizations , now normalized within us	- Sophistication of sound - Sophistication as colonized thought

21	Design		Software design as non-neutral, the heterosexual dimension of things, programming languages and cumbia, orientations informed by social practices, design spaces together, Distinctions between objects and space	5	**Participants denounced:** - The heterosexual dimension of programming languages and cumbia **Participants suggested:** - Spaces where to design together	Ways of subverting oppressive design choices
22	Global North		First World, Unites States, Neatness and success stories through materiality, precariousness	4	**Participants mentioned:** - Gay narratives of success are wrapped around neatness and materiality	- Narratives from the Global North - Aspirational narratives
23	Consent		Inclusive language within our meetings, preferred pronouns, street	3	**Participants used:** - Inclusive language within our meetings - As a political statement - At the university in a regular basis	- Using inclusive language as political statement - Places where is common to use inclusive language
24	Extractivism		Extractivism and music in Latin America, Intellectual extractivism, to salvage a more respectful worldview	3		
25	Khipus		Textiles, Archeological way instead of alive, Worldview	3	**Participants mentioned:** - Khipus are studied as an archeological way rather than as alive	- Characteristics of the study of Quipus

26	Seis8s		Flow, emulate, live coding	3	**Participants mentioned:** - Playing with Seis8s has a different flow than playing with more traditional instruments	- Flow in cumbia
27	Nation		National holidays, enforced disappearance	2		
28	Reading		The intermodality of reading, nullified sensations when we read	2	**Participants perceived:** - Our senses are nullified when reading	
29	Laughter		Fun, vulnerability	2		
30	market/Marketing		Color marketing	1		
31	Today		Covid-19	1		
32	Progress		Evolution	1		

A.4 Sei8s' code

A.4.1 The Main module

The Main module compiles Seis8s into a web application.

```
{-# LANGUAGE RecursiveDo, OverloadedStrings, JavaScriptFFI,
   FlexibleContexts #-}

module Main where

import System.IO
import Control.Monad.Trans
import Control.Monad.Fix
import Reflex.Dom as Dom hiding (getKeyEvent,preventDefault)
import Data.Time
import Data.Tempo
import Data.Time.Clock.POSIX
import Data.Aeson
import Data.Text (Text)
import qualified Data.Text as T
import qualified Data.Text.IO as T
import TextShow
import Control.Concurrent.MVar
import Control.Concurrent
import Data.Map.Strict
import Control.Monad
import GHCJS.DOM.Types (HTMLCanvasElement(..),uncheckedCastTo,JSVal,
   WebGLRenderingContext)
import JavaScript.Web.AnimationFrame
import GHCJS.Concurrent
import GHCJS.DOM.EventM
import Data.Bool
import Data.Maybe
import Data.Either
import GHCJS.Marshal.Pure
import Language.Javascript.JSaddle (liftJSM, toJSVal)
import GHCJS.Types
import Sound.OSC.Datum
import Data.Text.Encoding
import JavaScript.Object
import qualified Data.Map as Map
```

```
import Data.JSString.Text
import qualified Control.Concurrent as Con
import Debug.Trace
import Data.Char as C
import Language.Javascript.JSaddle
import Sound.MusicW

import Sound.Seis8s.Program
import Sound.Seis8s.Layer (Layer, emptyLayer)
import Sound.Seis8s.Parser
import Sound.Seis8s.GlobalMaterial


data RenderState = RenderState {
    t0 :: UTCTime,
    tSystemInit :: UTCTime, -- tSystemInit :: UTCTime,
    tempo :: Tempo,
    pVar :: ([Layer], GlobalMaterial) -- ([emptyLayer],
       defaultGlobalMaterial)
    }

main :: IO ()
main = do
  hSetBuffering stdout LineBuffering
  tNow <- getCurrentTime -- IO UTC
  wd <- webdirtNode
  initializeWebAudio wd
  mainWidgetWithHead headElement (bodyElement wd) -- IO ()

webdirtNode :: IO WebDirt
webdirtNode = do
  iwdn <- initializeWebDirtNode
  wd <- liftAudioIO $ newWebDirt iwdn
  return wd

headElement :: DomBuilder t m => m ()
headElement = do
  el "title" $ text "Seis8s"
  let attrs = fromList [("rel", "stylesheet"), ("type", "text/css"), ("href"
     , "style.css")]
  -- let attrs2 = fromList [("rel", "stylesheet"), ("type", "text/css"), ("
     href", "jquery.highlight-within-textarea.css")]
  elAttr "link" attrs $ return ()
  -- elAttr "link" attrs2 $ return ()
```

```
intro :: Text
intro = "cumbia teclado;\n\
       \cumbia bajo;\n\
       \cumbia guira;\n\
       \cumbia congas;"

ejemplo2 :: Text
ejemplo2 = " -- cumbia usando el comando acordes \n\
\acordes [re m, fa, la];\n\
\acompanamiento (2 4) $ cumbia teclado;\n\
 \tumbao 2 $ cumbia bajo;\n\
 \cumbia guira;\n\
 \tumbao 3 $ cumbia congas;"


ejemplo3 :: Text
ejemplo3 = "-- cumbia psicodelica \n\
\acordes [re m, fa, sol, re m]; \n\
\punteo [1a 3a 5a 6a] [1 2 2.5 3 4] $ sample 7 $ cumbia teclado; \n\
\acompaamiento (2 4 4.5) $ sample 2 $ cumbia teclado; \n\
\tumbao [(t a) (t a) (q a) (t a) (t a) (q a), (q p) (q a) (q p) (q a)] [ 1
   1.5 2 3 3.5 4, 1 2 3 4] $ congas; \n\
\tumbao [1a 2a 3a 5a, 1a 5a 3a] [1 2 3 4, 1 3 4] $ cumbia bajo; \n\
\ritmo 1 $ jamblock; \n\
\ritmo [1 2 3 4, 1 1.5 2 2.5 3 3.5 4 4.5] $ cumbia guira;"


-- "acordes [re m, fa, la]; \n\
-- \comps \"partido\"; \n\
-- \punteo [3a, 5a] [3, 4, 1 1.5 2 2.5] $ sample 3 $ acorden;\n\
-- \acompaamiento (2 4) $ vol 0.75 $ teclado; \n\
-- \tumbao 1 $ cumbia bajo; \n\
-- \ritmo [1 2 2.5 3 4 4.5] $ guira;\n\
-- \marcha [p t p a a] [1 2 3 4 4.5] $ pan 0 $ congas;"

-- "alternar 2 (acompanamiento (1 2)) $ acompanamiento (2 4) $ cumbia
   teclado; \n\
 -- \tumbao 2 $ cumbia bajo;\n\
 -- \cumbia guira;\n\
 -- \tumbao 3 $ cumbia congas;"

ejemplo4 :: Text
ejemplo4 = "-- cumbia psicodelica 2 \n\
\acordes [re m, fa, sol, re m]; \n\
```

```
\punteo [3a 5a 6a] [1 2 2.25 2.5 3 4] $ sample 8 $ teclado; \n\
\acompaamiento (4) $ sample 0 $ cumbia teclado; \n\
\alternar 4 (tumbao 0) $ tumbao 1 $ sample 3 $ cumbia bajo; \n\
\vol 0.65 $ cumbia gira; \n\
\tumbao 0 $ cumbia congas;"

ejemplo5 :: Text
ejemplo5 = "--cumbia tejana \n\
\tempo 0.5;\n\
\acordes [do, sol, fa];\n\
\punteo [1a 5a 6a] [1 2 2.5] $ acordeon; \n\
\tumbao [1a 3a 5a] [1 3 4] $ bajo;\n\
\acompanamiento ( 2 4) $ teclado; \n\
\ritmo [1 2 3 4, 1 1.5 2 2.5 3 3.5 4, 1] $ guira;\n\
\ritmo [1 2 3 4, 2 4] $ cumbia jamblock;\n\
\marcha [(q p) (q p) (q t) (q p) (q p) (q p) (q a) (q a), (q p) (q p) (q t)
   (q p) (q a) (q p) (t a) (t a)] [1 1.5 2 2.5 3 3.5 4 4.5, 1 1.5 2 2.5 3
   3.5 4 4.5] $ congas;"

ejemplo6 :: Text
ejemplo6 = "-- balada cumbia por Kofi Oduro\n\
\tempo 0.35;\n\
\acordes [re m, la, la, fa, do, do, fa, re m, la, la];\n\
\alternar 5 (acompanamiento (1 2 4)) $ acompanamiento (1 4) $ cumbia teclado
   ; \n\
\acompanamiento (2 3 4 4) $ cumbia teclado;\n\
\sample 4 $ teclado;\n\
\tumbao 5 $ cumbia congas;\n\
\vol 0.825 $ cumbia bajo;\n\
\cumbia guira;"

  -- "alternar 2 (acompanamiento (1 2)) $ acompanamiento (2 4) $ cumbia
     teclado;\n\
  -- \alternar 2 (tumbao 3) $ tumbao 1 $ cumbia bajo;\n\
  -- \ritmo ([1 1.5 2 2.5 3 3.5 4 4.5]) $ cumbia guira;\n\
  -- \alternar 4 (tumbao 4) $ tumbao 1 $ cumbia congas"

ejemplo7 :: Text
ejemplo7 = "-- balada por Kofi Oduro \n\
\tempo 0.173; \n\
\acordes [re m, fa,fa, sol, re m, fa, fa, sol, fa,fa]; \n\
\punteo [3a 4a 5a 2a 6a] [1 1.35 2.7 2.25 2.5 3 4.7 5.4] $ sample 8 $
   teclado; \n\
\acompanamiento (2) $ sample 1 $ teclado; \n\
\alternar 3 (tumbao 0) $ tumbao 1 $ sample 2 $ bajo; \n\
```

```
\vol 0.45 $ tumbao 1 $ guira; \n\
\tumbao 0 $ congas;"

ejemplo8 :: Text
ejemplo8 = "-- son cubano \n\
\acordes [do, do, do, do, fa, fa, fa, fa]; \n\
\tumbao [\"1a\" \"5a\", \"5a\" (\"1a\" 1),1a 5a, 5a 1a] [1 4, 1 3, 1 4, 1 4]
    $ sample 4 $ bajo;\n\
\punteo [1a 3a 5a, 6a 1a, 3a 5a, 6a 1a ] [1 3 4, 2 4, 2 4, 2 4] $ sample 1 $
    teclado;\n\
\marcha [p p t p, p p a a, p p t (t a), p p (t a) (t a)] [1 2 3 4, 1 2 3 4,
   1 2 3 4, 1 2 3 4] $ congas; \n\
\jamblock;"

ejemplo9 :: Text
ejemplo9 = "-- salsa suave \n\
\tempo 0.75; \n\
\compas \"4/4\"; \n\
\acordes [do m, do m , do m, re dim ]; \n\

\tumbao 0 $ sample 1 $ teclado; \n\
\tumbao [1a 5a 1a, 5a 1a] [1 2.5 4, 2.5 4] $ sample 4 $ bajo;\n\
\marcha [p p t p p p a a] [1 1.5 2 2.5 3 3.5 4 4.5] $ congas;\n\
\preset 1 $ cumbia clave;\n\
\ritmo [ 2 4 ] $ guira"

-- "punteo [\"3a\", \"5a\"] [3, 4, 1 1.5 2 2.5 ]$ sample 3 $ cumbia
   teclado;\n\
-- \alternar 2 (acompanamiento (1 2)) $ acompanamiento (2 4) $ cumbia
   teclado; \n\
-- \tumbao 2 $ cumbia bajo; \n\
-- \cumbia guira; \n\
-- \tumbao 3 $ cumbia congas; \n"

ejemplo10 :: Text
ejemplo10 = "-- salsa rpida \n\
\tempo 0.45;\n\
\armonia [re m];\n\

\vol 0.8 $ punteo [3a 4a 5a] [1 2.5 3.5 4, 4] $ sample 1 $ teclado;\n\

\alternar 2 (acompanamiento (2.5 3.5)) $ acompanamiento (1 2 3.5 4.5) $
   sample 0 $ teclado;\n\
\tumbao [\"1a\" \"5a\" (\"1a\" 1), \"5a\" \"1a\"] [1 2.5 4, 2.5 4] $ sample
   5 $ bajo;\n\
```

```
\marcha [p p t p p p a a] [1 1.5 2 2.5 3 3.5 4 4.5] $ congas;\n\
\preset 1 $ clave;"

-- "acordes [re m, fa, la];\n\
-- \punteo [\"3a\", \"5a\"] [3, 4, 1 1.5 2 2.5 ]$ sample 3 $ cumbia
   teclado;\n\
-- \alternar 2 (acompanamiento (1 2)) $ acompanamiento (2 4) $ cumbia
   teclado; \n\
-- \alternar 2 (acompanamiento (1 2)) $ acompanamiento (2 4) $ cumbia
   teclado;\n\
-- \alternar 2 (tumbao 3) $ tumbao 1 $ cumbia bajo;\n\
-- \ritmo ([1 1.5 2 2.5 3 3.5 4 4.5]) $ cumbia guira;\n\
-- \alternar 4 (tumbao 4) $ tumbao 1 $ cumbia congas"



navigateExamples :: Int -> Text
navigateExamples 0 = intro
navigateExamples 1 = ejemplo2
navigateExamples 2 = ejemplo3
navigateExamples 3 = ejemplo4
navigateExamples 4 = ejemplo5
navigateExamples 5 = ejemplo6
navigateExamples 6 = ejemplo7
navigateExamples 7 = ejemplo8
navigateExamples 8 = ejemplo9
navigateExamples 9 = ejemplo10
otherwise = intro

navigateExamplesWidget :: MonadWidget t m => Event t () -> m (Event t Text)
navigateExamplesWidget evButton = do
  numbs <- foldDyn (+) (0 :: Int) (1 <$ evButton) -- Dynamic Int
  let numbs' = fmap (\n -> mod n 10) numbs -- Dynamic Int
  let codeExamples = fmap navigateExamples numbs' -- Dynamic Text
  return $ updated codeExamples -- Event Text

attrsForGeneralInfo :: Bool -> Map.Map T.Text T.Text
attrsForGeneralInfo b = visibility b
  where visibility True = "class" =: "contenedorTextoIntro"
        visibility False = "class" =: "contenedorTextoIntro" <> "style" =: "
           display: none"


bodyElement :: MonadWidget t m => WebDirt -> m ()
```

```
bodyElement wd = do
  mv <- liftIO $ forkRenderThreads wd
  -- divClass "titulo" $ do
    -- text "Seis8s"
    -- elAttr "a" ("href" =: "https://l.facebook.com/l.php?u=https%3A%2F%2
       Finstagram.com%2Fmariapaula.jg%3Figshid%3Dpnjbzfn31ugn%26fbclid%3
       DIwAR1nvWI1UKeRIvdkzYVsFICxaQee2cVjQLS4IbQc2DdnvbhOkwvT4tZbTH4&h=
       AT3JFjrz0ZtST7h4CFALdMsX7L2ZB9VEN0UegRP0FAYMACdy79unNDgDAHkIeHgjP4E1Z3hOgOGoguT
       -rqgmkbbmtrnEh6_NqKnCHIp7z20g") (text "Imgen por / background art by
       : @mariapaula.jg")


  elClass "div" "contenedorPrincipal" $ mdo
    dynBoolForInfo <- toggle True evClickInfo
    let dynAttrsForGeneralInfo = attrsForGeneralInfo <$> dynBoolForInfo
    elDynAttr "div" dynAttrsForGeneralInfo $ do
      divClass "titulo" $ text "Seis8s"
      tabDisplay "botonesDeIdioma" "botonesDeIdiomaListItems"
         tabMapEscogerIdioma

    (evClickPlay, evClickStop, evClickInfo, examplesButton) <- elClass "div"
        "editor" $ mdo
      (evClickPlay', evClickStop', evClickInfo', examplesButton') <- divClass
          "playEinstrucciones" $ do
        evClickInfo'' <- divClass "infoButton" $ button "?"
        examplesButton'' <- divClass "examplesButton" $ button ""
        evClickStop'' <- divClass "stopButton" $ button "" -- ([emptyLayer],
            defaultGlobalMaterial)
        evClickPlay'' <- divClass "playButton" $ button ""
        consoleInfo' <- holdDyn "Haz sonar el cdigo presionando el botn  |
           Make the code sound by pressing the  button" consoleInfo
        divClass "consoleInfo" $ dynText $ fmap T.pack consoleInfo'
        return (evClickPlay'', evClickStop'', evClickInfo'', examplesButton
           '')
    --
      consoleInfo <- divClass "textAreaEditor" $ do
        let textAttrs = constDyn $ fromList [("class", "maineditor"){--("
           class", "class-example"),--}]
        code <- do
          -- liftIO $ jq_highlight_brackets
          -- numbs <- foldDyn (+) (0 :: Int) (1 <$ examplesButton) --
             Dynamic Int
          -- let codeExamples = fmap navigateExamples numbs -- Dynamic Text
          navigateExamplesWidget' <- navigateExamplesWidget examplesButton
          textArea $ def & textAreaConfig_attributes .~ textAttrs &
```

```
            textAreaConfig_initialValue .~ intro & textAreaConfig_setValue
            .~ navigateExamplesWidget'
        e <- _element_raw . fst <$> el' "div" blank -- script or text
        let evaled = tagPromptlyDyn (_textArea_value code) evClickPlay --
           Event t Text
        let stopSound = tagPromptlyDyn (constDyn "silencio") evClickStop --
           Event t Text
        consoleInfo' <- performEvaluate' mv $ leftmost [evaled, stopSound] --
            performEvaluate' pVar evaled
        return consoleInfo'
      return (evClickPlay', evClickStop', evClickInfo', examplesButton')
    return ()


-- performEvaluate' :: (PerformEvent t m, MonadIO (Performable m)) => MVar
    ([Layer], GlobalMaterial) -> Event t Text -> m ()
performEvaluate' :: (PerformEvent t m, MonadIO (Performable m)) => MVar
   RenderState -> Event t Text -> m (Event t String)
performEvaluate' mv e = performEvent $ ffor e $ \textAreaCode -> liftIO $ do
  let p = parseLang $ T.unpack textAreaCode
  let
  case p of
    Right x -> do
      tNow <- getCurrentTime
      rs <- takeMVar mv
      putMVar mv $ rs {
        pVar = x
      }
      return ""
    Left x -> return $ "error: " ++ x


forkRenderThreads :: WebDirt -> IO (MVar RenderState)
forkRenderThreads wd = do
  tNow <- liftIO getCurrentTime
  mv <- newMVar $ RenderState {
       tSystemInit = tNow,
       t0 = tNow,
       tempo = Tempo {freq = 0.50, time=tNow, Data.Tempo.count=0},
       pVar = ([emptyLayer], defaultGlobalMaterial)
       }
  forkIO $ renderThread wd mv -- tNow evalT tempo wd pVar
  return mv

-- renderThread :: UTCTime -> UTCTime -> Tempo -> WebDirt -> MVar ([Layer
```

```
   ], GlobalMaterial) -> IO ()
-- renderThread t0 evalT tempo wd pVar = do
renderThread :: WebDirt -> MVar RenderState -> IO ()
renderThread wd mv = do
  mv' <- takeMVar mv
  mv'' <- renderer wd mv'
  putMVar mv mv''
  -- threadDelay 100000
  let renderEnd = (t0 mv'') -- (t0 mv'')
  let targetWakeUpTime = addUTCTime (-0.1) renderEnd
  tNow <- liftIO $ getCurrentTime --currentTime
  let diff = diffUTCTime targetWakeUpTime tNow
  when (diff > 0) $ liftIO $ threadDelay $ floor $ realToFrac $ diff *
     1000000
  renderThread wd mv


renderer :: WebDirt -> RenderState -> IO RenderState
renderer wd mv = do
  let iw = t0 mv
  let ew = addUTCTime (0.1 :: NominalDiffTime) iw
  let rend = renderForStandalone (pVar mv) iw ew --render (fromJust $ pVar
     mv) (tempo mv) iw ew -- [Event], i.e. Event = (UTCTime, M.Map T.Text
     Datum)
  let singleJSValevents = fmap (\(u, m) -> ((realToFrac $
     utcTimeToPOSIXSeconds u) + 0.0, fmap datumToJSVal m)) (fst rend)-- (
     realToFrac $ diffUTCTime u (tSystemInit mv), fmap datumToJSVal m)) (
     fst rend) -- [IO JSVal]
  renderedCodes <- sequence $ fmap mapTextJSValToJSVal singleJSValevents --
     IO [JSVal]
  sequence_ $ fmap (\r -> playSample wd r) renderedCodes -- [IO ()] -> IO ()
  return $ mv {t0 = ew, tempo = snd rend} -- mv {t0 = ew, tempo = snd rend}


tabMapEscogerIdioma :: MonadWidget t m => Map.Map Int (Text, m ())
tabMapEscogerIdioma = Map.fromList[ (1, ("Espaol", tabMapEspanol')),
          (2, ("English", tabMapEnIngles'))]

tabMapEspanol' :: MonadWidget t m => m ()
tabMapEspanol' = do
  tabDisplay "menuButtons" "menuContentContainer" tabMapEspanol
  return ()

tabMapEnIngles' :: MonadWidget t m => m ()
tabMapEnIngles' = do
```

```
  tabDisplay "menuButtons" "menuContentContainer" tabMapEnIngles
  return ()

tabMapEspanol :: MonadWidget t m => Map.Map Int (Text, m ())
tabMapEspanol = Map.fromList[ (1, ("Sobre Seis8s", descripcion)), (2, ("
   Referencia", referencia)), (3, ("Agradecimientos", agradecimientos))]
          --(2, ("Canal de Discord", discordEspanol), (3, ("Ejemplos",
             ejemplos)), (4, ("Referencia", referencia)), (5, ("
             Agradecimientos", agradecimientos))]

tabMapEnIngles :: MonadWidget t m => Map.Map Int (Text, m ())
tabMapEnIngles = Map.fromList[ (1, ("About Seis8s", description)), (2, ("
   Reference", reference)), (3, ("Acknowledgements", acknowledgements))]
          -- (2, ("Discord Channel", discordEnglish), (3, ("Examples",
             examples)), (4, ("Reference", reference)), (5, ("
             Acknowledgements", acknowledgements))]

discordEspanol :: MonadWidget t m => m ()
discordEspanol = divClass "discord" $ do
  text "Unete al grupo de Discord: "
  elAttr "a" ("href" =: "https://discord.gg/ygEPS8tzzz") (text "https://
     discord.gg/ygEPS8tzzz")
  blank
  text "Contacto: navarrol@mcmaster.ca"
  -- el "br" $ blank
  -- text $ "Y deja un mensaje abajo! "
  -- liftJSM $ eval ("window.addEventListener('load',function() { \n\
  -- \commentBox('5731344961241088-proj'); \n\
  -- \});" :: String)
  return ()


discordEnglish :: MonadWidget t m => m ()
discordEnglish = divClass "discord" $ do
  text "Join the discord group: "
  elAttr "a" ("href" =: "https://discord.gg/ygEPS8tzzz") (text "https://
     discord.gg/ygEPS8tzzz")
  blank
  text "Contact: navarrol@mcmaster.ca"

  -- el "br" $ blank
  -- text $ "And leave a message below! "
  -- liftJSM $ eval ("window.addEventListener('load',function() { \n\
  -- \commentBox('5731344961241088-proj'); \n\
  -- \});" :: String)
```

```
  return ()

agradecimientos :: MonadWidget t m => m ()
agradecimientos = divClass "textoIntro" $ do
  text "Este proyecto es parte de mi doctorado llamado 'Plataformas
     culturalmente situadas de msica por computadora y es apoyada por el
     Fondo Mexicano para la Cultura y las Artes (FONCA), el Consejo Mexicano
      de Ciencia y Tecnologa (CONACYT) y el Consejo de Investigacin de
     Ciencias Sociales y Humanidades de Canad."
  divClass "contacto" $ do
    text "Agradecimientos especiales a"
    elAttr "a" ("href" =: "https://l.facebook.com/l.php?u=https%3A%2F%2
       Finstagram.com%2Fmariapaula.jg%3Figshid%3Dpnjbzfn31ugn%26fbclid%3
       DIwAR1nvWI1UKeRIvdkzYVsFICxaQee2cVjQLS4IbQc2DdnvbhOkwvT4tZbTH4&h=
       AT3JFjrz0ZtST7h4CFALdMsX7L2ZB9VEN0UegRPOFAYMACdy79unNDgDAHkIeHgjP4E1Z3hOgOGoguTiy
       -rqgmkbbmtrnEh6_NqKnCHIp7z20g") (text "@mariapaula.jg")
    divClass "contacto" $ text "por la imgen que aparece detrs del editor."
  divClass "contacto" $ text "Contctame a traves de navarrol@mcmaster.ca"


acknowledgements :: MonadWidget t m => m ()
acknowledgements = divClass "textoIntro" $ do
  text "This project is part of my doctoral project called 'Culturally
     situated platforms for computer music' and is supported by the Mexican
     Fund for Culture and Arts (FONCA), the Mexican Council of Science and
     Technology (CONACYT), and Canadas Social Sciences and Humanities
     Research Council."
  divClass "contacto" $ do
    text "Special thanks to"
    elAttr "a" ("href" =: "https://l.facebook.com/l.php?u=https%3A%2F%2
       Finstagram.com%2Fmariapaula.jg%3Figshid%3Dpnjbzfn31ugn%26fbclid%3
       DIwAR1nvWI1UKeRIvdkzYVsFICxaQee2cVjQLS4IbQc2DdnvbhOkwvT4tZbTH4&h=
       AT3JFjrz0ZtST7h4CFALdMsX7L2ZB9VEN0UegRPOFAYMACdy79unNDgDAHkIeHgjP4E1Z3hOgOGoguTiy
       -rqgmkbbmtrnEh6_NqKnCHIp7z20g") (text "@mariapaula.jg")
    divClass "contacto" $ text "for the editor's background image."
  divClass "contacto" $ text "To contact me, send me an email to
     navarrol@mcmaster.ca"


descripcion :: MonadWidget t m => m ()
descripcion = divClass "textoIntro" $ do
 divClass "descripcion" $ text "Seis8s es un lenguaje informtico basado en
    la web que permite la interaccin en tiempo real con audio digital y
    conocimientos musicales localizados. Seis8s gira en torno a comandos que
     se relacionan con la msica latina bailable, tambin conocida como msica
```

```
    latina urbana o msica popular latina. Si deseas saber ms, agregate al
    canal de Discord de Seis8s:"
 elAttr "a" ("href" =: "https://discord.gg/ygEPS8tzzz") (text "https://
    discord.gg/ygEPS8tzzz")
 elClass "h3" "empiezaAquiTitulo" $ text "Cmo usar Seis8s?"
 elClass "ol" "empiezaAqui" $ do
   el "li" $ text "Presiona  para tocar el cdigo de ejemplo que aparece en
      el editor a la derecha. Presiona  para detenerlo."
   el "li" $ text "Explora ms ejemplos presionando  y despus presionando
      para tocar cada ejemplo."
   el "li" $ text "Aprende cmo modificar los ejemplos o crear un nuevo cdigo
       consultando la seccin de Referencia en el men de arriba."

 -- text "seis8s (pronunciado 'seis octavos') es un lenguaje de programacin
     que permite la interaccin en tiempo real con audio digital y
    conocimiento musical localizado, particularmente de msicas de
    Latinoamrica. Seis8s es un proyecto reciente que pretende ser
    colaborativo, a travs de conocimiento musical consensuado desde las
    diferentes fronteras personales y colectivas que existen en conexin
    con Amrica Latina. Seis8s tambin espera ser una crtica ideolgica del
    sistema mundial de msica por computadora dominante en lugar de una
    abstraccin acrtica de las distintas visiones del mundo. El primer '
    mdulo' de seis8s produce msica influenciada por la cumbia sonidera, un
     estilo particular de la clase trabajadora mexicana en Mxico y Estados
     Unidos. Para obtener ms informacin sobre Cumbia sonidera, consulte el
     libro "
 -- elAttr "a" ("href" =: "http://beyond-digital.org/sonideros/EPS%20Libro
    -%20Sonideros%20en%20las%20aceras,%20vengase%20la%20gozadera%20-%20
    PDFvert.pdf") (text "Sonideros en las aceras, vngase a gozadera.")
 return ()
   -- text "Tambin puedes unirte al grupo de Discord para continuar la
      conversacin y preguntar cosas:"
   -- link "https://discord.gg/ygEPS8tzzz"
   -- return ()


description :: MonadWidget t m => m ()
description = divClass "textoIntro" $ do
  text "Seis8s is a web-based computer language that allows real-time
     interaction with digital audio and localized musical knowledge. Seis8s
     revolves around commands that relate to Latin dance music also known as
      urban Latin music or Latin popular music. Join Seis8s' Discord channel
      to know more:"
  elAttr "a" ("href" =: "https://discord.gg/ygEPS8tzzz") (text "https://
     discord.gg/ygEPS8tzzz")
```

```
  elClass "h3" "empiezaAquiTitulo" $ text "How to start using Seis8s?"
  elClass "ol" "empiezaAqui" $ do
    el "li" $ text "Click  to play the example code from the editor on the
       right. Click  to stop the sound."
    el "li" $ text "Explore more examples by clicking  and  to play the
       example."
    el "li" $ text "Learn how to modify the examples or create your own code
        by consulting the Reference section on the menu above."

  -- text "seis8s (pronounced 'seis octavos') is a programming language
     that allows real-time interaction with digital audio and localized
     musical knowledge, particularly of Latin American music. Seis8s is a
     recent project that aims to be collaborative, through consensual
     musical knowledge from the different personal and collective borders
     that exist in connection with Latin America. Six8s also hopes to be
     an ideological critique of the dominant world computer music system
     rather than an uncritical abstraction of various worldviews. The
     first 'module' of six8s produces music influenced by the cumbia
     sonidera, a particular style of the Mexican working class in Mexico
     and the United States. For more information on Cumbia sonidera, see
     the book "
  -- elAttr "a" ("href" =: "http://beyond-digital.org/sonideros/EPS%20Libro
     -%20Sonideros%20en%20las%20aceras,%20vengase%20la%20gozadera%20-%20
     PDFvert.pdf") (text "Sonideros en las aceras, vngase a gozadera.")
  return ()

  -- text "You can also join the Discord group to continue the conversation
      and ask questions:"
  -- link "https://discord.gg/ygEPS8tzzz"
  -- return ()



ejemplos :: MonadWidget t m => m ()
ejemplos = divClass "ejemplosCss" $ do
  text "Copie cualquiera de los siguientes bloques de cdigo y pguelo en el
     editor de texto de la derecha. Hazlo sonar presionando el boton ."
  text "Para silenciar los sonidos borra tu cdigo y vuelve a presionar el
     botn ."
  let textAttrs = constDyn $ fromList [("class", "ejemCode")]
  let textAttrs' = constDyn $ fromList [("class", "ejemCodeLargo")]

  -- liftIO $ jq_highlight_brackets
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ intro -- text
```

```
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ ejemplo2 -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs' &
     textAreaConfig_initialValue .~ ejemplo3 -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs' &
     textAreaConfig_initialValue .~ ejemplo4 -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs' &
     textAreaConfig_initialValue .~ ejemplo5 -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs' &
     textAreaConfig_initialValue .~ ejemplo6 -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs' &
     textAreaConfig_initialValue .~ ejemplo7 -- text
  return ()


examples :: MonadWidget t m => m ()
examples = divClass "ejemplosCss" $ do
  text "Copy any of the following code blocks and paste it into the text
     editor on the right. Make it sound by pressing the  button."
  text $ "To silence the sounds, delete your code and press the  button
     again (it will take a few seconds to stop, but if it doesn't stop,
     please reload the webpage)."
  let textAttrs = constDyn $ fromList [("class", "ejemCode")]
  let textAttrs' = constDyn $ fromList [("class", "ejemCodeLargo")]
  -- liftIO $ jq_highlight_brackets
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ intro -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ ejemplo2 -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs' &
     textAreaConfig_initialValue .~ ejemplo3 -- text
  textArea $ def & textAreaConfig_attributes .~ textAttrs' &
     textAreaConfig_initialValue .~ ejemplo4 -- text
  return ()


referencia :: MonadWidget t m => m ()
referencia = divClass "referenciaCss" $ do
  text "Prueba seis8s escribiendo el siguiente cdigo: "
  let textAttrs = constDyn $ fromList [("class", "refCode")]
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ "cumbia bajo;"
  text "Hazlo sonar presionando el boton . Una vez que lo ejecutes, debes
     escuchar el sonido de un bajo con el ritmo base de la cumbia!"
```

```
el "h3" $ text "Instrumentos"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "teclado; congas; jamblock; guira; bajo;
   "
text "Para silenciar los sonidos borra todo el cdigo y ejecuta de nuevo.
   Tambien puedes utilizar el comando:"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "silencio"

el "h3" $ text "Sintaxis_bsica"
text "La funcin de estilo se coloca a la izquierda del instrumento: "
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "cumbia teclado;"
text "Todas las funciones que modifican el estilo se agregan a la
   izquierda del instrumento seguidas de un parntesis o un signo de peso '
   $': "
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompaamiento 2 (cumbia teclado);"
text "El signo $ es equivalente a los parntesis: "
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompaamiento 2 $ cumbia teclado;"
text "Para silenciar los sonidos puedes utilizar el comando:"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "silencio"

el "h3" $ text "Funciones_del_bajo"
divClass "imagen-bajo" $ return ()
elClass "span" "comandosCss" $ text "sample "
text "permite cambiar la muestra de audio o sample. Acepta nmeros enteros
   iguales o mayores que 0. Hay 4 pre-sets disponibles."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "sample 0 $ cumbia bajo;"
elClass "span" "comandosCss" $ text "tumbao "
text "permite accessar a los distintos pre-sets del bajo. Acepta nmeros
   enteros iguales o mayores que 0."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao 1 $ cumbia bajo;"
text "Con la funcin "
elClass "span" "comandosCss" $ text "tumbao "
text $ "tambin puedes sobreescribir las notas de los tumbaos."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao (\"1a\" \"3a\" \"5a\") $ cumbia
   bajo;"
text "Tambin puedes reesecribir los ritmos de los tumbaos."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
```

```
   textAreaConfig_initialValue .~ "tumbao (\"1a\" \"5a\") (1 3) $ cumbia
   bajo;"
text "Tambin se pueden hacer listas de notas y ritmos de la siguiente
   forma: "
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao [\"1a\" \"5a\", \"1a\" \"3a\"
   \"5a\"] [1 3, 1 3 4] $ cumbia bajo;"

el "h3" $ text "Funciones_del_teclado"
divClass "imagen-teclado" $ return ()
elClass "span" "comandosCss" $ text "sample "
text "permite cambiar la muestra de audio o sample. Acepta nmeros enteros
   iguales o mayores que 0."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "sample 1 $ cumbia teclado;"
elClass "span" "comandosCss" $ text "tumbao "
text "permite accessar a los distintos pre-sets del teclado como lo
   hicimos con el bajo arriba. Acepta nmeros enteros iguales o mayores que
    0."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao 2 $ cumbia teclado;"
elClass "span" "comandosCss" $ text "acompanamiento "
text "o "
elClass "span" "comandosCss" $ text "acompaamiento "
text "modifica el ritmo en el que se tocan los acordes o bloques de notas
   del teclado."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 $ cumbia teclado;"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento (2 4) $ cumbia teclado;"
text "Tambin sirve para modificar las notas del teclado. Acepta hasta un
   mximo de 4 notas."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 (\"1a\" \"3a\" \"5a\")
    $ cumbia teclado;"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento (2 4) (\"1a\" \"3a\" \"5
   a\")"
text "Tambin se puede modificar la octava de la nota, es decir que tan
   grave o agudo suena."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 (\"1a\" \"3a\" (\"5a\"
    1)) $ cumbia teclado;"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 (\"1a\" \"3a\" (\"5a\"
```

```
    (-1)) $ cumbia teclado;"

el "h3" $ text "Funciones_de_las_congas"
divClass "imagen-congas" $ return ()
elClass "span" "comandosCss" $ text "tumbao "
text "permite accessar a los distintos pre-sets de las congas como lo
   hicimos con teclado y el bajo arriba. Acepta nmeros enteros iguales o
   mayores que 0. Hay 4 disponibles."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao 4 $ cumbia congas;"
elClass "span" "comandosCss" $ text "marcha "
text $ "permite asignar los golpes de la palma, tapado y abierto a las
   congas. Por default suena el tambor quinto de la conga"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha (\"p\" \"t\" \"p\" \"a\") $
   cumbia congas;"
text $ "Tambin permite escribir el rimto de las congas."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha (\"p\" \"t\" \"p\" \"a\" \"a\")
   (1 2 3 4 4.5) $ cumbia congas;"
text $ "Tambin permite accesar a otros tambores de las congas como la
   tumbadora."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha (\"p\" \"t\" \"p\" (t \"a\") (t
   \"a\")) (1 2 3 4 4.5) $ cumbia congas;"
text "Tambin se pueden hacer listas de notas y ritmos de la siguiente
   forma."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha [\"p\" \"t\" \"p\" \"a\", \"p\"
   \"t\" \"p\" (t \"a\") (t \"a\")] [1 2 3 4, 1 2 3 4 4.5] $ cumbia congas
   ;"

el "h3" $ text "Funciones_de_la_guira"
divClass "imagen-guira" $ return ()
elClass "span" "comandosCss" $ text "preset "
text " permite accesar a algunos ritmos pre-cargados de la guira. Hay 2
   disponibles."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "preset 0 $ cumbia guira;"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "preset 1 $ cumbia guira;"

el "h3" $ text "Funciones_del_jamblock"
divClass "imagen-jam" $ return ()
textArea $ def & textAreaConfig_attributes .~ textAttrs &
```

```
    textAreaConfig_initialValue .~ "cumbia jamblock;"
  text "Ms funciones vienen en camino!"

  return ()


reference :: MonadWidget t m => m ()
reference = divClass "referenciaCss" $ do
  text "Try seis8s by writing the following code: "
  let textAttrs = constDyn $ fromList [("class", "refCode")]
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "cumbia bajo;"
  text "Make it sound by pressing the  button. Once you play it you should
    hear the sound of a bass with the base rhythm of the cumbia!"

  el "h3" $ text "Instruments"
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "teclado; congas; jamblock; guira; bajo;
    "
  text "To silence the sounds you can use the command: "
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "silencio"

  el "h3" $ text "Basic syntax"
  text "The style function is placed to the left of the instrument: "
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "cumbia teclado;"
  text "All the functions that modify the style are added to the left of the
     instrument followed by a parenthesis or a '$' sign: "
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "acompaamiento 2 (cumbia teclado);"
  text "The $ sign is equivalent to parentheses: "
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "acompaamiento 2 $ cumbia teclado;"
  text "To silence the sounds you can use the command: "
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "silencio"

  el "h3" $ text "Bass functions"
  divClass "imagen-bajo" $ return ()
  elClass "span" "comandosCss" $ text "sample "
  text "allows you to change the audio sample or sample. Accepts whole
    numbers equal to or greater than 0. There are 4 pre-sets available."
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
    textAreaConfig_initialValue .~ "sample 0 $ cumbia bajo;"
```

```
elClass "span" "comandosCss" $ text "tumbao "
text "allows access to the different bass pre-sets. Accepts whole numbers
   equal to or greater than 0."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao 1 $ cumbia bajo;"
text "With the function "
elClass "span" "comandosCss" $ text "tumbao "
text $ "you can also overwrite the notes of the tumbaos.s"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao (\"1a\" \"3a\" \"5a\") $ cumbia
   bajo;"
text "You can also rewrite the rhythms of the tumbaos."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao (\"1a\" \"5a\") (1 3) $ cumbia
   bajo;"
text "Note and rhythm lists can also be made as follows: "
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao [\"1a\" \"5a\", \"1a\" \"3a\"
   \"5a\"] [1 3, 1 3 4] $ cumbia bajo;"

el "h3" $ text "Functions of the keyboard"
divClass "imagen-teclado" $ return ()
elClass "span" "comandosCss" $ text "sample "
text "allows you to change the audio sample or sample. Accepts whole
   numbers equal to or greater than 0."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "sample 1 $ cumbia teclado;"
elClass "span" "comandosCss" $ text "tumbao "
text "allows you to access the different keyboard pre-sets as we did with
   the bass on top. Accepts whole numbers equal to or greater than 0."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao 2 $ cumbia teclado;"
elClass "span" "comandosCss" $ text "acompanamiento "
text "o "
elClass "span" "comandosCss" $ text "acompaamiento "
text "changes in rhythm chords of the keyboard."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 $ cumbia teclado;"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento (2 4) $ cumbia teclado;"
text "It is also used to modify the notes on the keyboard. Accept up to a
   maximum of 4 notes."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 (\"1a\" \"3a\" \"5a\")
    $ cumbia teclado;"
```

```
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento (2 4) (\"1a\" \"3a\" \"5
   a\")"
text "You can also modify the octave of the note, that is, how low or high
    it sounds."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 (\"1a\" \"3a\" (\"5a\"
    1)) $ cumbia teclado;"
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "acompanamiento 2 (\"1a\" \"3a\" (\"5a\"
    (-1)) $ cumbia teclado;"

el "h3" $ text "Conga functions"
divClass "imagen-congas" $ return ()
elClass "span" "comandosCss" $ text "tumbao "
text "allows you to access the different pre-sets of the congas as we did
   with the keyboard and the bass on top. Accepts whole numbers equal to
   or greater than 0. There are 4 available."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "tumbao 4 $ cumbia congas;"
elClass "span" "comandosCss" $ text "marcha "
text $ "allows you to assign the hits of the palm, covered and open to the
    congas. By default the fifth drum of the conga plays."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha (\"p\" \"t\" \"p\" \"a\") $
   cumbia congas;"
text $ "It also allows you to write the rhythm of the congas."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha (\"p\" \"t\" \"p\" \"a\" \"a\")
   (1 2 3 4 4.5) $ cumbia congas;"
text $ "It also allows access to other conga drums such as the tumbadora."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha (\"p\" \"t\" \"p\" (t \"a\") (t
   \"a\")) (1 2 3 4 4.5) $ cumbia congas;"
text "Note and rhythm lists can also be made as follows."
textArea $ def & textAreaConfig_attributes .~ textAttrs &
   textAreaConfig_initialValue .~ "marcha [\"p\" \"t\" \"p\" \"a\", \"p\"
   \"t\" \"p\" (t \"a\") (t \"a\")] [1 2 3 4, 1 2 3 4 4.5] $ cumbia congas
   ;"

el "h3" $ text "Functions of te gira"
divClass "imagen-guira" $ return ()
elClass "span" "comandosCss" $ text "preset "
text "allows you to access some pre-loaded rhythms of the guide. There are
    2 available."
```

```
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ "preset 0 $ cumbia guira;"
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ "preset 1 $ cumbia guira;"

  el "h3" $ text "Funciones_del_jamblock"
  divClass "imagen-jam" $ return ()
  textArea $ def & textAreaConfig_attributes .~ textAttrs &
     textAreaConfig_initialValue .~ "cumbia jamblock;"
  text "More functions are coming on their way!"
  return ()

-- foreign import javascript safe
-- -- "document.querySelector('#estuary-root')"
-- -- "$('.array-example').highlightWithinTextarea({highlight: ['1', '2',
   '3', '4', '5', '6', '7', '8', '9', '0']});"
-- "function myFunction() { \n\
-- \document.getElementById(\"myTextarea\").select(); \n\
-- \}"
-- js_arrayexample :: IO JSVal
--
-- newtype JQuery = JQuery JSVal
-- -- newtype Event = Event JSVal
--
-- foreign import javascript unsafe
-- "jQuery(document).ready(function(){$('textarea').select()});"
-- jq_select :: IO JQuery

--
-- foreign import javascript unsafe -- "$('.array-example').
   highlightWithinTextarea({highlight: ['1', '2', '3', '4', '5', '6', '7',
    '8', '9', '0']});"
-- "jQuery(document).ready(function(){$('.class-example').
   highlightWithinTextarea({\n\
-- \highlight: [{highlight: ['$', '--', '(', ')', '[', ']'], className: '
   green' }, \n\
-- \{highlight: [ ' '], className: 'black'}, \n\
-- \{highlight: [ 'cumbia'], className: 'vino'}, \n\
-- \{highlight: [ 'alternar', 'tumbao', 'ritmo', 'acompanamiento', '
   acompaamiento', 'marcha', 'punteo', 'sample'], className: 'red'}, \n\
-- \{highlight: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0'],
   className: 'green'}, \n\
-- \{highlight: ['teclado', 'bajo', 'guira', 'guiro', 'gira', 'giro', 'jam
   ', 'congas', 'acordeon', 'silencio']}, \n\
-- \{highlight: ['acordes', 'armonia', 'armona', 'compas', 'comps'],
```

```
   className: 'yellow'}] \n\
-- \})});"
-- jq_highlight_brackets :: IO JQuery
-- -- \, {highlight: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0']}]
   \n\

  -- "jQuery(document).ready(function(){$('.class-example').
     highlightWithinTextarea({\n\
  -- \highlight: [{highlight: '$' , className: 'red' }, \n\
  -- \{highlight: '(', className: 'red'}, {highlight: ')', className: 'red
     '}, \n\
  -- \{highlight: '[', className: 'blue'}, {highlight: ']', className: '
     blue'}, \n\
  -- \{highlight: ['teclado', 'bajo', 'guira', 'jam', 'congas', 'acordeon
     ']}] \n\
  -- \})});"

-- foreign import javascript unsafe -- $('.array-example').
   highlightWithinTextarea({highlight: ['orange', /ba(na)*/gi, [0, 5] ]});
-- "jQuery(document).ready(function(){$('.array-example').
   highlightWithinTextarea({\n\
-- \highlight: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0']})});"
-- jq_highlight_numbers :: IO JQuery


type NoteEvent = (UTCTime, Map Text Datum)

datumToJSVal :: Datum -> JSVal
datumToJSVal (Int32 x) = pToJSVal x
datumToJSVal (Double x) = pToJSVal x
datumToJSVal (ASCII_String x) = pToJSVal $ decodeUtf8 x
datumToJSVal _ = nullRef

utcTimeToAudioSeconds :: (UTCTime,Double) -> UTCTime -> Double
utcTimeToAudioSeconds (t0utc,t0audio) t1utc = realToFrac $
   utcTimeToPOSIXSeconds $ addUTCTime clockDiff t1utc
  where clockDiff = realToFrac t0audio - utcTimeToPOSIXSeconds t0utc

utcTimeToAudioSeconds' :: (UTCTime,Double) -> UTCTime -> Double
utcTimeToAudioSeconds' (t0utc,t0event) tSystemInit = realToFrac $
   utcTimeToPOSIXSeconds $ addUTCTime clockDiff tSystemInit
  where clockDiff = realToFrac t0event - utcTimeToPOSIXSeconds t0utc
  --
  -- 11.10 - 11.00 = 0.20
  -- 0.10 + 11.00 = 11.10
```

```
  -- 0.20 + 11.10 = 11.30

-- what is the Double on (UTCTime,Double)?
noteEventToWebDirtJSVal :: UTCTime -> (NominalDiffTime, Map.Map Text JSVal)
   -> IO JSVal
noteEventToWebDirtJSVal tnow (s,m) = do
    let t' = addUTCTime s tnow -- 1 14.53 = 14.54
    let t'' = diffUTCTime t' tnow -- 14.54 - 14.53 = 0.01 --NominalDiffTime
    mapTextJSValToJSVal (realToFrac t'', m)

newtype WebDirt = WebDirt JSVal

instance PToJSVal WebDirt where pToJSVal (WebDirt x) = x

newWebDirt :: AudioIO m => Node -> m WebDirt
newWebDirt n = do
  ctx <- audioContext
  liftIO $ js_newWebDirt ctx n

foreign import javascript unsafe
  "$1.initializeWebAudio()"
  initializeWebAudio :: WebDirt -> IO ()

foreign import javascript unsafe
  "$r = new WebDirt('static/samples/sampleMap.json','static/samples',0,null
     ,0.010,$1,$2)"
  js_newWebDirt :: AudioContext -> Node -> IO WebDirt

foreign import javascript unsafe
  "try { $1.playSample($2) } catch(e) { console.log(e)} "
  playSample :: WebDirt -> JSVal -> IO ()

-- temporary, just for testing
foreign import javascript unsafe
  "try { $1.playSample({ buffer: $2 }) } catch(e) { console.log(e)} "
  playBuffer :: WebDirt -> JSVal -> IO ()

foreign import javascript unsafe
 "commentBox('5731344961241088-proj')"
 commentbox :: IO ()


initializeWebDirtNode :: IO Node
initializeWebDirtNode = liftAudioIO $ do
  webDirtOutput' <- createGain 1.0
```

```
  dest <- createDestination
  connectNodes webDirtOutput' dest --clave!
  let wdout = webDirtOutput'
  return wdout

mapTextJSValToJSVal :: (Double, Map.Map Text JSVal) -> IO JSVal
mapTextJSValToJSVal (t,m) = do
  o <- create
  unsafeSetProp "whenPosix" (pToJSVal t) o
  Map.traverseWithKey (\k v -> unsafeSetProp (textToJSString k) v o) m
  return $ jsval o
```

A.4.2 The Program module

The Program module that runs the Seis8s program.

```
module Sound.Seis8s.Program where

import Sound.Seis8s.GlobalMaterial
import Sound.Seis8s.Style as S
import Sound.Seis8s.Layer

import Data.IntMap.Strict


-- type Program = (GlobalMaterial,Style,Instrument)
-- data Layer = Layer (S.Style, Instrument)
type Program = ([Layer], GlobalMaterial)
-- type Program = (Layer, GlobalMaterial)

-- emptyLayer = Layer (defaultStyle, emptyInstrument)
```

A.4.3 The Parser module

The Parser module translates functions into commands available to the end user.

```
{-# LANGUAGE OverloadedStrings #-}

module Sound.Seis8s.Parser (parseLang, render, renderForStandalone) where

import Sound.Seis8s.Program
import Sound.Seis8s.GlobalMaterial
```

```
import Sound.Seis8s.Style as S
import Sound.Seis8s.Layer
import Sound.Seis8s.LayerState
import Sound.Seis8s.Harmony
import Sound.Seis8s.Rhythm
import Sound.Seis8s.Generic

import Language.Haskellish as LH
import qualified Language.Haskell.Exts as Exts
import Control.Applicative
import Data.IntMap.Strict
import Control.Monad.State
import qualified Data.Map as Map
import qualified Sound.OSC as H
import qualified Data.Text as T
import qualified Data.List as List
import Data.List.Split (splitOn)
import Data.Bifunctor
import Data.Tempo
import Data.Time
import Data.Fixed
import Data.Maybe
import Data.Char as C
import qualified Sound.OSC as H

type H = Haskellish GlobalMaterial

-- data Program = Program [Layer] GlobalMaterial
-- type Program = ([Layer], GlobalMaterial)

-- f :: (Style, Intrument)
-- f cumbia teclado

-- so I can do :
-- cumbia teclado
-- (noDownBeats cumbia) teclado

parseLang :: String -> Either String ([Layer], GlobalMaterial)
parseLang s | all C.isSpace s' = return ([emptyLayer], defaultGlobalMaterial
   )
            | otherwise = (f . Exts.parseExp) $ ( "do {" ++ s' ++ "}" )
            -- | otherwise = (f . Exts.parseExp) $ ( "do {" ++ s ++ "}" )
    where
      s' = preParse s
```

```
    f (Exts.ParseOk x) = runHaskellish layers defaultGlobalMaterial x --
       Either String (a, st)
    f (Exts.ParseFailed l s) = Left s


preParse :: String -> String
preParse s = do
  let s' = T.replace (T.pack "do") (T.pack "di") (T.pack s) -- Text
  let s'' = T.replace (T.pack "M") (T.pack "maj") s' -- Text
  let s''' = T.replace (T.pack "#") (T.pack "aug") s'' -- Text
  let s'''' = T.replace (T.pack "J") (T.pack "justa") s''' -- Text
  let s''''' = T.replace (T.pack "tecladi") (T.pack "teclado") s'''' -- Text
  let s'''''' = T.replace (T.pack "partidi") (T.pack "partido") s''''' --
     Text
  let s''''''' = LH.removeComments $ T.unpack s''''''
  s'''''''


layers :: H [Layer]
layers = listOfDoStatements statement

statement :: H Layer
statement = parseLayer <|> globalStatement <|> silencio

-- GlobalMaterial
-- e.g. clave dosTres --gets parsed 1st that changes the state
-- then harmony Cmaj Emin -- 2nd, and changes the state
-- then clave tresDos -- 3rd -- and changes the state

globalStatement :: H Layer
globalStatement = do
  f <- globalMaterialFunctions -- progressionToGm
  st <- get
  let newState = f st
  put newState
  return emptyLayer

silencio :: H Layer
silencio = emptyLayer <$ reserved "silencio"

globalMaterialFunctions :: H (GlobalMaterial -> GlobalMaterial)
globalMaterialFunctions = parseSetChordProg
                    <|> parseCompasPartido
                    <|> parsetempoAGlobalMaterial
```

```
-- comps "partido"
-- comps ""
-- compas "4/4"
parseCompasPartido :: H (GlobalMaterial -> GlobalMaterial)
parseCompasPartido = parseCompasPartido' <*> string

parseCompasPartido' :: H (String -> GlobalMaterial -> GlobalMaterial)
parseCompasPartido' = compasAGlobalMaterial <$ (reserved "compas" <|>
   reserved "comps")


compasAGlobalMaterial :: String -> GlobalMaterial -> GlobalMaterial
compasAGlobalMaterial s gm = gm {compas = establecerCompas s}


parsetempoAGlobalMaterial :: H (GlobalMaterial -> GlobalMaterial)
parsetempoAGlobalMaterial = parsetempoAGlobalMaterial' <*> double

parsetempoAGlobalMaterial' :: H (Double -> GlobalMaterial -> GlobalMaterial)
parsetempoAGlobalMaterial' = tempoAGlobalMaterial <$ reserved "tempo"

tempoAGlobalMaterial :: Double -> GlobalMaterial -> GlobalMaterial
tempoAGlobalMaterial newFreq gm = gm {tempoForStandalone = Tempo { freq =
   toRational $ newFreq, time=mytime 0, Data.Tempo.count=0}}
--
parseSetChordProg :: H (GlobalMaterial -> GlobalMaterial)
parseSetChordProg = parseSetChordProgWMetre
               <|> parseSetChordProgMetreAuto

parseSetChordProgWMetre :: H (GlobalMaterial -> GlobalMaterial)
parseSetChordProgWMetre = parseSetChordProgWMetre' <*> chordList

parseSetChordProgWMetre' :: H ([Chord] -> GlobalMaterial -> GlobalMaterial)
parseSetChordProgWMetre' = parseSetChordProgWMetre'' <*> rationalOrInteger

parseSetChordProgWMetre'' :: H (Metre -> [Chord] -> GlobalMaterial ->
   GlobalMaterial)
parseSetChordProgWMetre'' = setChordProg <$ (reserved "armonia" <|> reserved
    "armona" <|> reserved "acordes")

setChordProg :: Metre -> [Chord] -> GlobalMaterial -> GlobalMaterial
setChordProg metre hs gm = gm { harmony = castProgression metre (compas gm)
   (multiplicarCompasInicioYFinal (compas gm) hs)}

-- parseSetChordProgMetreAuto :: H (GlobalMaterial -> GlobalMaterial)
```

```
-- parseSetChordProgMetreAuto = (reserved "armonia" >> return
   setChordProgMetreAuto) <*> chordListMetreAuto

parseSetChordProgMetreAuto :: H (GlobalMaterial -> GlobalMaterial)
parseSetChordProgMetreAuto = parseSetChordProgMetreAuto' <*>
   listaDePitchOrPitchType -- chordListMetreAuto

parseSetChordProgMetreAuto' :: H ([(Pitch, ChordType)] -> GlobalMaterial ->
   GlobalMaterial)
parseSetChordProgMetreAuto' = setChordProgMetreAuto <$ (reserved "armonia"
   <|> reserved "armona" <|> reserved "acordes")

setChordProgMetreAuto :: [(Pitch, ChordType)] -> GlobalMaterial ->
   GlobalMaterial
setChordProgMetreAuto hs gm = gm { harmony = castProgressionMetreAuto (
   compas gm) (listaDePitchOPitchWType hs)}

-- listaDePitchOPitchWType :: Double -> [(Pitch, ChordType)] -> [Chord]


chordList :: H [Chord]
chordList = list $ (chordParser <|> chordParserMajAuto)

chordParser :: H Chord
chordParser = chordParser' <*> rationalOrInteger

chordParser' :: H (Rational -> Chord)
chordParser' = chordParser'' <*> rationalOrInteger

chordParser'' :: H (Rational -> Rational -> Chord)
chordParser'' = chordParser''' <*> chordTypeParser

chordParser''' :: H (ChordType -> Rational -> Rational -> Chord)
chordParser''' = do
  p <- pitchParser
  return $ \t s e -> castHarmony p t s e

--
chordParserMajAuto :: H Chord
chordParserMajAuto = chordParserMajAuto' <*> rationalOrInteger

chordParserMajAuto' :: H (Rational -> Chord)
chordParserMajAuto' = chordParserMajAuto'' <*> rationalOrInteger

chordParserMajAuto'' :: H (Rational -> Rational -> Chord)
```

```
chordParserMajAuto'' = do
  p <- pitchParser
  return $ \s e -> castHarmonyMajAuto p s e


--parses a list of chords with default dur of 1c, e.g. armonia [c maj, d
   maj]
chordListMetreAuto :: H [Chord]
chordListMetreAuto = parsePitchtochord
                <|> chordListMetreAuto'

--
--
parsePitchtochord :: H [Chord]
parsePitchtochord = do
  p <- list pitchParser
  return $ pitchtochord p

pitchtochord :: [Pitch] -> [Chord]
pitchtochord ps = do
  let startandend = fmap (\s -> (toRational s, toRational s+1)) [0 .. (
     length ps)]
  let zipXSwithStartEnd = zip ps startandend --[(x,())]
  fmap (\(p, (s,e)) -> Chord p major (s,e)) zipXSwithStartEnd

pitchAPicthYTipo :: Pitch -> (Pitch, ChordType)
pitchAPicthYTipo p = (p, major)

--
chordListMetreAuto' :: H [Chord]
chordListMetreAuto' = do
  x <- listofpitchandchord
  return $ pitchandtypetochord x

listofpitchandchord :: H [(Pitch, ChordType)]
listofpitchandchord = list parsepitchandtypetotuple

parsepitchandtypetotuple :: H (Pitch, ChordType)
parsepitchandtypetotuple = parsepitchandtypetotuple' <*> chordTypeParser

parsepitchandtypetotuple' :: H (ChordType -> (Pitch, ChordType))
parsepitchandtypetotuple' = do
  p <- pitchParser
  return $ \t -> pitchandtypetotuple p t
```

```
pitchandtypetotuple :: Pitch -> ChordType -> (Pitch, ChordType)
pitchandtypetotuple p t = (p,t)

-- start here
pitchandtypetochord :: [(Pitch, ChordType)] -> [Chord]
pitchandtypetochord xs = do
  let startandend = fmap (\s -> (toRational s, toRational s+1)) [0 .. (
     length xs)]
  let zipXSwithStartEnd = zip xs startandend --[((),())]
  fmap (\((p,t), (s,e)) -> Chord p t (s,e)) zipXSwithStartEnd

--
-- parsePitchandtypetoPitchType :: H ()

listaDePitchOrPitchType :: H [(Pitch, ChordType)]
listaDePitchOrPitchType = list (parsePitchToPitchType <|>
   parsePitchandtypetoPitchType)

parsePitchToPitchType :: H (Pitch, ChordType)
parsePitchToPitchType = do
  p <- pitchParser
  return $ pitchToPitchType p

pitchToPitchType :: Pitch -> (Pitch, ChordType)
pitchToPitchType p = (p, major)


parsePitchandtypetoPitchType :: H (Pitch, ChordType)
parsePitchandtypetoPitchType = parsePitchandtypetoPitchType' <*>
   chordTypeParser


parsePitchandtypetoPitchType' :: H (ChordType -> (Pitch, ChordType))
parsePitchandtypetoPitchType' = do
  p <- pitchParser
  return $ \t -> pitchandtypetoPitchType p t

pitchandtypetoPitchType :: Pitch -> ChordType -> (Pitch, ChordType)
pitchandtypetoPitchType p t = (p, t)

--
listaDePitchOPitchWType :: [(Pitch, ChordType)] -> [Chord]
listaDePitchOPitchWType xs = do
  let startandend = fmap (\s -> (toRational s, toRational s+1)) [0 .. (
     length xs)]
```

```
  let zipXSwithStartEnd = zip xs startandend --[((),())]
  fmap (\((p,t), (s,e)) -> Chord p t (s, e)) zipXSwithStartEnd

--
multiplicarCompasInicioYFinal :: Double -> [Chord] -> [Chord]
multiplicarCompasInicioYFinal compas hs = fmap (
   multiplicarCompasInicioYFinal' compas) hs

multiplicarCompasInicioYFinal' :: Double -> Chord -> Chord
multiplicarCompasInicioYFinal' compas (Chord p t (s, e)) = Chord p t (s *
   toRational compas, e * toRational compas)

-- armonia 1 [C maj 0 1]
castProgression :: Rational -> Double -> [Chord] -> Progression
castProgression metre compas cs = Progression (metre * toRational compas) cs

-- armonia 1 [C maj 0 1]
castProgressionMetreAuto :: Double -> [Chord] -> Progression
castProgressionMetreAuto compas cs = do
  let metre = realToFrac $ length cs
  let cs' = multiplicarCompasInicioYFinal compas cs
  Progression (metre * toRational compas) cs'

castHarmony :: Pitch -> ChordType -> Rational -> Rational -> Chord
castHarmony p t s e = Chord p t (s, e)

castHarmonyMajAuto :: Pitch -> Rational -> Rational -> Chord
castHarmonyMajAuto p s e = Chord p major (s, e)


pitchParser :: H Pitch
pitchParser =
              c <$ (reserved "c" <|> reserved "di")
          <|> cs <$ (reserved "c'" <|> reserved "do'")
          <|> cs <$ (reserved "db" <|> reserved "reb")
          <|> d <$ (reserved "d" <|> reserved "re")
          <|> ds <$ (reserved "d'" <|> reserved "re'")
          <|> ds <$ (reserved "eb" <|> reserved "mib")
          <|> e <$ (reserved "e" <|> reserved "mi")
          <|> f <$ (reserved "f" <|> reserved "fa")
          <|> fs <$ (reserved "f'" <|> reserved "fa'")
          <|> fs <$ (reserved "gb" <|> reserved "solb")
          <|> g <$ (reserved "g" <|> reserved "sol")
          <|> gs <$ (reserved "g'" <|> reserved "sol'")
          <|> gs <$ (reserved "ab" <|> reserved "lab")
```

```
          <|> a <$ (reserved "a" <|> reserved "la")
          <|> as <$ (reserved "a'" <|> reserved "la'")
          <|> as <$ (reserved "bb" <|> reserved "sib")
          <|> b <$ (reserved "b" <|> reserved "si")

chordTypeParser :: H ChordType
chordTypeParser =
               major <$ (reserved "maj" <|> reserved "M")
            <|> minor <$ (reserved "min" <|> reserved "m")
            <|> major7 <$ (reserved "maj7" <|> reserved "M7")
            <|> minor7 <$ (reserved "min7" <|> reserved "m7")
            <|> dom <$ reserved "dom"
            <|> fifths <$ reserved "quintas"
            <|> sus4 <$ reserved "sus4"
            <|> sus2 <$ reserved "sus2"
            <|> aug <$ reserved "aug"
            <|> dim <$ reserved "dim"
            <|> dim7 <$ reserved "dim7"
            <|> semidim <$ reserved "sdim"

parseLayer :: H Layer
parseLayer = inst
         <|> transformadoresDeLayer

transformadoresDeLayer :: H Layer
transformadoresDeLayer = parseSeleccionarEstilo
                   <|> parseSeleccionarSample
                   <|> parseSeleccionarSamples
                   <|> parseTonicaYquinta
                   <|> parseTonicaYquinta2
                   <|> parseTonicaQoctava
                   <|> parseTonicaQtercera
                   <|> parseCambiarNota
                   <|> parseCambiarNotas
                   <|> parseCambiarRitmo
                   <|> parseCambiarRitmos
                   <|> parseCambiarRitmoAuto
                   <|> parseCambiarRitmosAuto
                   <|> parseCambiarIntervalo
                   <|> parseCambiarIntervalos
                   <|> parseCambiarIntervaloConOctava
                   <|> parseCambiarIntervalosConOctava
                   <|> parseCambiarIntervaloDouble
                   <|> parseCambiarIntervaloDoubleConOctava
                   <|> parseCambiarIntervalosDouble
```

```
                    <|> parseCambiarIntervalosDoubleConOctava
                    <|> parsePreset
                    <|> parseAlternar
                    <|> parseCambiarGain
                    <|> parseCambiarPaneo
                    <|> parsePunteo
                    <|> parsePunteos
                    <|> parseTumbao
                    <|> parseaTumbaoBajoVoicingSel
                    <|> parseaTumbaoBajoVoicingYRitmoSel
                    <|> parseaTumbaoBajoVoicingsYRitmoSel
                    <|> parseTumbaoCongasGolpesSel
                    <|> parseTumbaoCongasGolpesYRitmoSel
                    -- <|> parseTumbaoCongasListaDeGolpesSel
                    <|> parseTumbaoCongasListaDeGolpesYRitmoSel
                    <|> parseacompanamiento
                    <|> parseacompanamientos
                    <|> parseAcompanamientoConVoicingSel
                    <|> parseAcompanamientosConVoicingSel
--
inst :: H Layer
inst =
       teclado <$ reserved "teclado"
   <|> bajo <$ reserved "bajo"
   <|> guira <$ (reserved "guira" <|> reserved "guiro" <|> reserved "giro"
      <|> reserved "gira")
   <|> contras <$ reserved "contratiempos"
   <|> cuerda <$ reserved "cuerda"
   <|> acordeon <$ (reserved "acordeon" <|> reserved "acorden")
   <|> zampoa <$ (reserved "zampoa" <|> reserved "flauta")
   <|> tarola <$ reserved "tarola"
   <|> efecto <$ reserved "efecto"
   <|> altavoz <$ reserved "altavoz"
   <|> clave <$ reserved "clave"
   <|> jamblock <$ (reserved "jamblock" <|> reserved "jam" <|> reserved "
      block")
   <|> congas <$ reserved "congas"
   <|> extras <$ reserved "extras"


estilo :: H S.Style
estilo = defaultStyle <$ reserved "def"
     <|> cumbia <$ reserved "cumbia"

-- a function to change the style of the layer
```

```
parseSeleccionarEstilo :: H Layer
parseSeleccionarEstilo = parseSeleccionarEstilo' <*> parseLayer

parseSeleccionarEstilo' :: H (Layer -> Layer)
parseSeleccionarEstilo' = do
  e <- estilo
  return $ \c -> seleccionarEstilo e c

seleccionarEstilo :: S.Style -> Layer -> Layer
seleccionarEstilo e c = c {style = e}

-- a function for selecting a different sample n, e.g. (sample [2] cumbia)
    teclado
parseSeleccionarSamples :: H Layer
parseSeleccionarSamples = parseSeleccionarSamples' <*> parseLayer

parseSeleccionarSamples' :: H (Layer -> Layer)
parseSeleccionarSamples' = parseSeleccionarSamples'' <*> parseNPattern1

parseSeleccionarSamples'' :: H (NPattern -> Layer -> Layer)
parseSeleccionarSamples'' = seleccionarSamples <$ reserved "sample"

seleccionarSamples :: NPattern -> Layer -> Layer
seleccionarSamples is c = c {style = nuevoE}
  where nuevoE = (style c) {
                        acordeonSampleNPattern0 = is,
                        zampoaSampleNPattern0 = is,
                        cuerdaSampleNPattern0 = is,
                        tecladoSampleNPattern0 = is,
                        bassSampleNPattern0 = is,
                        guiraSampleNPattern0 = is,
                        contrasSampleNPattern0 = is,
                        tarolaSampleNPattern0 = is,
                        efectoSampleNPattern0 = is,
                        altavozSampleNPattern0 = is,
                        extrasSampleNPattern0 = is,
                        jamblockSampleNPattern0 = is,
                        claveSampleNPattern0 = is
                        -- congasSampleNPattern0 = is
                       }

-- a function to select a new sample from the folder
parseSeleccionarSample :: H Layer
parseSeleccionarSample = parseSeleccionarSample' <*> parseLayer
```

```
parseSeleccionarSample' :: H (Layer -> Layer)
parseSeleccionarSample' = parseSeleccionarSample'' <*> int

parseSeleccionarSample'' :: H (Int -> Layer -> Layer)
parseSeleccionarSample'' = seleccionarSample <$ reserved "sample"

seleccionarSample :: Int -> Layer -> Layer
seleccionarSample index c = c {style = nuevoE}
   where nuevoE = (style c) {
                             cuerdaSampleNPattern0 = NPattern1 [index],
                             acordeonSampleNPattern0 = NPattern1 [index],
                             zampoaSampleNPattern0 = NPattern1 [index],
                             tecladoSampleNPattern0 = NPattern1 [index],
                             bassSampleNPattern0 = NPattern1 [index],
                             guiraSampleNPattern0 = NPattern1 [index],
                             contrasSampleNPattern0 = NPattern1 [index],
                             tarolaSampleNPattern0 = NPattern1 [index],
                             efectoSampleNPattern0 = NPattern1 [index],
                             altavozSampleNPattern0 = NPattern1 [index],
                             extrasSampleNPattern0 = NPattern1 [index],
                             claveSampleNPattern0 = NPattern1 [index],
                             jamblockSampleNPattern0 = NPattern1 [index],
                             congasSampleNPattern0 = NPattern1 [index]
                            }

-- TO DO: punteo [mi do] ... Y CON MIDINOTE
-- punteo "1a" 3 $ cumbia teclado;
-- parsePunteoConMidiNote :: H Layer
-- parsePunteoConMidiNote = parsePunteoConMidiNote' <*> parseLayer
--
-- parsePunteoConMidiNote' :: H (Layer -> Layer)
-- parsePunteoConMidiNote' = parsePunteoConMidiNote'' <*>
   rationalOrInteger
--
-- parsePunteoConMidiNote'' :: H (Rational -> Layer -> Layer)
-- parsePunteoConMidiNote'' = parsePunteoConMidiNote''' <*>
   parseUnNAListaDeN
--
-- parsePunteoConMidiNote''' :: H ([N] -> Rational -> Layer -> Layer)
-- parsePunteoConMidiNote''' = punteoConMidiNote <$ (reserved "punteo")
--
-- punteoConMidiNote :: [N] -> Rational -> Layer -> Layer
-- punteoConMidiNote nota ataque c = c {style = nuevoE}
-- where
-- rPat = cambiarRitmo'''' 1 [[ataque]]
```

```
-- nuevoE = (style c) {
-- tecladoRhythmPattern0 = rPat, --
-- tecladoPitchPattern0 = ("midinote", nota), --new ("intervalo", concat
   notes)-- ("acorde", [note])
-- acordeonRhythmPattern0 = rPat,
-- acordeonPitchPattern0 = ("midinote", nota),
--
-- cuerdaRhythmPattern0 = rPat,
-- cuerdaPitchPattern0 = ("midinote", nota),
--
-- extrasRhythmPattern0 = rPat,
-- extrasPitchPattern0 = ("midinote", nota)
-- }

-- tecladoPitchPattern0 = ("intervalo", [("unisono" , 0, 0)]),
-- type PitchPattern = (PitchType, [Note])

octavarPatronDeTono :: PitchPattern -> PitchPattern
octavarPatronDeTono (pitchType, [(relacion, double, octava)]) = do


-- octavar punteo
octavarPunteo :: Layer -> Layer
octavarPunteo c = c {style = nuevoE}
  where
    nuevoE = (style c) {
                  fmap snd tecladoPitchPattern0 --new ("intervalo", concat
                     notes)-- ("acorde", [note])
                  --
                  -- acordeonPitchPattern0 = ("intervalo", nota),
                  --
                  -- zampoaPitchPattern0 = ("intervalo", nota),
                  --
                  -- cuerdaRhythmPattern0 = rPat,
                  -- cuerdaPitchPattern0 = ("intervalo", nota),
                  --
                  -- extrasRhythmPattern0 = rPat,
                  -- extrasPitchPattern0 = ("intervalo", nota)
  }


-- punteo "1a" 3 $ cumbia teclado;
parsePunteo :: H Layer
parsePunteo = parsePunteo' <*> parseLayer
```

```
parsePunteo' :: H (Layer -> Layer)
parsePunteo' = parsePunteo'' <*> rationalOrInteger

parsePunteo'' :: H (Rational -> Layer -> Layer)
parsePunteo'' = parsePunteo''' <*> parseUnStringAListadeNotas --

parsePunteo''' :: H ([Note] -> Rational -> Layer -> Layer)
parsePunteo''' = punteo <$ (reserved "punteo")

punteo :: [Note] -> Rational -> Layer -> Layer
punteo nota ataque c = c {style = nuevoE}
  where
    rPat = cambiarRitmo'''' 1 [[ataque]]
    nuevoE = (style c) {
                  tecladoRhythmPattern0 = rPat, --
                  tecladoPitchPattern0 = ("intervalo", nota), --new ("
                     intervalo", concat notes)-- ("acorde", [note])

                  acordeonRhythmPattern0 = rPat,
                  acordeonPitchPattern0 = ("intervalo", nota),

                  zampoaRhythmPattern0 = rPat,
                  zampoaPitchPattern0 = ("intervalo", nota),

                  cuerdaRhythmPattern0 = rPat,
                  cuerdaPitchPattern0 = ("intervalo", nota),

                  extrasRhythmPattern0 = rPat,
                  extrasPitchPattern0 = ("intervalo", nota)
  }

-- punteo ["f" "5a", "f" "3a" "5a"] [1 3, 1 3 4] $ cumbia acordeon;
parsePunteos :: H Layer
parsePunteos = parsePunteos' <*> parseLayer

parsePunteos' :: H (Layer -> Layer)
parsePunteos' = parsePunteos'' <*> parseListasDeListasDeAtaques --
   rationalList --

parsePunteos'' :: H ( [[Rational]] -> Layer -> Layer)
parsePunteos'' = parsePunteos''' <*> praseListaDeListaStringAListaDeAcordes

parsePunteos''' :: H ([[Note]] -> [[Rational]] -> Layer -> Layer)
parsePunteos''' = listaDepunteos <$ (reserved "punteo")
```

```
listaDepunteos :: [[Note]] -> [[Rational]] -> Layer -> Layer
listaDepunteos notes rs c = c {style = nuevoE}
  where
    metre = toRational $ length rs -- [[Nothing], [1, 2, 3]] = metre 2 -- (
        realToFrac $ floor rs') + 1
    rPat = cambiarRitmo'''' metre rs
    nPat (NPattern1 xs) = NPattern1 $ concat $ replicate (length rPat) xs
    pPat = take (length rPat) $ concat notes -- new
    nuevoE = (style c) {
                          tecladoRhythmPattern0 = rPat, --
                          tecladoSampleNPattern0 = nPat $
                             tecladoSampleNPattern0 (style c), --
                             listaDeStringsANPattern nPat notes,
                          tecladoPitchPattern0 = ("intervalo", pPat), --new
                             ("intervalo", concat notes)-- ("acorde", [note
                             ])

                          acordeonRhythmPattern0 = rPat,
                          acordeonSampleNPattern0 = nPat $
                             acordeonSampleNPattern0 (style c),
                          acordeonPitchPattern0 = ("intervalo", pPat),

                          zampoaRhythmPattern0 = rPat,
                          zampoaSampleNPattern0 = nPat $
                             acordeonSampleNPattern0 (style c),
                          zampoaPitchPattern0 = ("intervalo", pPat),

                          cuerdaRhythmPattern0 = rPat,
                          cuerdaSampleNPattern0 = nPat $
                             cuerdaSampleNPattern0 (style c),
                          cuerdaPitchPattern0 = ("intervalo", pPat),

                          extrasRhythmPattern0 = rPat,
                          extrasSampleNPattern0 = nPat $
                             extrasSampleNPattern0 (style c),
                          extrasPitchPattern0 = ("intervalo", pPat)
                        }


-- tumbao ("f" "3a" "5a") $ cumbia bajo;
parseaTumbaoBajoVoicingSel :: H Layer
parseaTumbaoBajoVoicingSel = parseaTumbaoBajoVoicingSel' <*> parseLayer

parseaTumbaoBajoVoicingSel' :: H (Layer -> Layer)
```

```
parseaTumbaoBajoVoicingSel' = parseaTumbaoBajoVoicingSel'' <*> parseNoteList
    --

parseaTumbaoBajoVoicingSel'' :: H ([Note] -> Layer -> Layer)
parseaTumbaoBajoVoicingSel'' = tumbaoBajoVoicingSel <$ (reserved "tumbao")

tumbaoBajoVoicingSel :: [Note] -> Layer -> Layer
tumbaoBajoVoicingSel notes c = c {style = nuevoE}
  where
    nuevoE = (style c) {
                         bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)],
                             --
                         bassSampleNPattern0 = bassSampleNPattern0 (style c)
                            , -- listaDeStringsANPattern nPat notes,
                         bassPitchPattern0 = ("intervalo", notes)-- ("acorde
                            ", [note])
                       }

-- tumbao ("f" 5a") (1 3) $ cumbia bajo;
parseaTumbaoBajoVoicingYRitmoSel :: H Layer
parseaTumbaoBajoVoicingYRitmoSel = parseaTumbaoBajoVoicingYRitmoSel' <*>
   parseLayer

parseaTumbaoBajoVoicingYRitmoSel' :: H (Layer -> Layer)
parseaTumbaoBajoVoicingYRitmoSel' = parseaTumbaoBajoVoicingYRitmoSel'' <*>
   parseAtaquesAListaDeAtaques -- [] -- rationalList --

parseaTumbaoBajoVoicingYRitmoSel'' :: H ( [Rational] -> Layer -> Layer)
parseaTumbaoBajoVoicingYRitmoSel'' = parseaTumbaoBajoVoicingYRitmoSel''' <*>
    parseNoteList

parseaTumbaoBajoVoicingYRitmoSel''' :: H ([Note] -> [Rational] -> Layer ->
   Layer)
parseaTumbaoBajoVoicingYRitmoSel''' = tumbaoBajoVoicingYRitmoSel <$ (
   reserved "tumbao")

tumbaoBajoVoicingYRitmoSel :: [Note] -> [Rational] -> Layer -> Layer
tumbaoBajoVoicingYRitmoSel notes rs c = c {style = nuevoE}
  where
    metre = 1
    -- indices = [0 .. metre]
    -- rs' = fmap (\n -> if (n == 0) then 0 else (abs $ n - 1)) rs -- [1,
       2, 3, 4] a [0, 1, 2, 3]
    rPat = cambiarRitmo'' metre rs -- fmap (\n -> (metre, (realToFrac n) /4)
       ) rs'-- [(1, (realToFrac n') / 4)]
```

```
    nPat (NPattern1 xs) = NPattern1 $ concat $ replicate (length rPat) xs
    pPat = take (length rPat) notes -- new
    nuevoE = (style c) {
                          bassRhythmPattern0 = rPat, --
                          bassSampleNPattern0 = nPat $ bassSampleNPattern0 (
                             style c), -- listaDeStringsANPattern nPat notes
                             ,
                          bassPitchPattern0 = ("intervalo", pPat)-- new ("
                             intervalo", notes)-- ("acorde", [note])
                        }

-- tumbao ["f" "5a", "f" "3a" "5a"] [1 3, 1 3 4] $ cumbia bajo;
parseaTumbaoBajoVoicingsYRitmoSel :: H Layer
parseaTumbaoBajoVoicingsYRitmoSel = parseaTumbaoBajoVoicingsYRitmoSel' <*>
   parseLayer

parseaTumbaoBajoVoicingsYRitmoSel' :: H (Layer -> Layer)
parseaTumbaoBajoVoicingsYRitmoSel' = parseaTumbaoBajoVoicingsYRitmoSel'' <*>
    parseListasDeListasDeAtaques -- rationalList --

parseaTumbaoBajoVoicingsYRitmoSel'' :: H ( [[Rational]] -> Layer -> Layer)
parseaTumbaoBajoVoicingsYRitmoSel'' = parseaTumbaoBajoVoicingsYRitmoSel'''
   <*> praseListaDeListaStringAListaDeAcordes

parseaTumbaoBajoVoicingsYRitmoSel''' :: H ([[Note]] -> [[Rational]] -> Layer
    -> Layer)
parseaTumbaoBajoVoicingsYRitmoSel''' = tumbaoBajoVoicingsYRitmoSel <$ (
   reserved "tumbao")

tumbaoBajoVoicingsYRitmoSel :: [[Note]] -> [[Rational]] -> Layer -> Layer
tumbaoBajoVoicingsYRitmoSel notes rs c = c {style = nuevoE}
  where
    -- rs' = ((maximum $ concat rs) - 1) / 4
    metre = toRational $ length rs -- [[Nothing], [1, 2, 3]] = metre 2 -- (
       realToFrac $ floor rs') + 1
    rPat = cambiarRitmo'''' metre rs
    nPat (NPattern1 xs) = NPattern1 $ concat $ replicate (length rPat) xs
    pPat = take (length rPat) $ concat notes -- new
    nuevoE = (style c) {
                          bassRhythmPattern0 = rPat, --
                          bassSampleNPattern0 = nPat $ bassSampleNPattern0 (
                             style c), -- listaDeStringsANPattern nPat notes
                             ,
                          bassPitchPattern0 = ("intervalo", pPat) --new ("
                             intervalo", concat notes)-- ("acorde", [note])
```

```
                    }

-- marcha ("p" "t" "p" "a") $ cumbia congas -- accepts only 4 beats
parseTumbaoCongasGolpesSel :: H Layer
parseTumbaoCongasGolpesSel = parseTumbaoCongasGolpesSel' <*> parseLayer

parseTumbaoCongasGolpesSel' :: H (Layer -> Layer)
parseTumbaoCongasGolpesSel' = parseTumbaoCongasGolpesSel'' <*>
   parseNAListaDeN -- ["a" "t"] = ["a", "t"]-- congasN

parseTumbaoCongasGolpesSel'' :: H ([N] -> Layer -> Layer)
parseTumbaoCongasGolpesSel'' = tumbaoCongasGolpesSel <$ (reserved "tumbao"
   <|> reserved "marcha")

tumbaoCongasGolpesSel :: [N] -> Layer -> Layer
tumbaoCongasGolpesSel xs c = c {style = nuevoE}
  where
    -- ns = fmap nSample xs --[nSample x1, nSample x2 ...] = [0, 1, ...]
    -- nPat = fmap (\x -> ("quinto", x)) ns -- [("quinto", 0), ("quinto",
       1) ...]
    rPat = take (length xs) $ [(1, 0), (1, 0.25), (1, 0.5), (1, 0.75)] --
       congasRhythmPattern0 (style c)
    pPat = take (length xs) $ cycle [("mn", 60, 0)] -- snd $
       congasPitchPattern0 (style c)
    nuevoE = (style c) {
      congasRhythmPattern0 = rPat,-- congasRhythmPattern0 (style c), -- [(1,
          0), (1, 0.25), (1, 0.5), (1, 0.75)],
      congasSampleNPattern0 = NPattern2 xs, -- [("quinto", 0), ("quinto", 1)
          ...]
      congasPitchPattern0 = ("midinote", pPat) -- ("midinote", take 4 $
         cycle [("mn", 60, 0)])
    }


--tumbao ("p" "t" "p" "a") (1 2 3 4 4.5) $ cumbia congas
parseTumbaoCongasGolpesYRitmoSel :: H Layer
parseTumbaoCongasGolpesYRitmoSel = parseTumbaoCongasGolpesYRitmoSel' <*>
   parseLayer

parseTumbaoCongasGolpesYRitmoSel' :: H (Layer -> Layer)
parseTumbaoCongasGolpesYRitmoSel' = parseTumbaoCongasGolpesYRitmoSel'' <*>
   parseAtaquesAListaDeAtaques-- parseListasDeListasDeAtaques --
   rationalList -- ["a" "t"] = ["a", "t"]-- congasN

parseTumbaoCongasGolpesYRitmoSel'' :: H ([Rational] -> Layer -> Layer)
```

```
parseTumbaoCongasGolpesYRitmoSel'' = parseTumbaoCongasGolpesYRitmoSel''' <*>
    parseNAListaDeN -- ["a" "t"] = ["a", "t"]-- congasN

parseTumbaoCongasGolpesYRitmoSel''' :: H ([N] -> [Rational] -> Layer ->
   Layer)
parseTumbaoCongasGolpesYRitmoSel''' = tumbaoCongasGolpesYRitmoSel <$ (
   reserved "tumbao" <|> reserved "marcha")

tumbaoCongasGolpesYRitmoSel :: [N] -> [Rational] -> Layer -> Layer
tumbaoCongasGolpesYRitmoSel xs rs c = c {style = nuevoE}
  where
    metre = 1
    rPat = cambiarRitmo'' metre rs
    nPat = take (length rPat) xs
    pPat = take (length rPat) $ cycle [("mn", 60, 0)]-- snd $
       congasPitchPattern0 (style c)
    nuevoE = (style c) {
      congasRhythmPattern0 = rPat, -- [(1, 0), (1, 0.25), (1, 0.5), (1,
         0.75)],
      congasSampleNPattern0 = NPattern2 nPat, -- [("quinto", 0), ("quinto",
         1) ...]
      congasPitchPattern0 = ("midinote", pPat) -- ("midinote", take 4 $
         cycle [("mn", 60, 0)])
    }

-- tumbao ["p" "s" "p" (q "a"), "p" "s" "p" (t "a")] $ cumbia congas; --
   pendiente
parseTumbaoCongasListaDeGolpesSel :: H Layer
parseTumbaoCongasListaDeGolpesSel = parseTumbaoCongasListaDeGolpesSel' <*>
   parseLayer

parseTumbaoCongasListaDeGolpesSel' :: H (Layer -> Layer)
parseTumbaoCongasListaDeGolpesSel' = parseTumbaoCongasListaDeGolpesSel'' <*>
    parseListaDeNAListaDeListaDeN -- ["a" "t"] = ["a", "t"]-- congasN

parseTumbaoCongasListaDeGolpesSel'' :: H ([[N]] -> Layer -> Layer)
parseTumbaoCongasListaDeGolpesSel'' = tumbaoCongasListaDeGolpesSel <$ (
   reserved "tumbao" <|> reserved "marcha")

-- ahora es tumbao ("p" "t" "p" "a") $ cumbia congas pero debe ser tumbao
   ("p" "t" "p" "a") $ cumbia congas;
tumbaoCongasListaDeGolpesSel :: [[N]] -> Layer -> Layer
tumbaoCongasListaDeGolpesSel xs c = c {style = nuevoE}
  where
    -- metre = 1 * (length xs) -- eg. 1 * 2 => [..., ...]
```

```
    -- rs' = fmap (\(m, n) -> if (n == 0) then 0 else (abs $ n - 1)) rs --
       [1, 2, 3, 4] a [0, 1, 2, 3]
    -- rPat = fmap (\(m, n) -> (metre, (realToFrac n) /4)) rs'-- [(1, (
       realToFrac n') / 4)]

    nuevoE = (style c) {
      congasRhythmPattern0 = congasRhythmPattern0 (style c), -- [(1, 0), (1,
          0.25), (1, 0.5), (1, 0.75)],
      congasSampleNPattern0 = NPattern2 $ concat xs, -- [("quinto", 0), ("
         quinto", 1) ...]
      congasPitchPattern0 = congasPitchPattern0 (style c) -- ("midinote",
         take 4 $ cycle [("mn", 60, 0)])
    }


-- marcha ["p" "t" "p" (q "a"), "p" "t" "p" (t "a") (t "a")] [1 2 3 4, 1 2
    3 4 4.5] $ cumbia congas;
parseTumbaoCongasListaDeGolpesYRitmoSel :: H Layer
parseTumbaoCongasListaDeGolpesYRitmoSel =
   parseTumbaoCongasListaDeGolpesYRitmoSel' <*> parseLayer

parseTumbaoCongasListaDeGolpesYRitmoSel' :: H (Layer -> Layer)
parseTumbaoCongasListaDeGolpesYRitmoSel' =
   parseTumbaoCongasListaDeGolpesYRitmoSel'' <*>
   parseListasDeListasDeAtaques -- rationalList

parseTumbaoCongasListaDeGolpesYRitmoSel'' :: H ([[Rational]] -> Layer ->
   Layer)
parseTumbaoCongasListaDeGolpesYRitmoSel'' =
   parseTumbaoCongasListaDeGolpesYRitmoSel''' <*>
   parseListaDeNAListaDeListaDeN

parseTumbaoCongasListaDeGolpesYRitmoSel''' :: H ([[N]] -> [[Rational]] ->
   Layer -> Layer)
parseTumbaoCongasListaDeGolpesYRitmoSel''' =
   tumbaoCongasListaDeGolpesYRitmoSel <$ (reserved "tumbao" <|> reserved "
   marcha")

tumbaoCongasListaDeGolpesYRitmoSel :: [[N]] -> [[Rational]] -> Layer ->
   Layer
tumbaoCongasListaDeGolpesYRitmoSel xs rs c = c {style = nuevoE}
  where
    metre = toRational $ length rs -- [[Nothing], [1, 2, 3]] = metre 2 -- (
       realToFrac $ floor rs') + 1
    rPat = cambiarRitmo'''' metre rs
```

```
    nPat = take (length rPat) $ concat xs
    pPat = take (length rPat) $ cycle [("mn", 60, 0)] -- snd $
       congasPitchPattern0 (style c)
    nuevoE = (style c) {
      congasRhythmPattern0 = rPat, -- [(1, 0), (1, 0.25), (1, 0.5), (1,
         0.75)],
      congasSampleNPattern0 = NPattern2 nPat, -- [("quinto", 0), ("quinto",
         1) ...]
      congasPitchPattern0 = ("midinote", pPat) -- ("midinote", take 4 $
         cycle [("mn", 60, 0)])
    }

    -- congasRhythmPattern0 = [(1, 0), (1, 0.25), (1, 0.5), (1, 0.75)],
    -- -- congasSampleNPattern0 = [1, 2, 1, 2],
    -- congasSampleNPattern0 = NPattern2 [("quinto", 0), ("quinto", 1), ("
       quinto", 0), ("quinto", 1)],
    -- congasPitchPattern0 = ("midinote", take 4 $ cycle [("mn", 60, 0)]),
    -- congasPanPattern0 = 0.5,
    -- congasGainPattern0 = 1,
parseUnNAuto :: H (Maybe N)
parseUnNAuto = do
  s <- parseLiteratOrString
  return $ unNAuto s

unNAuto :: String -> Maybe N
unNAuto "p" = Just ("quinto", 0)
unNAuto "t" = Just ("quinto", 1)
unNAuto "a" = Just ("quinto", 2)
unNAuto _ = Nothing

-- parseUnN :: H (Maybe N)
-- parseUnN = parseUnN' <*> parseLiteratOrString
--
-- parseUnN' :: H (String -> Maybe N)
-- parseUnN' = do
-- s1 <- parseLiteratOrString
-- return $ \s2 -> unN s2 s2

parseQuinto :: H (Maybe N)
parseQuinto = parseQuinto' <*> parseLiteratOrString

parseQuinto' :: H (String -> Maybe N)
parseQuinto' = (unN "quinto") <$ reserved "q"

parseTumba :: H (Maybe N)
```

```
parseTumba = parseTumba' <*> parseLiteratOrString

parseTumba' :: H (String -> Maybe N)
parseTumba' = (unN "tumba") <$ reserved "t"

unN :: String -> String -> Maybe N
unN f "p" = Just (f, 0)
unN f "t" = Just (f, 1)
unN f "a" = Just (f, 2)
unN _ _ = Nothing

parseLiteratOrString :: H String
parseLiteratOrString = string
                  <|> identifier

parseCongasN :: H (Maybe N) -- [("quinto", 0), ("quinto", 1) ...]
parseCongasN = parseUnNAuto
           <|> parseQuinto --parseUnN
           <|> parseTumba

parseListaDeNAListaDeListaDeN :: H [[N]]
parseListaDeNAListaDeListaDeN = list parseNAListaDeN


parseNAListaDeN :: H [N]
parseNAListaDeN = parseUnNAListaDeN
              <|> parseDosNAListaDeN
              <|> parseTresNAListaDeN
              <|> parseCuatroNAListaDeN
              <|> parseCincoNAListaDeN
              <|> parseSeisNAListaDeN
              <|> parseSieteNAListaDeN
              <|> parseOchoNAListaDeN
              <|> parseNueveNAListaDeN
              <|> parseDiezNAListaDeN
              <|> parseOnceNAListaDeN
              <|> parseDoceNAListaDeN
              <|> parseTreceNAListaDeN
              <|> parseCatorceNAListaDeN
              <|> parseQuinceNAListaDeN
              <|> parseDieciseisNAListaDeN

-- ("p" "t" "p" (t "a") ...)
parseDieciseisNAListaDeN :: H [N]
parseDieciseisNAListaDeN = parseDieciseisNAListaDeN' <*> parseCongasN
```

```
parseDieciseisNAListaDeN' :: H (Maybe N -> [N])
parseDieciseisNAListaDeN' = parseDieciseisNAListaDeN'' <*> parseCongasN

parseDieciseisNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN'' = parseDieciseisNAListaDeN''' <*> parseCongasN

parseDieciseisNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN''' = parseDieciseisNAListaDeN'''' <*> parseCongasN

parseDieciseisNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> [N])
parseDieciseisNAListaDeN'''' = parseDieciseisNAListaDeN''''' <*>
   parseCongasN

parseDieciseisNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> [N])
parseDieciseisNAListaDeN''''' = parseDieciseisNAListaDeN'''''' <*>
   parseCongasN

parseDieciseisNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
   N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN'''''' = parseDieciseisNAListaDeN''''''' <*>
   parseCongasN

parseDieciseisNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
    N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN''''''' = parseDieciseisNAListaDeN'''''''' <*>
   parseCongasN

parseDieciseisNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN'''''''' = parseDieciseisNAListaDeN''''''''' <*>
   parseCongasN

parseDieciseisNAListaDeN''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN''''''''' = parseDieciseisNAListaDeN'''''''''' <*>
   parseCongasN

parseDieciseisNAListaDeN'''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> [N])
parseDieciseisNAListaDeN'''''''''' = parseDieciseisNAListaDeN''''''''''' <*>
    parseCongasN
```

```
parseDieciseisNAListaDeN'''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> [N])
parseDieciseisNAListaDeN'''''''''''' = parseDieciseisNAListaDeN''''''''''''
   <*> parseCongasN

parseDieciseisNAListaDeN''''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN''''''''''''' = parseDieciseisNAListaDeN'''''''''''''
    <*> parseCongasN

parseDieciseisNAListaDeN'''''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe
   N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN'''''''''''''' = parseDieciseisNAListaDeN
   '''''''''''''' <*> parseCongasN

parseDieciseisNAListaDeN''''''''''''''' :: H (Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN''''''''''''''' = parseDieciseisNAListaDeN
   ''''''''''''''' <*> parseCongasN

parseDieciseisNAListaDeN'''''''''''''''' :: H (Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDieciseisNAListaDeN'''''''''''''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16 ->
     dieciseisNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15
     n16

dieciseisNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N]
dieciseisNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16 =
    catMaybes [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14,
   n15, n16]

-- ("p" "t" "p" (t "a") ...)
parseQuinceNAListaDeN :: H [N]
parseQuinceNAListaDeN = parseQuinceNAListaDeN' <*> parseCongasN
```

```
parseQuinceNAListaDeN' :: H (Maybe N -> [N])
parseQuinceNAListaDeN' = parseQuinceNAListaDeN'' <*> parseCongasN

parseQuinceNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN'' = parseQuinceNAListaDeN''' <*> parseCongasN

parseQuinceNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN''' = parseQuinceNAListaDeN'''' <*> parseCongasN

parseQuinceNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   [N])
parseQuinceNAListaDeN'''' = parseQuinceNAListaDeN''''' <*> parseCongasN

parseQuinceNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> [N])
parseQuinceNAListaDeN''''' = parseQuinceNAListaDeN'''''' <*> parseCongasN

parseQuinceNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN'''''' = parseQuinceNAListaDeN''''''' <*> parseCongasN

parseQuinceNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN''''''' = parseQuinceNAListaDeN'''''''' <*>
   parseCongasN

parseQuinceNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN'''''''' = parseQuinceNAListaDeN''''''''' <*>
   parseCongasN

parseQuinceNAListaDeN''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
   N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [
   N])
parseQuinceNAListaDeN''''''''' = parseQuinceNAListaDeN'''''''''' <*>
   parseCongasN

parseQuinceNAListaDeN'''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> [N])
```

```
parseQuinceNAListaDeN'''''''''''' = parseQuinceNAListaDeN'''''''''''' <*>
   parseCongasN

parseQuinceNAListaDeN'''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN'''''''''''' = parseQuinceNAListaDeN'''''''''''' <*>
   parseCongasN

parseQuinceNAListaDeN'''''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN'''''''''''''' = parseQuinceNAListaDeN'''''''''''''' <*>
    parseCongasN

parseQuinceNAListaDeN'''''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseQuinceNAListaDeN'''''''''''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 ->
     quinceNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15

quinceNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> [N]
quinceNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 =
   catMaybes [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14,
   n15]

-- ("p" "t" "p" (t "a") ...)
parseCatorceNAListaDeN :: H [N]
parseCatorceNAListaDeN = parseCatorceNAListaDeN' <*> parseCongasN

parseCatorceNAListaDeN' :: H (Maybe N -> [N])
parseCatorceNAListaDeN' = parseCatorceNAListaDeN'' <*> parseCongasN

parseCatorceNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN'' = parseCatorceNAListaDeN''' <*> parseCongasN

parseCatorceNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN''' = parseCatorceNAListaDeN'''' <*> parseCongasN

parseCatorceNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    [N])
```

```
parseCatorceNAListaDeN'''' = parseCatorceNAListaDeN''''' <*> parseCongasN

parseCatorceNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> [N])
parseCatorceNAListaDeN''''' = parseCatorceNAListaDeN'''''' <*> parseCongasN

parseCatorceNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN'''''' = parseCatorceNAListaDeN''''''' <*>
   parseCongasN

parseCatorceNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN''''''' = parseCatorceNAListaDeN'''''''' <*>
   parseCongasN

parseCatorceNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
   N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN'''''''' = parseCatorceNAListaDeN''''''''' <*>
   parseCongasN

parseCatorceNAListaDeN''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
    N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN''''''''' = parseCatorceNAListaDeN'''''''''' <*>
   parseCongasN

parseCatorceNAListaDeN'''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> [N])
parseCatorceNAListaDeN'''''''''' = parseCatorceNAListaDeN''''''''''' <*>
   parseCongasN

parseCatorceNAListaDeN''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> [N])
parseCatorceNAListaDeN''''''''''' = parseCatorceNAListaDeN'''''''''''' <*>
   parseCongasN

parseCatorceNAListaDeN'''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN'''''''''''' = parseCatorceNAListaDeN''''''''''''' <*>
    parseCongasN
```

```
parseCatorceNAListaDeN'''''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> [N])
parseCatorceNAListaDeN'''''''''''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 -> catorceNAListaDeN
      n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14

catorceNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> [N]
catorceNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 = catMaybes
    [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14]

-- ("p" "t" "p" (t "a") ...)
parseTreceNAListaDeN :: H [N]
parseTreceNAListaDeN = parseTreceNAListaDeN' <*> parseCongasN

parseTreceNAListaDeN' :: H (Maybe N -> [N])
parseTreceNAListaDeN' = parseTreceNAListaDeN'' <*> parseCongasN

parseTreceNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseTreceNAListaDeN'' = parseTreceNAListaDeN''' <*> parseCongasN

parseTreceNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseTreceNAListaDeN''' = parseTreceNAListaDeN'''' <*> parseCongasN

parseTreceNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [
   N])
parseTreceNAListaDeN'''' = parseTreceNAListaDeN''''' <*> parseCongasN

parseTreceNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseTreceNAListaDeN''''' = parseTreceNAListaDeN'''''' <*> parseCongasN

parseTreceNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> [N])
parseTreceNAListaDeN'''''' = parseTreceNAListaDeN''''''' <*> parseCongasN

parseTreceNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> [N])
parseTreceNAListaDeN''''''' = parseTreceNAListaDeN'''''''' <*> parseCongasN

parseTreceNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
```

```
parseTreceNAListaDeN'''''''' = parseTreceNAListaDeN''''''''' <*>
   parseCongasN

parseTreceNAListaDeN''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseTreceNAListaDeN''''''''' = parseTreceNAListaDeN'''''''''' <*>
   parseCongasN

parseTreceNAListaDeN'''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
   N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseTreceNAListaDeN'''''''''' = parseTreceNAListaDeN''''''''''' <*>
   parseCongasN

parseTreceNAListaDeN''''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
    N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseTreceNAListaDeN''''''''''' = parseTreceNAListaDeN'''''''''''' <*>
   parseCongasN

parseTreceNAListaDeN'''''''''''' :: H (Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> [N])
parseTreceNAListaDeN'''''''''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 -> treceNAListaDeN n1 n2
      n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13

treceNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> [N]
treceNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 = catMaybes [n1,
   n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13]

-- ("p" "t" "p" (t "a") ...)
parseDoceNAListaDeN :: H [N]
parseDoceNAListaDeN = parseDoceNAListaDeN' <*> parseCongasN

parseDoceNAListaDeN' :: H (Maybe N -> [N])
parseDoceNAListaDeN' = parseDoceNAListaDeN'' <*> parseCongasN

parseDoceNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseDoceNAListaDeN'' = parseDoceNAListaDeN''' <*> parseCongasN

parseDoceNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
```

```
parseDoceNAListaDeN''' = parseDoceNAListaDeN'''' <*> parseCongasN

parseDoceNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseDoceNAListaDeN'''' = parseDoceNAListaDeN''''' <*> parseCongasN

parseDoceNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseDoceNAListaDeN''''' = parseDoceNAListaDeN'''''' <*> parseCongasN

parseDoceNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> [N])
parseDoceNAListaDeN'''''' = parseDoceNAListaDeN''''''' <*> parseCongasN

parseDoceNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> Maybe N -> [N])
parseDoceNAListaDeN''''''' = parseDoceNAListaDeN'''''''' <*> parseCongasN

parseDoceNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDoceNAListaDeN'''''''' = parseDoceNAListaDeN''''''''' <*> parseCongasN

parseDoceNAListaDeN''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDoceNAListaDeN''''''''' = parseDoceNAListaDeN'''''''''' <*>
   parseCongasN

parseDoceNAListaDeN'''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseDoceNAListaDeN'''''''''' = parseDoceNAListaDeN''''''''''' <*>
   parseCongasN

parseDoceNAListaDeN''''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe
   N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseDoceNAListaDeN''''''''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 -> doceNAListaDeN n1 n2 n3
     n4 n5 n6 n7 n8 n9 n10 n11 n12

doceNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> [N]
```

```
doceNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 = catMaybes [n1, n2,
   n3, n4, n5, n6, n7, n8, n9, n10, n11, n12]

-- ("p" "t" "p" (t "a") ...)
parseOnceNAListaDeN :: H [N]
parseOnceNAListaDeN = parseOnceNAListaDeN' <*> parseCongasN

parseOnceNAListaDeN' :: H (Maybe N -> [N])
parseOnceNAListaDeN' = parseOnceNAListaDeN'' <*> parseCongasN

parseOnceNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseOnceNAListaDeN'' = parseOnceNAListaDeN''' <*> parseCongasN

parseOnceNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseOnceNAListaDeN''' = parseOnceNAListaDeN'''' <*> parseCongasN

parseOnceNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseOnceNAListaDeN'''' = parseOnceNAListaDeN''''' <*> parseCongasN

parseOnceNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseOnceNAListaDeN''''' = parseOnceNAListaDeN'''''' <*> parseCongasN

parseOnceNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> [N])
parseOnceNAListaDeN'''''' = parseOnceNAListaDeN''''''' <*> parseCongasN

parseOnceNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> Maybe N -> [N])
parseOnceNAListaDeN''''''' = parseOnceNAListaDeN'''''''' <*> parseCongasN

parseOnceNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseOnceNAListaDeN'''''''' = parseOnceNAListaDeN''''''''' <*> parseCongasN

parseOnceNAListaDeN''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseOnceNAListaDeN''''''''' = parseOnceNAListaDeN'''''''''' <*>
   parseCongasN

parseOnceNAListaDeN'''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
    -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseOnceNAListaDeN'''''''''' = do
```

```
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 -> onceNAListaDeN n1 n2 n3 n4 n5
      n6 n7 n8 n9 n10 n11

onceNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N]
onceNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 = catMaybes [n1, n2, n3,
   n4, n5, n6, n7, n8, n9, n10, n11]

-- ("p" "t" "p" (t "a") ...)
parseDiezNAListaDeN :: H [N]
parseDiezNAListaDeN = parseDiezNAListaDeN' <*> parseCongasN

parseDiezNAListaDeN' :: H (Maybe N -> [N])
parseDiezNAListaDeN' = parseDiezNAListaDeN'' <*> parseCongasN

parseDiezNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseDiezNAListaDeN'' = parseDiezNAListaDeN''' <*> parseCongasN

parseDiezNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseDiezNAListaDeN''' = parseDiezNAListaDeN'''' <*> parseCongasN

parseDiezNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseDiezNAListaDeN'''' = parseDiezNAListaDeN''''' <*> parseCongasN

parseDiezNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseDiezNAListaDeN''''' = parseDiezNAListaDeN'''''' <*> parseCongasN

parseDiezNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> [N])
parseDiezNAListaDeN'''''' = parseDiezNAListaDeN''''''' <*> parseCongasN

parseDiezNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> Maybe N -> [N])
parseDiezNAListaDeN''''''' = parseDiezNAListaDeN'''''''' <*> parseCongasN

parseDiezNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDiezNAListaDeN'''''''' = parseDiezNAListaDeN''''''''' <*> parseCongasN

parseDiezNAListaDeN''''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseDiezNAListaDeN''''''''' = do
```

```
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 -> diezNAListaDeN n1 n2 n3 n4 n5 n6
     n7 n8 n9 n10

diezNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N]
diezNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 = catMaybes [n1, n2, n3, n4,
   n5, n6, n7, n8, n9, n10]

-- ("p" "t" "p" (t "a") ...)
parseNueveNAListaDeN :: H [N]
parseNueveNAListaDeN = parseNueveNAListaDeN' <*> parseCongasN

parseNueveNAListaDeN' :: H (Maybe N -> [N])
parseNueveNAListaDeN' = parseNueveNAListaDeN'' <*> parseCongasN

parseNueveNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseNueveNAListaDeN'' = parseNueveNAListaDeN''' <*> parseCongasN

parseNueveNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseNueveNAListaDeN''' = parseNueveNAListaDeN'''' <*> parseCongasN

parseNueveNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [
   N])
parseNueveNAListaDeN'''' = parseNueveNAListaDeN''''' <*> parseCongasN

parseNueveNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseNueveNAListaDeN''''' = parseNueveNAListaDeN'''''' <*> parseCongasN

parseNueveNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> [N])
parseNueveNAListaDeN'''''' = parseNueveNAListaDeN''''''' <*> parseCongasN

parseNueveNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> [N])
parseNueveNAListaDeN''''''' = parseNueveNAListaDeN'''''''' <*> parseCongasN

parseNueveNAListaDeN'''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N
   -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N])
parseNueveNAListaDeN'''''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 -> nueveNAListaDeN n1 n2 n3 n4 n5 n6 n7
     n8 n9
```

```
nueveNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N]
nueveNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 n9 = catMaybes [n1, n2, n3, n4, n5,
   n6, n7, n8, n9]

-- ("p" "t" "p" (t "a") ...)
parseOchoNAListaDeN :: H [N]
parseOchoNAListaDeN = parseOchoNAListaDeN' <*> parseCongasN

parseOchoNAListaDeN' :: H (Maybe N -> [N])
parseOchoNAListaDeN' = parseOchoNAListaDeN'' <*> parseCongasN

parseOchoNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseOchoNAListaDeN'' = parseOchoNAListaDeN''' <*> parseCongasN

parseOchoNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseOchoNAListaDeN''' = parseOchoNAListaDeN'''' <*> parseCongasN

parseOchoNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseOchoNAListaDeN'''' = parseOchoNAListaDeN''''' <*> parseCongasN

parseOchoNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseOchoNAListaDeN''''' = parseOchoNAListaDeN'''''' <*> parseCongasN

parseOchoNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> [N])
parseOchoNAListaDeN'''''' = parseOchoNAListaDeN''''''' <*> parseCongasN

parseOchoNAListaDeN''''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> Maybe N -> [N])
parseOchoNAListaDeN''''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 n8 -> ochoNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8

ochoNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> Maybe N -> [N]
ochoNAListaDeN n1 n2 n3 n4 n5 n6 n7 n8 = catMaybes [n1, n2, n3, n4, n5, n6,
   n7, n8]

-- ("p" "t" "p" (t "a") ...)
parseSieteNAListaDeN :: H [N]
parseSieteNAListaDeN = parseSieteNAListaDeN' <*> parseCongasN
```

```
parseSieteNAListaDeN' :: H (Maybe N -> [N])
parseSieteNAListaDeN' = parseSieteNAListaDeN'' <*> parseCongasN

parseSieteNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseSieteNAListaDeN'' = parseSieteNAListaDeN''' <*> parseCongasN

parseSieteNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseSieteNAListaDeN''' = parseSieteNAListaDeN'''' <*> parseCongasN

parseSieteNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [
   N])
parseSieteNAListaDeN'''' = parseSieteNAListaDeN''''' <*> parseCongasN

parseSieteNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseSieteNAListaDeN''''' = parseSieteNAListaDeN'''''' <*> parseCongasN

parseSieteNAListaDeN'''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
    Maybe N -> Maybe N -> [N])
parseSieteNAListaDeN'''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 n7 -> sieteNAListaDeN n1 n2 n3 n4 n5 n6 n7

sieteNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> Maybe N -> [N]
sieteNAListaDeN n1 n2 n3 n4 n5 n6 n7 = catMaybes [n1, n2, n3, n4, n5, n6, n7
   ]

-- ("p" "t" "p" (t "a") ...)
parseSeisNAListaDeN :: H [N]
parseSeisNAListaDeN = parseSeisNAListaDeN' <*> parseCongasN

parseSeisNAListaDeN' :: H (Maybe N -> [N])
parseSeisNAListaDeN' = parseSeisNAListaDeN'' <*> parseCongasN

parseSeisNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseSeisNAListaDeN'' = parseSeisNAListaDeN''' <*> parseCongasN

parseSeisNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseSeisNAListaDeN''' = parseSeisNAListaDeN'''' <*> parseCongasN

parseSeisNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ])
parseSeisNAListaDeN'''' = parseSeisNAListaDeN''''' <*> parseCongasN
```

```
parseSeisNAListaDeN''''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N])
parseSeisNAListaDeN''''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 n6 -> seisNAListaDeN n1 n2 n3 n4 n5 n6

seisNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N ->
   Maybe N -> [N]
seisNAListaDeN n1 n2 n3 n4 n5 n6 = catMaybes [n1, n2, n3, n4, n5, n6]

-- ("p" "t" "p" (t "a") ...)
parseCincoNAListaDeN :: H [N]
parseCincoNAListaDeN = parseCincoNAListaDeN' <*> parseCongasN

parseCincoNAListaDeN' :: H (Maybe N -> [N])
parseCincoNAListaDeN' = parseCincoNAListaDeN'' <*> parseCongasN

parseCincoNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseCincoNAListaDeN'' = parseCincoNAListaDeN''' <*> parseCongasN

parseCincoNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
parseCincoNAListaDeN''' = parseCincoNAListaDeN'''' <*> parseCongasN

parseCincoNAListaDeN'''' :: H (Maybe N -> Maybe N -> Maybe N -> Maybe N -> [
   N])
parseCincoNAListaDeN'''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 n5 -> cincoNAListaDeN n1 n2 n3 n4 n5

cincoNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N
   ]
cincoNAListaDeN n1 n2 n3 n4 n5 = catMaybes [n1, n2, n3, n4, n5]


-- ("p" "t" "p" $ t "a")
parseCuatroNAListaDeN :: H [N]
parseCuatroNAListaDeN = parseCuatroNAListaDeN' <*> parseCongasN

parseCuatroNAListaDeN' :: H (Maybe N -> [N])
parseCuatroNAListaDeN' = parseCuatroNAListaDeN'' <*> parseCongasN

parseCuatroNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseCuatroNAListaDeN'' = parseCuatroNAListaDeN''' <*> parseCongasN

parseCuatroNAListaDeN''' :: H (Maybe N -> Maybe N -> Maybe N -> [N])
```

```
parseCuatroNAListaDeN''' = do
  n1 <- parseCongasN
  return $ \n2 n3 n4 -> cuatroNAListaDeN n1 n2 n3 n4

cuatroNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> Maybe N -> [N]
cuatroNAListaDeN n1 n2 n3 n4 = catMaybes [n1, n2, n3, n4]

-- ("a" t "t" "a")
parseTresNAListaDeN :: H [N]
parseTresNAListaDeN = parseTresNAListaDeN' <*> parseCongasN

parseTresNAListaDeN' :: H (Maybe N -> [N])
parseTresNAListaDeN' = parseTresNAListaDeN'' <*> parseCongasN

parseTresNAListaDeN'' :: H (Maybe N -> Maybe N -> [N])
parseTresNAListaDeN'' = do
  n1 <- parseCongasN
  return $ \n2 n3 -> tresNAListaDeN n1 n2 n3

tresNAListaDeN :: Maybe N -> Maybe N -> Maybe N -> [N]
tresNAListaDeN n1 n2 n3 = catMaybes [n1, n2, n3]

-- ("a", t "a")
parseDosNAListaDeN :: H [N]
parseDosNAListaDeN = parseDosNAListaDeN' <*> parseCongasN

parseDosNAListaDeN' :: H (Maybe N -> [N])
parseDosNAListaDeN' = do
  n1 <- parseCongasN
  return $ \n2 -> dosNAListaDeN n1 n2

dosNAListaDeN :: Maybe N -> Maybe N -> [N]
dosNAListaDeN n1 n2 = catMaybes [n1, n2]

-- ("a")
parseUnNAListaDeN :: H [N]
parseUnNAListaDeN = do
  n <- parseCongasN
  return $ unNAListaDeN n

unNAListaDeN :: Maybe N -> [N]
unNAListaDeN n = catMaybes [n]

parseTumbao :: H Layer
parseTumbao = parseTumbao' <*> parseLayer
```

```
parseTumbao' :: H (Layer -> Layer) -- (tonicaYquinta cumbia) bajo
parseTumbao' = parseTumbao'' <*> int

parseTumbao'' :: H (Int -> Layer -> Layer) -- (tonicaYquinta cumbia) bajo
parseTumbao'' = tumbao <$ reserved "tumbao"

tumbao :: Int -> Layer -> Layer -- ?

tumbao 0 c = c {style = nuevoE}
  where nuevoE = (style c) {
  bassPitchPattern0 = bassPitchPattern0 (style c),
  bassRhythmPattern0 = bassRhythmPattern0 (style c),

  congasRhythmPattern0 = [(1, 0), (1, 0.25), (1, 0.5), (1, 0.75)],
  congasSampleNPattern0 = NPattern2 [("quinto", 0), ("quinto", 1), ("quinto"
     , 0), ("quinto", 1)],
  congasPitchPattern0 = ("midinote", take 4 $ cycle [("mn", 60, 0)]),

  tecladoPitchPattern0 = tecladoPitchPattern2 (style c),
  tecladoRhythmPattern0 = tecladoRhythmPattern2 (style c)
  -- tecladoSampleNPattern0 = tecladoSampleNPattern2 (style c)
}

-- tonicaQuinta
tumbao 1 c = c {style = nuevoE}
  where nuevoE = (style c) {
                          bassPitchPattern0 = ("intervalo", [(intervalo "
                             unisono" 0), (intervalo "5a" 0)]), -- index
                             from list of pitches i.e. [60, 67]
                          bassRhythmPattern0 = [(1, 0), (1, 0.5)], --i.e. [
                               ],

                          congasRhythmPattern0 = [(1, 0), (1, 0.25), (1, 0.5)
                             , (1, 0.75)],
                          congasSampleNPattern0 = NPattern2 [("quinto", 2), (
                             "quinto", 1), ("quinto", 0), ("quinto", 1)],
                          congasPitchPattern0 = ("midinote", take 4 $ cycle
                             [("mn", 60, 0)]),

                          tecladoPitchPattern0 = tecladoPitchPattern3 (style
                             c),
                          tecladoRhythmPattern0 = tecladoRhythmPattern3 (
                             style c)
```

```
                        -- tecladoSampleNPattern0 = tecladoSampleNPattern3
                            (style c)
                        }
-- tonicaQuinta2
tumbao 2 c = c {style = nuevoE}
  where nuevoE = (style c) {
                        bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)],
                        bassPitchPattern0 = ("intervalo", [intervalo "
                           unisono" 0, intervalo "5a" 0, intervalo "5a"
                           (-1)]), -- index from list of pitches i.e. [60,
                            64, 67]

                        congasRhythmPattern0 = [(1, 0), (1, 0.125), (1,
                           0.25), (1, 0.5), (1, 0.75)],
                        congasSampleNPattern0 = NPattern2 [("quinto", 2), (
                           "quinto", 2), ("quinto", 1), ("quinto", 0), ("
                           quinto", 1)],
                        congasPitchPattern0 = ("midinote", take 5 $ cycle
                           [("mn", 60, 0)])

                       }
-- tonicaAuintaOctava
tumbao 3 c = c {style = nuevoE}
  where nuevoE = (style c) {
                        bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)],
                        bassPitchPattern0 = ("intervalo", [intervalo "
                           unisono" 0, intervalo "5a" 0, intervalo "8a" 0])
                           , -- index from list of pitches i.e. [60, 64,
                           67]

                        congasRhythmPattern0 = [(1, 0), (1, 0.125), (1,
                           0.25), (1, 0.5), (1, 0.75)],
                        congasSampleNPattern0 = NPattern2 [("quinto", 0), (
                           "quinto", 2), ("quinto", 1), ("quinto", 0), ("
                           quinto", 1)],
                        congasPitchPattern0 = ("midinote", take 5 $ cycle
                           [("mn", 60, 0)])

                       }
-- tonicaQuintaTercera
tumbao 4 c = c {style = nuevoE}
  where nuevoE = (style c) {
                        bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)],
                        bassPitchPattern0 = ("intervalo", [intervalo "
                           unisono" 0, intervalo "5a" 0, intervalo "3a" 0])
```

```
                        , -- index from list of pitches i.e. [60, 64,
                           67]

                        congasRhythmPattern0 = [(1, 0), (1, 0.125), (1,
                           0.25), (1, 0.5), (1, 0.625), (1, 0.75)],
                        congasSampleNPattern0 = NPattern2 [("quinto", 0), (
                           "quinto", 2), ("quinto", 1), ("quinto", 0), ("
                           tumba", 0), ("quinto", 1)],
                        congasPitchPattern0 = ("midinote", take 6 $ cycle
                           [("mn", 60, 0)])
                      }

tumbao _ c = c

-- transforms the preset bass to just fundamental and fifth of the chord
-- e.g (tonicaYquinta cumbia) bajo

parseTonicaYquinta :: H Layer
parseTonicaYquinta = parseTonicaYquinta' <*> parseLayer

parseTonicaYquinta' :: H (Layer -> Layer) -- (tonicaYquinta cumbia) bajo
parseTonicaYquinta' = tonicaYquinta <$ reserved "tonicayquinta"

-- una funcin que devuelve a tonica y la quinta del bajo
tonicaYquinta :: Layer -> Layer -- ?
tonicaYquinta c = c {style = nuevoE}
  where nuevoE = (style c) {
                        bassPitchPattern0 = ("intervalo", [(intervalo "
                           unisono" 0), (intervalo "5a" 0)]), -- index
                           from list of pitches i.e. [60, 67]
                        bassRhythmPattern0 = [(1, 0), (1, 0.5)] --i.e. [
                            ],
                        }

-- Arriba, el bajo toca la tnica, la quinta y la quinta una octava ms alta
   .
parseTonicaYquinta2 :: H Layer
parseTonicaYquinta2 = parseTonicaYquinta2' <*> parseLayer

parseTonicaYquinta2' :: H (Layer -> Layer)
parseTonicaYquinta2' = tonicaYquinta2 <$ reserved "tonicayquinta2"

tonicaYquinta2 :: Layer -> Layer
tonicaYquinta2 c = c {style = nuevoE}
  where nuevoE = (style c) {
```

```
                        bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)],
                        bassPitchPattern0 = ("intervalo", [intervalo "
                            unisono" 0, intervalo "5a" 0, intervalo "5a"
                            (-1)]) -- index from list of pitches i.e. [60,
                            64, 67]
                      }

--tonicaQtonica $ cumbia bajo, el bajo toca la tnica, la quinta y la
    octava alta de la tnica.
parseTonicaQoctava :: H Layer
parseTonicaQoctava = parseTonicaQoctava' <*> parseLayer

parseTonicaQoctava' :: H (Layer -> Layer)
parseTonicaQoctava' = tonicaQoctava <$ reserved "tonicaQoctava"

tonicaQoctava :: Layer -> Layer
tonicaQoctava c = c {style = nuevoE}
  where nuevoE = (style c) {
                        bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)],
                        bassPitchPattern0 = ("intervalo", [intervalo "
                            unisono" 0, intervalo "5a" 0, intervalo "8a" 0])
                             -- index from list of pitches i.e. [60, 64,
                            67]
                      }

-- tonicaQtercera $ cumbia bajo, el bajo toca la tnica, la quinta y la
    tercer del acorde.
parseTonicaQtercera :: H Layer
parseTonicaQtercera = parseTonicaQtercera' <*> parseLayer

parseTonicaQtercera' :: H (Layer -> Layer)
parseTonicaQtercera' = tonicaQtercera <$ reserved "tonicaQtercera"

tonicaQtercera :: Layer -> Layer
tonicaQtercera c = c {style = nuevoE}
  where nuevoE = (style c) {
                        bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)],
                        bassPitchPattern0 = ("intervalo", [intervalo "
                            unisono" 0, intervalo "5a" 0, intervalo "3a" 0])
                             -- index from list of pitches i.e. [60, 64,
                            67]
                      }
--

-- a function for changing the preset pitch pattern provided by the style
```

```
parseCambiarNotas :: H Layer
parseCambiarNotas = parseCambiarNotas' <*> parseLayer

parseCambiarNotas' :: H (Layer -> Layer)
parseCambiarNotas' = parseCambiarNotas'' <*> doubleList

parseCambiarNotas'' :: H ([Double] -> Layer -> Layer)
parseCambiarNotas'' = cambiarNotas <$ reserved "nota"

cambiarNotas :: [Double] -> Layer -> Layer
cambiarNotas ps c = c {style = nuevoE}
  where nuevoE = (style c) {
                          cuerdaPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          acordeonPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          zampoaPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          tecladoPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          bassPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          efectoPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          altavozPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          guiraPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          tarolaPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          contrasPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          extrasPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          clavePitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          jamblockPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps),
                          congasPitchPattern0 = ("midinote",
                             listDeNotasConRelacion "mn" ps)
                         }

listDeNotasConRelacion :: Relacion -> [Double] -> [(Relacion, Double, Octava
   )]
listDeNotasConRelacion r ns = fmap (\n -> (r, n, 0)) ns
```

```
-- cambia una sola nota
parseCambiarNota :: H Layer
parseCambiarNota = parseCambiarNota' <*> parseLayer

parseCambiarNota' :: H (Layer -> Layer)
parseCambiarNota' = parseCambiarNota'' <*> double

parseCambiarNota'' :: H (Double -> Layer -> Layer)
parseCambiarNota'' = cambiarNota <$ reserved "nota"

cambiarNota :: Double -> Layer -> Layer
cambiarNota ps c = c {style = nuevoE}
  where nuevoE = (style c) {
                          cuerdaPitchPattern0 = ("midinote", [("mn", ps, 0)])
                             ,
                          acordeonPitchPattern0 = ("midinote", [("mn", ps, 0)
                             ]),
                          zampoaPitchPattern0 = ("midinote", [("mn", ps, 0)])
                             ,
                          tecladoPitchPattern0 = ("midinote", [("mn", ps, 0)
                             ]),
                          bassPitchPattern0= ("midinote", [("mn", ps, 0)]),
                          efectoPitchPattern0 = ("midinote", [("mn", ps, 0)])
                             ,
                          altavozPitchPattern0 = ("midinote", [("mn", ps, 0)
                             ]),
                          tarolaPitchPattern0 = ("midinote", [("mn", ps, 0)])
                             ,
                          contrasPitchPattern0 = ("midinote", [("mn", ps, 0)
                             ]),
                          guiraPitchPattern0 = ("midinote", [("mn", ps, 0)]),
                          extrasPitchPattern0 = ("midinote", [("mn", ps, 0)])
                             ,
                          clavePitchPattern0 = ("midinote", [("mn", ps, 0)]),
                          jamblockPitchPattern0 = ("midinote", [("mn", ps, 0)
                             ]),
                          congasPitchPattern0 = ("midinote", [("mn", ps, 0)])
                           }

-- provee una lista de intervalos con Double y una octava seleccionable, e
   .g. intervalo [0 1, "5a" 2]

parseCambiarIntervalosDoubleConOctava :: H Layer
```

```
parseCambiarIntervalosDoubleConOctava =
   parseCambiarIntervalosDoubleConOctava' <*> parseLayer

parseCambiarIntervalosDoubleConOctava' :: H (Layer -> Layer)
parseCambiarIntervalosDoubleConOctava' =
   parseCambiarIntervalosDoubleConOctava'' <*> listofTupleofdouble

parseCambiarIntervalosDoubleConOctava'' :: H ([(Double, Double)] -> Layer ->
    Layer)
parseCambiarIntervalosDoubleConOctava'' = cambiarIntervalosDoubleConOctava <
   $ reserved "intervalo"

listofTupleofdouble :: H [(Double, Double)]
listofTupleofdouble = list parseTupleofdouble

parseTupleofdouble :: H (Double, Double)
parseTupleofdouble = parseTupleofdouble' <*> double

parseTupleofdouble' :: H (Double -> (Double, Double))
parseTupleofdouble' = do
  i <- double
  return $ \o -> tupleofdouble i o

tupleofdouble :: Double -> Double -> (Double, Double)
tupleofdouble index octava = (index, octava)

cambiarIntervalosDoubleConOctava :: [(Double, Double)] -> Layer -> Layer
cambiarIntervalosDoubleConOctava is c = c {style = nuevoE}
  where
    is' = fmap (\(index, octava) -> intervaloDouble index octava) is
    nuevoE = (style c) {
                           cuerdaPitchPattern0 = ("intervalo", is'),
                           acordeonPitchPattern0 = ("intervalo", is'),
                           zampoaPitchPattern0 = ("intervalo", is'),
                           tecladoPitchPattern0 = ("intervalo", is'),
                           bassPitchPattern0= ("intervalo", is'),
                           efectoPitchPattern0 = ("intervalo", is'),
                           altavozPitchPattern0 = ("intervalo", is'),
                           tarolaPitchPattern0 = ("intervalo", is'),
                           guiraPitchPattern0 = ("intervalo", is'),
                           contrasPitchPattern0 = ("intervalo", is'),
                           extrasPitchPattern0 = ("intervalo", is'),
                           congasPitchPattern0 = ("intervalo", is'),
                           jamblockPitchPattern0 = ("intervalo", is'),
                           clavePitchPattern0 = ("intervalo", is')
```

```
                          }
--
parseCambiarIntervaloDoubleConOctava :: H Layer
parseCambiarIntervaloDoubleConOctava = parseCambiarIntervaloDoubleConOctava'
    <*> parseLayer

parseCambiarIntervaloDoubleConOctava' :: H (Layer -> Layer)
parseCambiarIntervaloDoubleConOctava' = parseCambiarIntervaloDoubleConOctava
   '' <*> double

parseCambiarIntervaloDoubleConOctava'' :: H (Double -> Layer -> Layer)
parseCambiarIntervaloDoubleConOctava'' =
   parseCambiarIntervaloDoubleConOctava''' <*> double

parseCambiarIntervaloDoubleConOctava''' :: H (Double -> Double -> Layer ->
   Layer)
parseCambiarIntervaloDoubleConOctava''' = cambiarIntervaloDoubleConOctava <$
    reserved "intervalo"

cambiarIntervaloDoubleConOctava :: Double -> Double -> Layer -> Layer
cambiarIntervaloDoubleConOctava index octava c = c {style = nuevoE}
  where
    nuevoE = (style c) {
                          cuerdaPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          acordeonPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          zampoaPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          tecladoPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          bassPitchPattern0= ("intervalo", [intervaloDouble
                             index octava]),
                          efectoPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          altavozPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          tarolaPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          guiraPitchPattern0 = ("intervalo", [intervaloDouble
                              index octava]),
                          contrasPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          extrasPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
```

```
                          congasPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          jamblockPitchPattern0 = ("intervalo", [
                             intervaloDouble index octava]),
                          clavePitchPattern0 = ("intervalo", [intervaloDouble
                              index octava])

                          }
--
parseCambiarIntervalosDouble :: H Layer
parseCambiarIntervalosDouble = parseCambiarIntervalosDouble' <*> parseLayer

parseCambiarIntervalosDouble' :: H (Layer -> Layer)
parseCambiarIntervalosDouble' = parseCambiarIntervalosDouble'' <*>
   doubleList

parseCambiarIntervalosDouble'' :: H ([Double] -> Layer -> Layer)
parseCambiarIntervalosDouble'' = cambiarIntervalosDouble <$ reserved "
   intervalo"

cambiarIntervalosDouble :: [Double] -> Layer -> Layer
cambiarIntervalosDouble indices c = c {style = nuevoE}
  where
    indices' = fmap (\index -> intervaloDouble index 0) indices
    nuevoE = (style c) {
                           cuerdaPitchPattern0 = ("intervalo", indices'),
                           acordeonPitchPattern0 = ("intervalo", indices'),
                           zampoaPitchPattern0 = ("intervalo", indices'),
                           tecladoPitchPattern0 = ("intervalo", indices'),
                           bassPitchPattern0= ("intervalo", indices'),
                           efectoPitchPattern0 = ("intervalo", indices'),
                           altavozPitchPattern0 = ("intervalo", indices'),
                           guiraPitchPattern0 = ("intervalo", indices'),
                           contrasPitchPattern0 = ("intervalo", indices'),
                           tarolaPitchPattern0 = ("intervalo", indices'),
                           extrasPitchPattern0 = ("intervalo", indices'),
                           clavePitchPattern0 = ("intervalo", indices'),
                           jamblockPitchPattern0 = ("intervalo", indices'),
                           congasPitchPattern0 = ("intervalo", indices')
                           }

parseCambiarIntervaloDouble :: H Layer
parseCambiarIntervaloDouble = parseCambiarIntervaloDouble' <*> parseLayer

parseCambiarIntervaloDouble' :: H (Layer -> Layer)
```

```
parseCambiarIntervaloDouble' = parseCambiarIntervaloDouble'' <*> double

parseCambiarIntervaloDouble'' :: H (Double -> Layer -> Layer)
parseCambiarIntervaloDouble'' = cambiarIntervaloDouble <$ reserved "
   intervalo"

cambiarIntervaloDouble :: Double -> Layer -> Layer
cambiarIntervaloDouble index c = c {style = nuevoE}
  where
    nuevoE = (style c) {
                          cuerdaPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          acordeonPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          tecladoPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          zampoaPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          bassPitchPattern0= ("intervalo", [intervaloDouble
                             index 0]),
                          efectoPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          tarolaPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          guiraPitchPattern0 = ("intervalo", [intervaloDouble
                              index 0]),
                          contrasPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          altavozPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          extrasPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          congasPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          jamblockPitchPattern0 = ("intervalo", [
                             intervaloDouble index 0]),
                          clavePitchPattern0 = ("intervalo", [intervaloDouble
                              index 0])
                          }
-- provee una lista de intervalos con una octava seleccionable, e.g.
   intervalo ["3a" 1, "5a" 2]

parseCambiarIntervalosConOctava :: H Layer
parseCambiarIntervalosConOctava = parseCambiarIntervalosConOctava' <*>
   parseLayer
```

```
parseCambiarIntervalosConOctava' :: H (Layer -> Layer)
parseCambiarIntervalosConOctava' = parseCambiarIntervalosConOctava'' <*>
   listofTupleofStringDouble

parseCambiarIntervalosConOctava'' :: H ([Index] -> Layer -> Layer)
parseCambiarIntervalosConOctava'' = cambiarIntervalosConOctava <$ reserved "
   intervalo"

data Index = IndexOctavaDef String | IndexConOctava {indice :: String,
   octava :: Double}
-- --
-- listOfIndex :: H [Index]
-- listOfIndex = list parseTupleofStringDouble

listofTupleofStringDouble :: H [Index]
listofTupleofStringDouble = list parseTupleofStringDouble

parseTupleofStringDouble :: H Index
parseTupleofStringDouble = parseTupleofStringDouble' <*> double

parseTupleofStringDouble' :: H (Double -> Index)
parseTupleofStringDouble' = do
  s <- string
  return $ \o -> tupleofStringDouble s o

tupleofStringDouble :: String -> Double -> Index --(String, Double)
tupleofStringDouble index octava = IndexConOctava index octava

cambiarIntervalosConOctava :: [Index] -> Layer -> Layer
cambiarIntervalosConOctava is c = c {style = nuevoE}
  where
    is' = fmap (\x -> intervalo (indice x) (octava x)) is
    nuevoE = (style c) {
                           cuerdaPitchPattern0 = ("intervalo", is'),
                           acordeonPitchPattern0 = ("intervalo", is'),
                           zampoaPitchPattern0 = ("intervalo", is'),
                           tecladoPitchPattern0 = ("intervalo", is'),
                           bassPitchPattern0= ("intervalo", is'),
                           efectoPitchPattern0 = ("intervalo", is'),
                           altavozPitchPattern0 = ("intervalo", is'),
                           guiraPitchPattern0 = ("intervalo", is'),
                           tarolaPitchPattern0 = ("intervalo", is'),
                           contrasPitchPattern0 = ("intervalo", is'),
                           extrasPitchPattern0 = ("intervalo", is'),
```

```
                          congasPitchPattern0 = ("intervalo", is'),
                          jamblockPitchPattern0 = ("intervalo", is'),
                          clavePitchPattern0 = ("intervalo", is')
                          }

-- provee un intervalo con una octava seleccionable, e.g. intervalo "3a" 1
parseCambiarIntervaloConOctava :: H Layer
parseCambiarIntervaloConOctava = parseCambiarIntervaloConOctava' <*>
   parseLayer

parseCambiarIntervaloConOctava' :: H (Layer -> Layer)
parseCambiarIntervaloConOctava' = parseCambiarIntervaloConOctava'' <*>
   double

parseCambiarIntervaloConOctava'' :: H (Double -> Layer -> Layer)
parseCambiarIntervaloConOctava'' = parseCambiarIntervaloConOctava''' <*>
   string

parseCambiarIntervaloConOctava''' :: H (String -> Double -> Layer -> Layer)
parseCambiarIntervaloConOctava''' = cambiarIntervaloConOctava <$ reserved "
   intervalo"

cambiarIntervaloConOctava :: String -> Double -> Layer -> Layer
cambiarIntervaloConOctava index octava c = c {style = nuevoE}
  where
    nuevoE = (style c) {
                          cuerdaPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          acordeonPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          zampoaPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          tecladoPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          bassPitchPattern0= ("intervalo", [intervalo index
                             octava]),
                          efectoPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          tarolaPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          guiraPitchPattern0 = ("intervalo", [intervalo index
                              octava]),
                          contrasPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
```

```
                          altavozPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          extrasPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          congasPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          jamblockPitchPattern0 = ("intervalo", [intervalo
                             index octava]),
                          clavePitchPattern0 = ("intervalo", [intervalo index
                              octava])
                          }

-- provee los intervalos de una lista
-- ("intervalo", [intervalo "unisono" 0, intervalo "3a" 0, intervalo "5a"
   0])
parseCambiarIntervalos :: H Layer
parseCambiarIntervalos = parseCambiarIntervalos' <*> parseLayer

parseCambiarIntervalos' :: H (Layer -> Layer)
parseCambiarIntervalos' = parseCambiarIntervalos'' <*> stringList

parseCambiarIntervalos'' :: H ([String] -> Layer -> Layer)
parseCambiarIntervalos'' = cambiarIntervalos <$ reserved "intervalo"

cambiarIntervalos :: [String] -> Layer -> Layer
cambiarIntervalos indices c = c {style = nuevoE}
  where
    indices' = fmap (\index -> intervalo index 0) indices
    nuevoE = (style c) {
                           cuerdaPitchPattern0 = ("intervalo", indices'),
                           acordeonPitchPattern0 = ("intervalo", indices'),
                           zampoaPitchPattern0 = ("intervalo", indices'),
                           tecladoPitchPattern0 = ("intervalo", indices'),
                           bassPitchPattern0= ("intervalo", indices'),
                           efectoPitchPattern0 = ("intervalo", indices'),
                           tarolaPitchPattern0 = ("intervalo", indices'),
                           guiraPitchPattern0 = ("intervalo", indices'),
                           contrasPitchPattern0 = ("intervalo", indices'),
                           altavozPitchPattern0 = ("intervalo", indices'),
                           extrasPitchPattern0 = ("intervalo", indices'),
                           congasPitchPattern0 = ("intervalo", indices'),
                           jamblockPitchPattern0 = ("intervalo", indices'),
                           clavePitchPattern0 = ("intervalo", indices')
                           }
-- provee el intervalo con respecto a la tonica y cualidad del acorde
```

```
parseCambiarIntervalo :: H Layer
parseCambiarIntervalo = parseCambiarIntervalo' <*> parseLayer

parseCambiarIntervalo' :: H (Layer -> Layer)
parseCambiarIntervalo' = parseCambiarIntervalo'' <*> string

parseCambiarIntervalo'' :: H (String -> Layer -> Layer)
parseCambiarIntervalo'' = cambiarIntervalo <$ reserved "intervalo"

cambiarIntervalo :: String -> Layer -> Layer
cambiarIntervalo index c = c {style = nuevoE}
  where nuevoE = (style c) {
                          cuerdaPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          acordeonPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          zampoaPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          tecladoPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          bassPitchPattern0= ("intervalo", [intervalo index
                             0]),
                          efectoPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          altavozPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          guiraPitchPattern0 = ("intervalo", [intervalo index
                              0]),
                          contrasPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          tarolaPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          extrasPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          congasPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          jamblockPitchPattern0 = ("intervalo", [intervalo
                             index 0]),
                          clavePitchPattern0 = ("intervalo", [intervalo index
                              0])
                          }
-- ritmo [1 2, 1 2 3 4]
parseCambiarRitmosAuto :: H Layer
parseCambiarRitmosAuto = parseCambiarRitmosAuto' <*> parseLayer
```

```
parseCambiarRitmosAuto' :: H (Layer -> Layer)
parseCambiarRitmosAuto' = parseCambiarRitmosAuto'' <*>
   parseListasDeListasDeAtaques -- rationalList

parseCambiarRitmosAuto'' :: H ([[Rational]] -> Layer -> Layer)
parseCambiarRitmosAuto'' = cambiarRitmosAuto <$ reserved "ritmo"

cambiarRitmosAuto :: [[Rational]] -> Layer -> Layer
cambiarRitmosAuto [x] c = c {style = nuevoE}
  where
    -- [x'] = concat [x]
    -- attack' = (x' - 1) / 4 -- metre? [1, 2, 3, 4] => 1 [0, 0.25, 0.5,
       0.75]
    -- metre |x' <= 0 = 1
         -- |otherwise = (realToFrac $ floor attack') + 1
    metre = toRational $ length [x] -- [[Nothing], [1, 2, 3]] = metre 2 -- (
       realToFrac $ floor rs') + 1
    rPat = cambiarRitmo'''' metre [x]
    nuevoE = (style c) {
                          cuerdaRhythmPattern0 = rPat,
                          acordeonRhythmPattern0 = rPat,
                          zampoaRhythmPattern0 = rPat,
                          tecladoRhythmPattern0 = rPat,
                          bassRhythmPattern0 = rPat,
                          guiraRhythmPattern0 = rPat,
                          contrasRhythmPattern0 = rPat,
                          tarolaRhythmPattern0 = rPat,
                          efectoRhythmPattern0 = rPat,
                          altavozRhythmPattern0 = rPat,
                          extrasRhythmPattern0 = rPat,
                          claveRhythmPattern0 = rPat,
                          jamblockRhythmPattern0 = rPat,
                          congasRhythmPattern0 = rPat
                          }

cambiarRitmosAuto attacks c = c {style = nuevoE}
  where
    -- attacks' = concat attacks
    -- attack' = ((maximum attacks') - 1) / 4 -- metre? [1, 2, 3, 4] => 1
       [0, 0.25, 0.5, 0.75]
    -- metre = (realToFrac $ floor attack') + 1
    metre = toRational $ length attacks -- [[Nothing], [1, 2, 3]] = metre 2
       -- (realToFrac $ floor rs') + 1
    rPat = cambiarRitmo'''' metre attacks
```

```
    nuevoE = (style c) {
                          cuerdaRhythmPattern0 = rPat,
                          acordeonRhythmPattern0 = rPat,
                          zampoaRhythmPattern0 = rPat,
                          tecladoRhythmPattern0 = rPat,
                          bassRhythmPattern0 = rPat,
                          guiraRhythmPattern0 = rPat,
                          contrasRhythmPattern0 = rPat,
                          tarolaRhythmPattern0 = rPat,
                          efectoRhythmPattern0 = rPat,
                          altavozRhythmPattern0 = rPat,
                          extrasRhythmPattern0 = rPat,
                          claveRhythmPattern0 = rPat,
                          jamblockRhythmPattern0 = rPat,
                          congasRhythmPattern0 = rPat
                          }

-- ritmo 4 cumbia cuerda, ritmo 1
parseCambiarRitmoAuto :: H Layer
parseCambiarRitmoAuto = parseCambiarRitmoAuto' <*> parseLayer

parseCambiarRitmoAuto' :: H (Layer -> Layer)
parseCambiarRitmoAuto' = parseCambiarRitmoAuto'' <*> rationalOrInteger

parseCambiarRitmoAuto'' :: H (Rational -> Layer -> Layer)
parseCambiarRitmoAuto'' = cambiarRitmoMetreAuto <$ reserved "ritmo"

cambiarRitmoMetreAuto :: Rational -> Layer -> Layer
cambiarRitmoMetreAuto attack c = c {style = nuevoE}
  where
    attack' = (attack - 1) / 4
    metre |attack <= 0 = 1 -- avoid metre of 0
          |otherwise = (realToFrac $ floor attack') + 1 -- metre? [1, 2, 3,
             4] => 1 [0, 0.25, 0.5, 0.75]

    nuevoE = (style c) {
                          cuerdaRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          acordeonRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          zampoaRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          tecladoRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
```

```
                          bassRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          guiraRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          contrasRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          tarolaRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          efectoRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          altavozRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          extrasRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          congasRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          jamblockRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          claveRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)]
                          }

-- type RhythmicPattern = [(Rational,Rational)]
-- ritmo 0.5 cumbia cuerda
parseCambiarRitmo :: H Layer
parseCambiarRitmo = parseCambiarRitmo' <*> parseLayer

parseCambiarRitmo' :: H (Layer -> Layer)
parseCambiarRitmo' = parseCambiarRitmo'' <*> rationalOrInteger

parseCambiarRitmo'' :: H (Rational -> Layer -> Layer)
parseCambiarRitmo'' = parseCambiarRitmo''' <*> rationalOrInteger

parseCambiarRitmo''' :: H (Rational -> Rational -> Layer -> Layer)
parseCambiarRitmo''' = cambiarRitmo <$ reserved "ritmo"

cambiarRitmo :: Rational -> Rational -> Layer -> Layer
cambiarRitmo metre attack c = c {style = nuevoE}
  where nuevoE = (style c) {
                          cuerdaRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          acordeonRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
                          zampoaRhythmPattern0 = catMaybes [cambiarRitmo' (
                             metre) (attack)],
```

```
                        tecladoRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        bassRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        guiraRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        contrasRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        tarolaRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        efectoRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        altavozRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        extrasRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        congasRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        jamblockRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)],
                        claveRhythmPattern0 = catMaybes [cambiarRitmo' (
                           metre) (attack)]
                        }

--e.g. ritmo 1 [0.125, 0.25] cumbia cuerda, deberia ser ritmo 1 [1 1.25
   1.5 1.75], ritmo 2 [1 2]
parseCambiarRitmos :: H Layer
parseCambiarRitmos = parseCambiarRitmos' <*> parseLayer

parseCambiarRitmos' :: H (Layer -> Layer)
parseCambiarRitmos' = parseCambiarRitmos'' <*> parseListasDeListasDeAtaques
   -- rationalList

parseCambiarRitmos'' :: H ([[Rational]] -> Layer -> Layer)
parseCambiarRitmos'' = parseCambiarRitmos''' <*> rationalOrInteger

parseCambiarRitmos''' :: H (Rational -> [[Rational]] -> Layer -> Layer)
parseCambiarRitmos''' = cambiarRitmos <$ reserved "ritmo"

cambiarRitmos :: Rational -> [[Rational]] -> Layer -> Layer
cambiarRitmos metre rs c = c {style = nuevoE}
  where
    -- attacks' = concat attacks
    nuevoE = (style c) {
                        cuerdaRhythmPattern0 = cambiarRitmo'''' metre rs,
```

```
                        acordeonRhythmPattern0 = cambiarRitmo'''' metre rs,
                        zampoaRhythmPattern0 = cambiarRitmo'''' metre rs,
                        tecladoRhythmPattern0 = cambiarRitmo'''' metre rs,
                        bassRhythmPattern0 = cambiarRitmo'''' metre rs,
                        guiraRhythmPattern0 = cambiarRitmo'''' metre rs,
                        contrasRhythmPattern0 = cambiarRitmo'''' metre rs,
                        tarolaRhythmPattern0 = cambiarRitmo'''' metre rs,
                        efectoRhythmPattern0 = cambiarRitmo'''' metre rs,
                        altavozRhythmPattern0 = cambiarRitmo'''' metre rs,
                        extrasRhythmPattern0 = cambiarRitmo'''' metre rs,
                        claveRhythmPattern0 = cambiarRitmo'''' metre rs,
                        jamblockRhythmPattern0 = cambiarRitmo'''' metre rs,
                        congasRhythmPattern0 = cambiarRitmo'''' metre rs
                        }

-- a function to change the attacks
cambiarRitmo' :: Rational -> Rational -> Maybe (Rational, Rational)
cambiarRitmo' metre attack = metreAndAttack
  where
    cuartosPorCompas = 4 -- * metre
    metreAndAttack | (attack >= 1) && (attack < (cuartosPorCompas + 1)) =
        Just (metre, attack') -- e.g. ritmo 1 [1 2 3 4] => ritmo 1 [0, 0.25,
         0.5, 0.75], 2 [1 2 3 4, 5 6 7 8] => ritmo 2 [0, 0.25, 0.5, 0.75, 1,
         1.25, 1.5, 1.75]
                   | otherwise = Nothing
                   where attack' = (attack - 1) / 4
                   -- where attack' = ((attack - 1) / 4) + indice

cambiarRitmo'' :: Rational -> [Rational] -> [(Rational, Rational)]
cambiarRitmo'' metre attacks = catMaybes $ fmap (cambiarRitmo' metre)
   attacks


cambiarRitmoSinMetre' :: Rational -> Rational -> Maybe Rational
cambiarRitmoSinMetre' metre attack = metreAndAttack
  where
    cuartosPorCompas = 4 -- * metre
    metreAndAttack | (attack >= 1) && (attack < (cuartosPorCompas + 1)) =
        Just attack' -- e.g. ritmo 1 [1 2 3 4] => ritmo 1 [0, 0.25, 0.5,
        0.75], 2 [1 2 3 4, 5 6 7 8] => ritmo 2 [0, 0.25, 0.5, 0.75, 1, 1.25,
         1.5, 1.75]
                   | otherwise = Nothing
                   where attack' = (attack - 1) / 4

cambiarRitmoSinMetre''' :: Rational -> [Rational] -> [Maybe Rational]
```

```
cambiarRitmoSinMetre''' metre attacks = fmap (cambiarRitmoSinMetre' metre)
   attacks

-- nota: falta dividir en /4 y luego sumar, ahora suma primero y divide
   despues pero no es lo mismo
cambiarRitmo'''' :: Rational -> [[Rational]] -> [(Rational, Rational)] -- [(
   metre, attack)]
cambiarRitmo'''' metre attacks = do
  let dividirAttacks = fmap (cambiarRitmoSinMetre''' metre) attacks -- [[
     Maybe Rational]]
  let zipIAttacks = zip [toRational 0 .. (metre - 1)] dividirAttacks -- [(0,
      [Just 1, Just 2.. ])), ...)]
  let sumarIaAttacks = fmap (\(i, xs) -> fmap (\x -> (+) <$> x <*> Just i)
     xs) zipIAttacks -- [[1,2, 3], [4, 5, 6]]
  let attacks' = catMaybes $ concat sumarIaAttacks
  fmap (\attack -> (metre, attack)) attacks'

-- cambia el gain
parseCambiarGain :: H Layer
parseCambiarGain = parseCambiarGain' <*> parseLayer

parseCambiarGain' :: H (Layer -> Layer)
parseCambiarGain' = parseCambiarGain'' <*> double

parseCambiarGain''':: H (Double -> Layer -> Layer)
parseCambiarGain'' = cambiarGain <$ (reserved "vol" <|> reserved "volumen")

cambiarGain :: Double -> Layer -> Layer
cambiarGain gain c = c {style = nuevoE}
  where nuevoE = (style c) {
                          cuerdaGainPattern0 = gain,
                          acordeonGainPattern0 = gain,
                          zampoaGainPattern0 = gain,
                          tecladoGainPattern0 = gain,
                          bassGainPattern0 = gain,
                          guiraGainPattern0 = gain,
                          contrasGainPattern0 = gain,
                          tarolaGainPattern0 = gain,
                          efectoGainPattern0 = gain,
                          altavozGainPattern0 = gain,
                          extrasGainPattern0 = gain,
                          congasGainPattern0 = gain,
                          jamblockGainPattern0 = gain,
                          claveGainPattern0 = gain
                          }
```

```
--cambia el paneo
parseCambiarPaneo :: H Layer
parseCambiarPaneo = parseCambiarPaneo' <*> parseLayer

parseCambiarPaneo' :: H (Layer -> Layer)
parseCambiarPaneo' = parseCambiarPaneo'' <*> double

parseCambiarPaneo''':: H (Double -> Layer -> Layer)
parseCambiarPaneo'' = cambiarPaneo <$ (reserved "pan" <|> reserved "paneo")

cambiarPaneo :: Double -> Layer -> Layer
cambiarPaneo pan c = c {style = nuevoE}
  where nuevoE = (style c) {
                          cuerdaPanPattern0 = pan,
                          acordeonPanPattern0 = pan,
                          zampoaPanPattern0 = pan,
                          tecladoPanPattern0 = pan,
                          bassPanPattern0 = pan,
                          guiraPanPattern0 = pan,
                          contrasPanPattern0 = pan,
                          tarolaPanPattern0 = pan,
                          efectoPanPattern0 = pan,
                          altavozPanPattern0 = pan,
                          extrasPanPattern0 = pan,
                          congasPanPattern0 = pan,
                          jamblockPanPattern0 = pan,
                          clavePanPattern0 = pan
                          }

-- a function that allows switching between presets
-- e.g: preset 1 cumbia bajo
parsePreset :: H Layer
parsePreset = parsePreset' <*> parseLayer

parsePreset' :: H (Layer -> Layer)
parsePreset' = parsePreset'' <*> int

parsePreset'' :: H (Int -> Layer -> Layer)
parsePreset'' = preset <$ reserved "preset"

preset :: Int -> Layer -> Layer
preset 0 c = c {style = nuevoE}
  where nuevoE = (style c) {
```

```
tecladoRhythmPattern0 = tecladoRhythmPattern0 (
   style c), -- ie. [   ],
tecladoSampleNPattern0 = tecladoSampleNPattern0 (
   style c),
tecladoPitchPattern0 = tecladoPitchPattern0 (style
   c),

cuerdaRhythmPattern0 = cuerdaRhythmPattern0 (style
   c),
cuerdaSampleNPattern0 = cuerdaSampleNPattern0 (
   style c),
cuerdaPitchPattern0 = cuerdaPitchPattern0 (style c)
   , -- or double? (nota [0, 2, 3]

acordeonRhythmPattern0 = acordeonRhythmPattern0 (
   style c),
acordeonSampleNPattern0 = acordeonSampleNPattern0 (
   style c),
acordeonPitchPattern0 = acordeonPitchPattern0 (
   style c), -- or double? (nota [0, 2, 3] cumbia)
    cuerda

zampoaRhythmPattern0 = zampoaRhythmPattern0 (style
   c),
zampoaSampleNPattern0 = zampoaSampleNPattern0 (
   style c),
zampoaPitchPattern0 = zampoaPitchPattern0 (style c)
   , -- or double? (nota [0, 2, 3]

bassRhythmPattern0 = bassRhythmPattern0 (style c),
   --i.e. [   ],
bassSampleNPattern0 = bassSampleNPattern0 (style c)
   ,
bassPitchPattern0 = bassPitchPattern0 (style c), --
    index from list of pitches i.e. [60, 64, 67]

guiraRhythmPattern0 = guiraRhythmPattern0 (style c)
   , --i.e. [   ]
guiraSampleNPattern0 = guiraSampleNPattern0 (style
   c),
guiraPitchPattern0 = guiraPitchPattern0 (style c),


contrasRhythmPattern0 = contrasRhythmPattern0 (
   style c),
```

```
                        contrasSampleNPattern0 = contrasSampleNPattern0 (
                           style c),
                        contrasPitchPattern0 = contrasPitchPattern0 (style
                           c),


                        tarolaRhythmPattern0 = tarolaRhythmPattern0 (style
                           c),
                        tarolaSampleNPattern0 =tarolaSampleNPattern0 (style
                            c),
                        tarolaPitchPattern0 = tarolaPitchPattern0 (style c)
                           ,

                        efectoRhythmPattern0 = efectoRhythmPattern0 (style
                           c),
                        efectoSampleNPattern0 = efectoSampleNPattern0 (
                           style c),
                        efectoPitchPattern0 = efectoPitchPattern0 (style c)
                           ,

                        extrasRhythmPattern0 = extrasRhythmPattern0 (style
                           c),
                        extrasSampleNPattern0 = extrasSampleNPattern0 (
                           style c),
                        extrasPitchPattern0 = extrasPitchPattern0 (style c)
                           ,

                        congasRhythmPattern0 = congasRhythmPattern0 (style
                           c),
                        congasSampleNPattern0 = congasSampleNPattern0 (
                           style c),
                        congasPitchPattern0 = congasPitchPattern0 (style c)
                           ,

                        claveRhythmPattern0 = claveRhythmPattern0 (style c)
                           ,
                        claveSampleNPattern0 = claveSampleNPattern0 (style
                           c),
                        clavePitchPattern0 = clavePitchPattern0 (style c)

                      }

preset 1 c = c {style = nuevoE}
  where nuevoE = (style c) {
```

```
                        tecladoRhythmPattern0 = tecladoRhythmPattern1 (
                           style c), -- ie. [   ],
                        tecladoSampleNPattern0 = tecladoSampleNPattern1 (
                           style c),

                        bassRhythmPattern0 = bassRhythmPattern1 (style c),
                           --i.e. [    ],
                        bassSampleNPattern0 = tecladoSampleNPattern0 (style
                            c),
                        bassPitchPattern0 = bassPitchPattern1 (style c),

                        claveRhythmPattern0 = claveRhythmPattern1 (style c)
                           ,
                        claveSampleNPattern0 = claveSampleNPattern1 (style
                           c),
                        clavePitchPattern0 = clavePitchPattern1 (style c)
                      }

preset 2 c = c {style = nuevoE}
   where nuevoE = (style c) {
                        bassRhythmPattern0 = bassRhythmPattern2 (style c),
                           --i.e. [    ],
                        bassSampleNPattern0 = tecladoSampleNPattern0 (style
                            c),
                        bassPitchPattern0 = bassPitchPattern2 (style c),

                        claveRhythmPattern0 = claveRhythmPattern2 (style c)
                           ,
                        claveSampleNPattern0 = claveSampleNPattern2 (style
                           c),
                        clavePitchPattern0 = clavePitchPattern2 (style c)
                        }
preset _ c = preset 0 c

-- funcion que modifica los acordes del teclado -- acompanamiento 2 => no
   ms de 4.99
parseacompanamiento :: H Layer
parseacompanamiento = parseacompanamiento' <*> parseLayer

parseacompanamiento' :: H (Layer -> Layer)
parseacompanamiento' = parseacompanamiento'' <*> rationalOrInteger

parseacompanamiento'' :: H (Rational -> Layer -> Layer)
parseacompanamiento'' = acompanamiento <$ (reserved "acompaamiento" <|>
   reserved "acompanamiento")
```

```
acompanamiento :: Rational -> Layer -> Layer
acompanamiento attack c = c {style = nuevoE}
  where
    -- n' | n == 0 = 0
       -- |otherwise = abs $ n - 1
    attack' = (attack - 1) / 4
    metre = (realToFrac $ floor attack') + 1 -- metre? [1, 2, 3, 4] => 1 [0,
        0.25, 0.5, 0.75]
    rPat = catMaybes [cambiarRitmo' metre attack] -- fmap (\n -> (metre, (
       realToFrac n) /4)) ns'
    nuevoE = (style c) {
                       tecladoRhythmPattern0 = rPat, -- [(1, (realToFrac n')
                          / 4)],
                       tecladoSampleNPattern0 = tecladoSampleNPattern0 (style
                           c),
                       tecladoPitchPattern0 = ("acorde", [intervalo "unisono"
                           0, intervalo "3a" 0, intervalo "5a" 0]) --
                          pitchPat (style c) -- ("acorde", [note])
                       }


-- funcion que modifica los acordes del teclado -- acompanamiento (2 4)

parseacompanamientos :: H Layer
parseacompanamientos = parseacompanamientos' <*> parseLayer

parseacompanamientos' :: H (Layer -> Layer)
parseacompanamientos' = parseacompanamientos'' <*>
   parseAtaquesAListaDeAtaques -- rationalList

parseacompanamientos'' :: H ([Rational] -> Layer -> Layer)
parseacompanamientos'' = acompanamientos <$ (reserved "acompaamiento" <|>
   reserved "acompanamiento")

acompanamientos :: [Rational] -> Layer -> Layer
acompanamientos ns c = c {style = nuevoE}
  where
    -- ns' = fmap (\n -> if (n == 0) then 0 else (abs $ n - 1)) ns -- [1,
       2, 3, 4] a [0, 1, 2, 3]
    -- rs' = ((maximum ns') - 1) / 4 -- metre? [1, 2, 3, 4] => 1 [0, 0.25,
       0.5, 0.75]
    ns' = ((maximum ns) - 1) / 4 -- metre? [1, 2, 3, 4] => 1 [0, 0.25, 0.5,
       0.75]
    metre = (realToFrac $ floor ns') + 1
```

```
    -- metre = 1
    rPat = cambiarRitmo'' metre ns -- fmap (\n -> (metre, (realToFrac n) /4)
       ) ns'
    nPat (NPattern1 xs) = NPattern1 $ concat $ replicate (length ns) xs
    nuevoE = (style c) {
                          tecladoRhythmPattern0 = rPat, --
                             listaDeStringsARhythmicPattern rPat notes,
                          tecladoSampleNPattern0 = nPat $
                             tecladoSampleNPattern0 (style c), --
                             listaDeStringsANPattern nPat notes,
                          tecladoPitchPattern0 = ("acorde", [intervalo "
                             unisono" 0, intervalo "3a" 0, intervalo "5a" 0])
                              -- ("acorde", concat $ notes) -- (PitchType, [
                             Note])
                        }

acompanamientosTest :: [Double] -> [Note] -> (String, String, String)
acompanamientosTest ns notes = do
  let ns' = fmap (\n -> if (n == 0) then 0 else (abs $ n - 1)) ns -- [1, 2,
     3, 4] a [0, 1, 2, 3]
  let metre = 1
  let notes' = replicate (length ns) [intervalo "unisono" 0, intervalo "3a"
     0, intervalo "5a" 0] -- [[Note]]
  let rPat = fmap (\n -> (metre, (realToFrac n) /4)) ns'
  let nPat = NPattern1 $ concat $ fmap (\x -> replicate (length ns) x) [0]
  let tecladoRhythmPattern = listaDeStringsARhythmicPattern rPat notes'
  let tecladoSampleNPattern = listaDeStringsANPattern nPat notes'
  let tecladoPitchPattern = ("acorde", concat $ notes')
  (show tecladoRhythmPattern, show tecladoSampleNPattern, show
     tecladoPitchPattern)

-- funcion que modifica los acordes del teclado -- acompanamiento 2 ("f"
   "3a" "5a") o acompanamiento 2 $ "f" ("3a" (-1)) "5a"
parseAcompanamientoConVoicingSel :: H Layer
parseAcompanamientoConVoicingSel = parseAcompanamientoConVoicingSel' <*>
   parseLayer

parseAcompanamientoConVoicingSel' :: H (Layer -> Layer)
parseAcompanamientoConVoicingSel' = parseAcompanamientoConVoicingSel'' <*>
   parseNoteList --

parseAcompanamientoConVoicingSel'' :: H ([Note] -> Layer -> Layer)
parseAcompanamientoConVoicingSel'' = parseAcompanamientoConVoicingSel''' <*>
    rationalOrInteger
```

```
parseAcompanamientoConVoicingSel''' :: H (Rational -> [Note] -> Layer ->
   Layer)
parseAcompanamientoConVoicingSel''' = acompanamientoConVoicingSel <$ (
   reserved "acompaamiento" <|> reserved "acompanamiento")

acompanamientoConVoicingSel :: Rational -> [Note] -> Layer -> Layer
acompanamientoConVoicingSel attack notes c = c {style = nuevoE}
  where
    attack' = (attack - 1) / 4
    metre = (realToFrac $ floor attack') + 1 -- metre? [1, 2, 3, 4] => 1 [0,
        0.25, 0.5, 0.75]
    rPat = catMaybes [cambiarRitmo' metre attack] -- fmap (\n -> (metre, (
       realToFrac n) /4)) ns'
    nPat = tecladoSampleNPattern0 (style c)
    nuevoE = (style c) {
                          tecladoRhythmPattern0 = rPat, --
                             listaDeStringsARhythmicPattern rPat notes,
                          tecladoSampleNPattern0 = nPat, --
                             listaDeStringsANPattern nPat notes,
                          tecladoPitchPattern0 = ("acorde", notes)-- ("acorde
                             ", [note])
                        }

---- test
-- p test this by taking the layer out
-- acompanamientoConVoicingSel :: Double -> [[Note]] -> Layer -> Layer
acompanamientoConVoicingSelTest n notes = do
    let n' | n == 0 = 1
           |otherwise = abs $ n - 1
    let rPat = [(1, (realToFrac n') / 4)]
    let nPat = NPattern1 [0]
    let tecladoRhythmPattern = listaDeStringsARhythmicPattern rPat notes
    let tecladoSampleNPattern = listaDeStringsANPattern nPat notes
    let tecladoPitchPattern = ("intervalo", listaDeStringsANote notes)
    (show tecladoRhythmPattern, show tecladoSampleNPattern, show
       tecladoPitchPattern)

-- acompanamiento (2 4) ("f" "3a" $ "5a" (-1))
parseAcompanamientosConVoicingSel :: H Layer
parseAcompanamientosConVoicingSel = parseAcompanamientosConVoicingSel' <*>
   parseLayer

parseAcompanamientosConVoicingSel' :: H (Layer -> Layer)
parseAcompanamientosConVoicingSel' = parseAcompanamientosConVoicingSel'' <*>
    parseNoteList -- parseNoteList --
```

```
   praseListaDeListaStringAListaDeAcordes

parseAcompanamientosConVoicingSel'' :: H ([Note] -> Layer -> Layer)
parseAcompanamientosConVoicingSel'' = parseAcompanamientosConVoicingSel'''
   <*> parseAtaquesAListaDeAtaques -- rationalList

parseAcompanamientosConVoicingSel''' :: H ([Rational] -> [Note] -> Layer ->
   Layer)
parseAcompanamientosConVoicingSel''' = acompanamientosConVoicingSel <$ (
   reserved "acompaamiento" <|> reserved "acompanamiento")

acompanamientosConVoicingSel :: [Rational] -> [Note] -> Layer -> Layer
acompanamientosConVoicingSel rs notes c = c {style = nuevoE}
  where
    -- metre = 1
    -- ns' = fmap (\n -> if (n == 0) then 0 else (abs $ n - 1)) ns -- [1,
       2, 3, 4] a [0, 1, 2, 3]
    -- rPat = fmap (\n -> (metre, (realToFrac n) /4)) ns'-- [(1, (
       realToFrac n') / 4)]
    rs' = ((maximum rs) - 1) / 4 -- metre? [1, 2, 3, 4] => 1 [0, 0.25, 0.5,
       0.75]
    metre = (realToFrac $ floor rs') + 1
    rPat = cambiarRitmo'' metre rs

    nPat (NPattern1 xs) = NPattern1 $ concat $ replicate (length rs) xs
    nuevoE = (style c) {
                          tecladoRhythmPattern0 = rPat, --
                             listaDeStringsARhythmicPattern rPat notes,
                          tecladoSampleNPattern0 = nPat $
                             tecladoSampleNPattern0 (style c), --
                             listaDeStringsANPattern nPat notes,
                          tecladoPitchPattern0 = ("acorde", notes)-- ("
                             intervalo", listaDeStringsANote notes)
                        }

parseNote :: H Note --(Relacion, Double, Octava)
parseNote = parseNoteConOctava
        <|> parseNoteConOctavaAuto
        --


parseNoteConOctava :: H Note
parseNoteConOctava = parseNoteConOctava' <*> double

parseNoteConOctava' :: H (Octava -> Note)
```

```
parseNoteConOctava' = do
  i <- string
  return $ \o -> intervalo i o


parseNoteConOctavaAuto :: H Note
parseNoteConOctavaAuto = do
  i <- string
  return $ intervalo i 0

parseNotesNoQuotes :: H [Note]
parseNotesNoQuotes = parseUnNoteConOctavaAuto
         <|> parseDosNoteConOctavaAuto
         <|> parseThreeNoteConOctavaAuto
         <|> parseFourNoteConOctavaAuto
         <|> parseFiveNoteConOctavaAuto
         <|> parseSixNoteConOctavaAuto
         <|> parseSevenNoteConOctavaAuto
         <|> parseEightNoteConOctavaAuto
         <|> parseNineNoteConOctavaAuto
         <|> parseTenNoteConOctavaAuto
         <|> parseElevenNoteConOctavaAuto
         <|> parseTwelveNoteConOctavaAuto
         <|> parseThirteenNoteConOctavaAuto
         <|> parseFourteenNoteConOctavaAuto
         <|> parseFifteenNoteConOctavaAuto
         <|> parseSixteenNoteConOctavaAuto



parseSixteenNoteConOctavaAuto :: H [Note]
parseSixteenNoteConOctavaAuto = parseSixteenNoteConOctavaAuto' <*>
   identifier

parseSixteenNoteConOctavaAuto' :: H (String -> [Note])
parseSixteenNoteConOctavaAuto' = parseSixteenNoteConOctavaAuto'' <*> int

parseSixteenNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'' = parseSixteenNoteConOctavaAuto''' <*>
   identifier

parseSixteenNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''' = parseSixteenNoteConOctavaAuto'''' <*> int
```

```
parseSixteenNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseSixteenNoteConOctavaAuto'''' = parseSixteenNoteConOctavaAuto''''' <*>
   identifier

parseSixteenNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseSixteenNoteConOctavaAuto''''' = parseSixteenNoteConOctavaAuto'''''' <*>
    int

parseSixteenNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''' = parseSixteenNoteConOctavaAuto'''''''
   <*> identifier

parseSixteenNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
    String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''' = parseSixteenNoteConOctavaAuto''''''''
    <*> int

parseSixteenNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''' = parseSixteenNoteConOctavaAuto
   ''''''''' <*> identifier

parseSixteenNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''' = parseSixteenNoteConOctavaAuto
   '''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''' = parseSixteenNoteConOctavaAuto
   ''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> [Note
   ])
parseSixteenNoteConOctavaAuto''''''''''' = parseSixteenNoteConOctavaAuto
   ''''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
```

```
parseSixteenNoteConOctavaAuto'''''''''''' = parseSixteenNoteConOctavaAuto
   '''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto'''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''' = parseSixteenNoteConOctavaAuto
   '''''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''' = parseSixteenNoteConOctavaAuto
   '''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto'''''''''''''' :: H (String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''' = parseSixteenNoteConOctavaAuto
   '''''''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''''' =
   parseSixteenNoteConOctavaAuto'''''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto'''''''''''''''' :: H (String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''''' =
   parseSixteenNoteConOctavaAuto'''''''''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''''''''''' :: H ( Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''''''' =
   parseSixteenNoteConOctavaAuto'''''''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto'''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''''''' =
   parseSixteenNoteConOctavaAuto'''''''''''''''''' <*> int
```

```
parseSixteenNoteConOctavaAuto'''''''''''''''''''' :: H (Int -> String -> Int
    -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto'''''''''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto''''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''''''''''''''' :: H (Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto'''''''''''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto''''''''''''''''''''''' :: H ( String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''''''''''''''''' :: H (Int -> String ->
    Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto'''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto'''''''''''''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto''''''''''''''''''''''''' :: H (String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''''''' <*> int

parseSixteenNoteConOctavaAuto'''''''''''''''''''''''''' :: H (Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
```

```
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''' :: H (String -> Int
    -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''' <*> int

parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''' :: H (Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''' <*> identifier

parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''' :: H (String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''' <*> int

parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''''' :: H (Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''''' =
   parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''''' <*>
   identifier

parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''''''' :: H (String ->
    Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseSixteenNoteConOctavaAuto''''''''''''''''''''''''''''''''''''' = do
  i1 <- int
```

```
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 i10 c10 i11
     c11 i12 c12 i13 c13 i14 c14 i15 c15 i16 c16 -> [catIntervalo i1 c1,
     catIntervalo i2 c2, catIntervalo i3 c3, catIntervalo i4 c4,
     catIntervalo i5 c5, catIntervalo i6 c6, catIntervalo i7 c7,
     catIntervalo i8 c8, catIntervalo i9 c9, catIntervalo i10 c10,
     catIntervalo i11 c11, catIntervalo i12 c12, catIntervalo i13 c13,
     catIntervalo i14 c14, catIntervalo i15 c15, catIntervalo i16 c16]


parseFifteenNoteConOctavaAuto :: H [Note]
parseFifteenNoteConOctavaAuto = parseFifteenNoteConOctavaAuto' <*>
   identifier

parseFifteenNoteConOctavaAuto' :: H (String -> [Note])
parseFifteenNoteConOctavaAuto' = parseFifteenNoteConOctavaAuto'' <*> int

parseFifteenNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'' = parseFifteenNoteConOctavaAuto''' <*>
   identifier

parseFifteenNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''' = parseFifteenNoteConOctavaAuto'''' <*> int

parseFifteenNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseFifteenNoteConOctavaAuto'''' = parseFifteenNoteConOctavaAuto''''' <*>
   identifier

parseFifteenNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseFifteenNoteConOctavaAuto''''' = parseFifteenNoteConOctavaAuto'''''' <*>
    int

parseFifteenNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''' = parseFifteenNoteConOctavaAuto''''''' <*>
   identifier

parseFifteenNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
    String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''' = parseFifteenNoteConOctavaAuto'''''''' <*>
    int

parseFifteenNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> [Note])
```

```
parseFifteenNoteConOctavaAuto''''''''' = parseFifteenNoteConOctavaAuto
   ''''''''' <*> identifier

parseFifteenNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''''' = parseFifteenNoteConOctavaAuto
   '''''''''' <*> int

parseFifteenNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''' = parseFifteenNoteConOctavaAuto
   ''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> [Note
   ])
parseFifteenNoteConOctavaAuto''''''''''' = parseFifteenNoteConOctavaAuto
   ''''''''''' <*> int

parseFifteenNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''' = parseFifteenNoteConOctavaAuto
   ''''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto''''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''''''''' = parseFifteenNoteConOctavaAuto
   ''''''''''''' <*> int

parseFifteenNoteConOctavaAuto'''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''' = parseFifteenNoteConOctavaAuto
   '''''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto''''''''''''''' :: H (String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''''''''''' = parseFifteenNoteConOctavaAuto
   ''''''''''''''' <*> int

parseFifteenNoteConOctavaAuto'''''''''''''''' :: H ( Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
```

```
   String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto'''''''''''''''' :: H (String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''' <*> int

parseFifteenNoteConOctavaAuto'''''''''''''''' :: H (Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto'''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''''' <*> int

parseFifteenNoteConOctavaAuto'''''''''''''''''' :: H (Int -> String -> Int
    -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto'''''''''''''''''' :: H ( String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''''' <*> int

parseFifteenNoteConOctavaAuto'''''''''''''''''''' :: H (Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''''''' <*> identifier
```

```
parseFifteenNoteConOctavaAuto'''''''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''''''''''' <*> int

parseFifteenNoteConOctavaAuto''''''''''''''''''''''''' :: H (Int -> String ->
    Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''''''''''''''''''''' =
   parseFifteenNoteConOctavaAuto''''''''''''''''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto'''''''''''''''''''''''''' :: H (String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''''''''''''' <*> int

parseFifteenNoteConOctavaAuto''''''''''''''''''''''''''' :: H (Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''''''''''''''''''''' =
   parseFifteenNoteConOctavaAuto''''''''''''''''''''''''' <*> identifier

parseFifteenNoteConOctavaAuto'''''''''''''''''''''''''' :: H (String -> Int
    -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFifteenNoteConOctavaAuto'''''''''''''''''''''''''' =
   parseFifteenNoteConOctavaAuto'''''''''''''''''''''''''' <*> int

parseFifteenNoteConOctavaAuto''''''''''''''''''''''''''' :: H (Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''''''''''''''''''''''' =
   parseFifteenNoteConOctavaAuto''''''''''''''''''''''''''' <*> identifier
```

```
parseFifteenNoteConOctavaAuto''''''''''''''''''''''''''''' :: H (String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFifteenNoteConOctavaAuto''''''''''''''''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 i10 c10 i11
     c11 i12 c12 i13 c13 i14 c14 i15 c15 -> [catIntervalo i1 c1,
     catIntervalo i2 c2, catIntervalo i3 c3, catIntervalo i4 c4,
     catIntervalo i5 c5, catIntervalo i6 c6, catIntervalo i7 c7,
     catIntervalo i8 c8, catIntervalo i9 c9, catIntervalo i10 c10,
     catIntervalo i11 c11, catIntervalo i12 c12, catIntervalo i13 c13,
     catIntervalo i14 c14, catIntervalo i15 c15]


parseFourteenNoteConOctavaAuto :: H [Note]
parseFourteenNoteConOctavaAuto = parseFourteenNoteConOctavaAuto' <*>
   identifier

parseFourteenNoteConOctavaAuto' :: H (String -> [Note])
parseFourteenNoteConOctavaAuto' = parseFourteenNoteConOctavaAuto'' <*> int

parseFourteenNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'' = parseFourteenNoteConOctavaAuto''' <*>
   identifier

parseFourteenNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto''' = parseFourteenNoteConOctavaAuto'''' <*>
   int

parseFourteenNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseFourteenNoteConOctavaAuto'''' = parseFourteenNoteConOctavaAuto''''' <*>
    identifier

parseFourteenNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseFourteenNoteConOctavaAuto''''' = parseFourteenNoteConOctavaAuto''''''
   <*> int

parseFourteenNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
    Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''' = parseFourteenNoteConOctavaAuto'''''''
    <*> identifier
```

```
parseFourteenNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto''''''' = parseFourteenNoteConOctavaAuto
   '''''''' <*> int

parseFourteenNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''' = parseFourteenNoteConOctavaAuto
   ''''''''' <*> identifier

parseFourteenNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int
    -> String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto''''''''' = parseFourteenNoteConOctavaAuto
   '''''''''' <*> int

parseFourteenNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''' = parseFourteenNoteConOctavaAuto
   ''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> [Note
   ])
parseFourteenNoteConOctavaAuto''''''''''' = parseFourteenNoteConOctavaAuto
   '''''''''''' <*> int

parseFourteenNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''' = parseFourteenNoteConOctavaAuto
   ''''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto''''''''''''' :: H (String -> Int -> String ->
    Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseFourteenNoteConOctavaAuto''''''''''''' = parseFourteenNoteConOctavaAuto
   '''''''''''''' <*> int

parseFourteenNoteConOctavaAuto'''''''''''''' :: H ( Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''' =
   parseFourteenNoteConOctavaAuto''''''''''''''' <*> identifier
```

```
parseFourteenNoteConOctavaAuto'''''''''''''''' :: H (String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''''' =
   parseFourteenNoteConOctavaAuto'''''''''''''''' <*> int

parseFourteenNoteConOctavaAuto''''''''''''''''' :: H (Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto''''''''''''''''' =
   parseFourteenNoteConOctavaAuto''''''''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto'''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''''''' =
   parseFourteenNoteConOctavaAuto'''''''''''''''''' <*> int

parseFourteenNoteConOctavaAuto''''''''''''''''''' :: H (Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto''''''''''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto'''''''''''''''''''' :: H ( String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto'''''''''''''''''''' <*> int

parseFourteenNoteConOctavaAuto''''''''''''''''''''' :: H (Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFourteenNoteConOctavaAuto''''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto''''''''''''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto'''''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto'''''''''''''''''''''' <*> int
```

```
parseFourteenNoteConOctavaAuto'''''''''''''''''''''' :: H (Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto'''''''''''''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto''''''''''''''''''''''' :: H (String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto''''''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto''''''''''''''''''''''' <*> int

parseFourteenNoteConOctavaAuto'''''''''''''''''''''''' :: H (Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto'''''''''''''''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto''''''''''''''''''''''''' :: H (String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto''''''''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto''''''''''''''''''''''''' <*> int

parseFourteenNoteConOctavaAuto'''''''''''''''''''''''''' :: H (Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseFourteenNoteConOctavaAuto'''''''''''''''''''''''''' =
   parseFourteenNoteConOctavaAuto'''''''''''''''''''''''''' <*> identifier

parseFourteenNoteConOctavaAuto''''''''''''''''''''''''''' :: H (String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseFourteenNoteConOctavaAuto''''''''''''''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 i10 c10 i11
     c11 i12 c12 i13 c13 i14 c14 -> [catIntervalo i1 c1, catIntervalo i2 c2,
      catIntervalo i3 c3, catIntervalo i4 c4, catIntervalo i5 c5,
```

```
     catIntervalo i6 c6, catIntervalo i7 c7, catIntervalo i8 c8,
     catIntervalo i9 c9, catIntervalo i10 c10, catIntervalo i11 c11,
     catIntervalo i12 c12, catIntervalo i13 c13, catIntervalo i14 c14]


parseThirteenNoteConOctavaAuto :: H [Note]
parseThirteenNoteConOctavaAuto = parseThirteenNoteConOctavaAuto' <*>
   identifier

parseThirteenNoteConOctavaAuto' :: H (String -> [Note])
parseThirteenNoteConOctavaAuto' = parseThirteenNoteConOctavaAuto'' <*> int

parseThirteenNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'' = parseThirteenNoteConOctavaAuto''' <*>
   identifier

parseThirteenNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto''' = parseThirteenNoteConOctavaAuto'''' <*>
   int

parseThirteenNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseThirteenNoteConOctavaAuto'''' = parseThirteenNoteConOctavaAuto''''' <*>
    identifier

parseThirteenNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseThirteenNoteConOctavaAuto''''' = parseThirteenNoteConOctavaAuto''''''
   <*> int

parseThirteenNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
    Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''' = parseThirteenNoteConOctavaAuto'''''''
    <*> identifier

parseThirteenNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto''''''' = parseThirteenNoteConOctavaAuto
   '''''''' <*> int

parseThirteenNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''''' = parseThirteenNoteConOctavaAuto
   ''''''''' <*> identifier
```

```
parseThirteenNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int
    -> String -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto''''''''' = parseThirteenNoteConOctavaAuto
   '''''''''' <*> int

parseThirteenNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''' = parseThirteenNoteConOctavaAuto
   ''''''''''' <*> identifier

parseThirteenNoteConOctavaAuto''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> [Note
   ])
parseThirteenNoteConOctavaAuto''''''''''' = parseThirteenNoteConOctavaAuto
   ''''''''''' <*> int

parseThirteenNoteConOctavaAuto'''''''''''' :: H ( Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''''' = parseThirteenNoteConOctavaAuto
   ''''''''''''' <*> identifier

parseThirteenNoteConOctavaAuto''''''''''''' :: H (String -> Int -> String ->
    Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseThirteenNoteConOctavaAuto''''''''''''' = parseThirteenNoteConOctavaAuto
   ''''''''''''' <*> int

parseThirteenNoteConOctavaAuto'''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''''''' =
   parseThirteenNoteConOctavaAuto'''''''''''''' <*> identifier

parseThirteenNoteConOctavaAuto''''''''''''''' :: H (String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto''''''''''''''' =
   parseThirteenNoteConOctavaAuto''''''''''''''' <*> int

parseThirteenNoteConOctavaAuto'''''''''''''''' :: H (Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''''''''' =
   parseThirteenNoteConOctavaAuto'''''''''''''''' <*> identifier
```

```
parseThirteenNoteConOctavaAuto'''''''''''''''' :: H ( String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''''''''' =
   parseThirteenNoteConOctavaAuto'''''''''''''''' <*> int

parseThirteenNoteConOctavaAuto''''''''''''''''' :: H (Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto''''''''''''''''' =
   parseThirteenNoteConOctavaAuto''''''''''''''''' <*> identifier

parseThirteenNoteConOctavaAuto'''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''''''''''' =
   parseThirteenNoteConOctavaAuto'''''''''''''''''' <*> int

parseThirteenNoteConOctavaAuto''''''''''''''''''' :: H (Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseThirteenNoteConOctavaAuto''''''''''''''''''' =
   parseThirteenNoteConOctavaAuto''''''''''''''''''' <*> identifier

parseThirteenNoteConOctavaAuto'''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''''''''''''' =
   parseThirteenNoteConOctavaAuto'''''''''''''''''''' <*> int

parseThirteenNoteConOctavaAuto''''''''''''''''''''' :: H (Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto''''''''''''''''''''' =
   parseThirteenNoteConOctavaAuto''''''''''''''''''''' <*> identifier

parseThirteenNoteConOctavaAuto'''''''''''''''''''''' :: H (String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
```

```
parseThirteenNoteConOctavaAuto''''''''''''''''''''''' =
   parseThirteenNoteConOctavaAuto''''''''''''''''''''''' <*> int

parseThirteenNoteConOctavaAuto'''''''''''''''''''''''' :: H (Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto'''''''''''''''''''''''' =
   parseThirteenNoteConOctavaAuto'''''''''''''''''''''''' <*> identifier

parseThirteenNoteConOctavaAuto''''''''''''''''''''''''' :: H (String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseThirteenNoteConOctavaAuto''''''''''''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 i10 c10 i11
     c11 i12 c12 i13 c13 -> [catIntervalo i1 c1, catIntervalo i2 c2,
     catIntervalo i3 c3, catIntervalo i4 c4, catIntervalo i5 c5,
     catIntervalo i6 c6, catIntervalo i7 c7, catIntervalo i8 c8,
     catIntervalo i9 c9, catIntervalo i10 c10, catIntervalo i11 c11,
     catIntervalo i12 c12, catIntervalo i13 c13]


parseTwelveNoteConOctavaAuto :: H [Note]
parseTwelveNoteConOctavaAuto = parseTwelveNoteConOctavaAuto' <*> identifier

parseTwelveNoteConOctavaAuto' :: H (String -> [Note])
parseTwelveNoteConOctavaAuto' = parseTwelveNoteConOctavaAuto'' <*> int

parseTwelveNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseTwelveNoteConOctavaAuto'' = parseTwelveNoteConOctavaAuto''' <*>
   identifier

parseTwelveNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''' = parseTwelveNoteConOctavaAuto'''' <*> int

parseTwelveNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseTwelveNoteConOctavaAuto'''' = parseTwelveNoteConOctavaAuto''''' <*>
   identifier

parseTwelveNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
```

```
parseTwelveNoteConOctavaAuto''''' = parseTwelveNoteConOctavaAuto'''''' <*>
   int

parseTwelveNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> [Note])
parseTwelveNoteConOctavaAuto'''''' = parseTwelveNoteConOctavaAuto''''''' <*>
    identifier

parseTwelveNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''' = parseTwelveNoteConOctavaAuto'''''''''
   <*> int

parseTwelveNoteConOctavaAuto''''''''' :: H (Int -> String -> Int -> String ->
    Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''' = parseTwelveNoteConOctavaAuto'''''''''''
    <*> identifier

parseTwelveNoteConOctavaAuto''''''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''' = parseTwelveNoteConOctavaAuto
   ''''''''''' <*> int

parseTwelveNoteConOctavaAuto''''''''''' :: H ( Int -> String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''' = parseTwelveNoteConOctavaAuto
   ''''''''''' <*> identifier

parseTwelveNoteConOctavaAuto''''''''''' :: H (String -> Int -> String -> Int
    -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''' = parseTwelveNoteConOctavaAuto
   ''''''''''' <*> int

parseTwelveNoteConOctavaAuto''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''' = parseTwelveNoteConOctavaAuto
   ''''''''''' <*> identifier

parseTwelveNoteConOctavaAuto''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''' = parseTwelveNoteConOctavaAuto
   ''''''''''' <*> int
```

```
parseTwelveNoteConOctavaAuto'''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto'''''''''''''' = parseTwelveNoteConOctavaAuto
   '''''''''''''' <*> identifier

parseTwelveNoteConOctavaAuto''''''''''''''' :: H ( String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''''''' = parseTwelveNoteConOctavaAuto
   ''''''''''''''' <*> int

parseTwelveNoteConOctavaAuto'''''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto'''''''''''''''' = parseTwelveNoteConOctavaAuto
   '''''''''''''''' <*> identifier

parseTwelveNoteConOctavaAuto''''''''''''''''' :: H (String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''''''''' = parseTwelveNoteConOctavaAuto
   ''''''''''''''''' <*> int

parseTwelveNoteConOctavaAuto'''''''''''''''''' :: H (Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto'''''''''''''''''' =
   parseTwelveNoteConOctavaAuto'''''''''''''''''' <*> identifier

parseTwelveNoteConOctavaAuto''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''''''''''' =
   parseTwelveNoteConOctavaAuto''''''''''''''''''' <*> int

parseTwelveNoteConOctavaAuto'''''''''''''''''''' :: H (Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseTwelveNoteConOctavaAuto'''''''''''''''''''' =
   parseTwelveNoteConOctavaAuto'''''''''''''''''''' <*> identifier
```

```
parseTwelveNoteConOctavaAuto'''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto'''''''''''''''''''' =
   parseTwelveNoteConOctavaAuto'''''''''''''''''''' <*> int

parseTwelveNoteConOctavaAuto''''''''''''''''''''' :: H (Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto''''''''''''''''''''' =
   parseTwelveNoteConOctavaAuto''''''''''''''''''''' <*> identifier

parseTwelveNoteConOctavaAuto'''''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseTwelveNoteConOctavaAuto'''''''''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 i10 c10 i11
     c11 i12 c12 -> [catIntervalo i1 c1, catIntervalo i2 c2, catIntervalo i3
      c3, catIntervalo i4 c4, catIntervalo i5 c5, catIntervalo i6 c6,
     catIntervalo i7 c7, catIntervalo i8 c8, catIntervalo i9 c9,
     catIntervalo i10 c10, catIntervalo i11 c11, catIntervalo i12 c12]

parseElevenNoteConOctavaAuto :: H [Note]
parseElevenNoteConOctavaAuto = parseElevenNoteConOctavaAuto' <*> identifier

parseElevenNoteConOctavaAuto' :: H (String -> [Note])
parseElevenNoteConOctavaAuto' = parseElevenNoteConOctavaAuto'' <*> int

parseElevenNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseElevenNoteConOctavaAuto'' = parseElevenNoteConOctavaAuto''' <*>
   identifier

parseElevenNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto''' = parseElevenNoteConOctavaAuto'''' <*> int

parseElevenNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseElevenNoteConOctavaAuto'''' = parseElevenNoteConOctavaAuto''''' <*>
   identifier
```

```
parseElevenNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseElevenNoteConOctavaAuto''''' = parseElevenNoteConOctavaAuto'''''' <*>
   int

parseElevenNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''' = parseElevenNoteConOctavaAuto''''''' <*>
    identifier

parseElevenNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto''''''' = parseElevenNoteConOctavaAuto'''''''' 
   <*> int

parseElevenNoteConOctavaAuto'''''''' :: H ( Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''' = parseElevenNoteConOctavaAuto''''''''' 
    <*> identifier

parseElevenNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto''''''''' = parseElevenNoteConOctavaAuto
   '''''''''' <*> int

parseElevenNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''''' = parseElevenNoteConOctavaAuto
   ''''''''''' <*> identifier

parseElevenNoteConOctavaAuto''''''''''' :: H (String -> Int -> String -> Int
    -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto''''''''''' = parseElevenNoteConOctavaAuto
   '''''''''''' <*> int

parseElevenNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''' = parseElevenNoteConOctavaAuto
   ''''''''''''' <*> identifier

parseElevenNoteConOctavaAuto''''''''''''' :: H ( String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
```

```
parseElevenNoteConOctavaAuto'''''''''''''' = parseElevenNoteConOctavaAuto
   '''''''''''''' <*> int

parseElevenNoteConOctavaAuto'''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''''' = parseElevenNoteConOctavaAuto
   '''''''''''''''' <*> identifier

parseElevenNoteConOctavaAuto'''''''''''''''' :: H (String -> Int -> String ->
    Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''''''' = parseElevenNoteConOctavaAuto
   '''''''''''''''' <*> int

parseElevenNoteConOctavaAuto'''''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''''''' = parseElevenNoteConOctavaAuto
   '''''''''''''''''' <*> identifier

parseElevenNoteConOctavaAuto'''''''''''''''''' :: H (String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''''''''' = parseElevenNoteConOctavaAuto
   '''''''''''''''''''' <*> int

parseElevenNoteConOctavaAuto'''''''''''''''''''' :: H (Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''''''''''' =
   parseElevenNoteConOctavaAuto'''''''''''''''''''' <*> identifier

parseElevenNoteConOctavaAuto'''''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''''''''''''' =
   parseElevenNoteConOctavaAuto'''''''''''''''''''''' <*> int

parseElevenNoteConOctavaAuto'''''''''''''''''''''' :: H (Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
```

```
parseElevenNoteConOctavaAuto'''''''''''''''''''' =
   parseElevenNoteConOctavaAuto'''''''''''''''''''' <*> identifier

parseElevenNoteConOctavaAuto'''''''''''''''''''' :: H (String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseElevenNoteConOctavaAuto'''''''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 i10 c10 i11
     c11 -> [catIntervalo i1 c1, catIntervalo i2 c2, catIntervalo i3 c3,
     catIntervalo i4 c4, catIntervalo i5 c5, catIntervalo i6 c6,
     catIntervalo i7 c7, catIntervalo i8 c8, catIntervalo i9 c9,
     catIntervalo i10 c10, catIntervalo i11 c11]


parseTenNoteConOctavaAuto :: H [Note]
parseTenNoteConOctavaAuto = parseTenNoteConOctavaAuto' <*> identifier

parseTenNoteConOctavaAuto' :: H (String -> [Note])
parseTenNoteConOctavaAuto' = parseTenNoteConOctavaAuto'' <*> int

parseTenNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseTenNoteConOctavaAuto'' = parseTenNoteConOctavaAuto''' <*> identifier

parseTenNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto''' = parseTenNoteConOctavaAuto'''' <*> int

parseTenNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [Note
   ])
parseTenNoteConOctavaAuto'''' = parseTenNoteConOctavaAuto''''' <*>
   identifier

parseTenNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseTenNoteConOctavaAuto''''' = parseTenNoteConOctavaAuto'''''' <*> int

parseTenNoteConOctavaAuto'''''' :: H ( Int -> String -> Int -> String -> Int
    -> String -> [Note])
parseTenNoteConOctavaAuto'''''' = parseTenNoteConOctavaAuto''''''' <*>
   identifier

parseTenNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto''''''' = parseTenNoteConOctavaAuto'''''''' <*> int
```

```
parseTenNoteConOctavaAuto''''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto''''''''' = parseTenNoteConOctavaAuto'''''''''' <*>
   identifier

parseTenNoteConOctavaAuto'''''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto'''''''''' = parseTenNoteConOctavaAuto''''''''''' <*>
    int

parseTenNoteConOctavaAuto''''''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto''''''''''' = parseTenNoteConOctavaAuto'''''''''''
   <*> identifier

parseTenNoteConOctavaAuto'''''''''''' :: H ( String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto'''''''''''' = parseTenNoteConOctavaAuto''''''''''''
    <*> int

parseTenNoteConOctavaAuto''''''''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String -> [
   Note])
parseTenNoteConOctavaAuto''''''''''''' = parseTenNoteConOctavaAuto
   ''''''''''''' <*> identifier

parseTenNoteConOctavaAuto'''''''''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> [Note])
parseTenNoteConOctavaAuto'''''''''''''' = parseTenNoteConOctavaAuto
   ''''''''''''''' <*> int

parseTenNoteConOctavaAuto''''''''''''''' :: H (Int -> String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> [Note])
parseTenNoteConOctavaAuto''''''''''''''' = parseTenNoteConOctavaAuto
   ''''''''''''''' <*> identifier

parseTenNoteConOctavaAuto'''''''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto'''''''''''''''' = parseTenNoteConOctavaAuto
   '''''''''''''''' <*> int
```

```
parseTenNoteConOctavaAuto'''''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto'''''''''''''''' = parseTenNoteConOctavaAuto
   '''''''''''''''' <*> identifier

parseTenNoteConOctavaAuto''''''''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto''''''''''''''''' = parseTenNoteConOctavaAuto
   ''''''''''''''''' <*> int

parseTenNoteConOctavaAuto'''''''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseTenNoteConOctavaAuto'''''''''''''''''' = parseTenNoteConOctavaAuto
   '''''''''''''''''' <*> identifier

parseTenNoteConOctavaAuto''''''''''''''''''' :: H (String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> [Note
   ])
parseTenNoteConOctavaAuto''''''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 i10 c10 -> [
     catIntervalo i1 c1, catIntervalo i2 c2, catIntervalo i3 c3,
     catIntervalo i4 c4, catIntervalo i5 c5, catIntervalo i6 c6,
     catIntervalo i7 c7, catIntervalo i8 c8, catIntervalo i9 c9,
     catIntervalo i10 c10]


parseNineNoteConOctavaAuto :: H [Note]
parseNineNoteConOctavaAuto = parseNineNoteConOctavaAuto' <*> identifier

parseNineNoteConOctavaAuto' :: H (String -> [Note])
parseNineNoteConOctavaAuto' = parseNineNoteConOctavaAuto'' <*> int

parseNineNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseNineNoteConOctavaAuto'' = parseNineNoteConOctavaAuto''' <*> identifier

parseNineNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto''' = parseNineNoteConOctavaAuto'''' <*> int

parseNineNoteConOctavaAuto'''' :: H ( Int -> String -> Int -> String -> [
   Note])
```

```
parseNineNoteConOctavaAuto'''' = parseNineNoteConOctavaAuto''''' <*>
   identifier

parseNineNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseNineNoteConOctavaAuto''''' = parseNineNoteConOctavaAuto'''''' <*> int

parseNineNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String -> Int
    -> String -> [Note])
parseNineNoteConOctavaAuto'''''' = parseNineNoteConOctavaAuto''''''' <*>
   identifier

parseNineNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto''''''' = parseNineNoteConOctavaAuto'''''''' <*>
   int

parseNineNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto'''''''' = parseNineNoteConOctavaAuto''''''''' <*>
    identifier

parseNineNoteConOctavaAuto''''''''' :: H ( String -> Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto''''''''' = parseNineNoteConOctavaAuto''''''''''
   <*> int

parseNineNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int -> String ->
    Int -> String -> Int -> String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto'''''''''' = parseNineNoteConOctavaAuto'''''''''''
    <*> identifier

parseNineNoteConOctavaAuto''''''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto''''''''''' = parseNineNoteConOctavaAuto
   ''''''''''''' <*> int

parseNineNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> Int -> String -> [
   Note])
parseNineNoteConOctavaAuto'''''''''''' = parseNineNoteConOctavaAuto
   '''''''''''''' <*> identifier

parseNineNoteConOctavaAuto''''''''''''' :: H (String -> Int -> String -> Int
    -> String -> Int -> String -> Int -> String -> Int -> String -> Int ->
```

```
   String -> [Note])
parseNineNoteConOctavaAuto'''''''''''' = parseNineNoteConOctavaAuto
   '''''''''''' <*> int

parseNineNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto'''''''''''' = parseNineNoteConOctavaAuto
   '''''''''''' <*> identifier

parseNineNoteConOctavaAuto'''''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto'''''''''''''' = parseNineNoteConOctavaAuto
   '''''''''''''' <*> int

parseNineNoteConOctavaAuto'''''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto'''''''''''''''' = parseNineNoteConOctavaAuto
   '''''''''''''''' <*> identifier

parseNineNoteConOctavaAuto'''''''''''''''''' :: H (String -> Int -> String ->
    Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> [Note])
parseNineNoteConOctavaAuto'''''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 i9 c9 -> [
     catIntervalo i1 c1, catIntervalo i2 c2, catIntervalo i3 c3,
     catIntervalo i4 c4, catIntervalo i5 c5, catIntervalo i6 c6,
     catIntervalo i7 c7, catIntervalo i8 c8, catIntervalo i9 c9]


parseEightNoteConOctavaAuto :: H [Note]
parseEightNoteConOctavaAuto = parseEightNoteConOctavaAuto' <*> identifier

parseEightNoteConOctavaAuto' :: H (String -> [Note])
parseEightNoteConOctavaAuto' = parseEightNoteConOctavaAuto'' <*> int

parseEightNoteConOctavaAuto'' :: H ( Int -> String -> [Note])
parseEightNoteConOctavaAuto'' = parseEightNoteConOctavaAuto''' <*>
   identifier

parseEightNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto''' = parseEightNoteConOctavaAuto'''' <*> int
```

```
parseEightNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseEightNoteConOctavaAuto'''' = parseEightNoteConOctavaAuto''''' <*>
   identifier

parseEightNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseEightNoteConOctavaAuto''''' = parseEightNoteConOctavaAuto'''''' <*> int

parseEightNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> [Note])
parseEightNoteConOctavaAuto'''''' = parseEightNoteConOctavaAuto''''''' <*>
   identifier

parseEightNoteConOctavaAuto''''''' :: H ( String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto''''''' = parseEightNoteConOctavaAuto'''''''' <*>
    int

parseEightNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto'''''''' = parseEightNoteConOctavaAuto'''''''''
   <*> identifier

parseEightNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto''''''''' = parseEightNoteConOctavaAuto''''''''''
    <*> int

parseEightNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto'''''''''' = parseEightNoteConOctavaAuto
   ''''''''''' <*> identifier

parseEightNoteConOctavaAuto''''''''''' :: H (String -> Int -> String -> Int
   -> String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto''''''''''' = parseEightNoteConOctavaAuto
   ''''''''''' <*> int

parseEightNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> Int -> String -> [
   Note])
parseEightNoteConOctavaAuto'''''''''''' = parseEightNoteConOctavaAuto
   ''''''''''' <*> identifier
```

```
parseEightNoteConOctavaAuto'''''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseEightNoteConOctavaAuto'''''''''''''' = parseEightNoteConOctavaAuto
   '''''''''''''' <*> int

parseEightNoteConOctavaAuto''''''''''''''' :: H (Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto''''''''''''''' = parseEightNoteConOctavaAuto
   ''''''''''''''' <*> identifier

parseEightNoteConOctavaAuto'''''''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> Int -> String -> [Note])
parseEightNoteConOctavaAuto'''''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 i8 c8 -> [catIntervalo i1
      c1, catIntervalo i2 c2, catIntervalo i3 c3, catIntervalo i4 c4,
     catIntervalo i5 c5, catIntervalo i6 c6, catIntervalo i7 c7,
     catIntervalo i8 c8]


parseSevenNoteConOctavaAuto :: H [Note]
parseSevenNoteConOctavaAuto = parseSevenNoteConOctavaAuto' <*> identifier

parseSevenNoteConOctavaAuto' :: H (String -> [Note])
parseSevenNoteConOctavaAuto' = parseSevenNoteConOctavaAuto'' <*> int

parseSevenNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseSevenNoteConOctavaAuto'' = parseSevenNoteConOctavaAuto''' <*>
   identifier

parseSevenNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseSevenNoteConOctavaAuto''' = parseSevenNoteConOctavaAuto'''' <*> int

parseSevenNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseSevenNoteConOctavaAuto'''' = parseSevenNoteConOctavaAuto''''' <*>
   identifier

parseSevenNoteConOctavaAuto''''' :: H ( String -> Int -> String -> Int ->
   String -> [Note])
parseSevenNoteConOctavaAuto''''' = parseSevenNoteConOctavaAuto'''''' <*> int
```

```
parseSevenNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String ->
   Int -> String ->[Note])
parseSevenNoteConOctavaAuto'''''' = parseSevenNoteConOctavaAuto''''''' <*>
   identifier

parseSevenNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String ->[Note])
parseSevenNoteConOctavaAuto''''''' = parseSevenNoteConOctavaAuto'''''''' <*>
    int

parseSevenNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String ->[Note])
parseSevenNoteConOctavaAuto'''''''' = parseSevenNoteConOctavaAuto'''''''''
   <*> identifier

parseSevenNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> [Note])
parseSevenNoteConOctavaAuto''''''''' = parseSevenNoteConOctavaAuto'''''''''
    <*> int

parseSevenNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int -> String
   -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseSevenNoteConOctavaAuto'''''''''' = parseSevenNoteConOctavaAuto
   ''''''''''' <*> identifier

parseSevenNoteConOctavaAuto''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> [
   Note])
parseSevenNoteConOctavaAuto''''''''''' = parseSevenNoteConOctavaAuto
   ''''''''''' <*> int

parseSevenNoteConOctavaAuto'''''''''''' :: H (Int -> String -> Int -> String
    -> Int -> String -> Int -> String -> Int -> String -> Int -> String -> [
   Note])
parseSevenNoteConOctavaAuto'''''''''''' = parseSevenNoteConOctavaAuto
   '''''''''''' <*> identifier

parseSevenNoteConOctavaAuto''''''''''''' :: H (String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseSevenNoteConOctavaAuto''''''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 i7 c7 -> [catIntervalo i1 c1,
     catIntervalo i2 c2, catIntervalo i3 c3, catIntervalo i4 c4,
     catIntervalo i5 c5, catIntervalo i6 c6, catIntervalo i7 c7]
```

```
parseSixNoteConOctavaAuto :: H [Note]
parseSixNoteConOctavaAuto = parseSixNoteConOctavaAuto' <*> identifier

parseSixNoteConOctavaAuto' :: H (String -> [Note])
parseSixNoteConOctavaAuto' = parseSixNoteConOctavaAuto'' <*> int

parseSixNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseSixNoteConOctavaAuto'' = parseSixNoteConOctavaAuto''' <*> identifier

parseSixNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseSixNoteConOctavaAuto''' = parseSixNoteConOctavaAuto'''' <*> int

parseSixNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [Note
   ])
parseSixNoteConOctavaAuto'''' = parseSixNoteConOctavaAuto''''' <*>
   identifier

parseSixNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseSixNoteConOctavaAuto''''' = parseSixNoteConOctavaAuto'''''' <*> int

parseSixNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String -> Int
   -> String -> [Note])
parseSixNoteConOctavaAuto'''''' = parseSixNoteConOctavaAuto''''''' <*>
   identifier

parseSixNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseSixNoteConOctavaAuto''''''' = parseSixNoteConOctavaAuto'''''''' <*> int

parseSixNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
parseSixNoteConOctavaAuto'''''''' = parseSixNoteConOctavaAuto''''''''' <*>
   identifier

parseSixNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseSixNoteConOctavaAuto''''''''' = parseSixNoteConOctavaAuto'''''''''' <*>
    int

parseSixNoteConOctavaAuto'''''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> Int -> String -> [Note])
```

```
parseSixNoteConOctavaAuto'''''''''' = parseSixNoteConOctavaAuto''''''''''
   <*> identifier

parseSixNoteConOctavaAuto'''''''''' :: H (String -> Int -> String -> Int ->
    String -> Int -> String -> Int -> String -> Int -> String -> [Note])
parseSixNoteConOctavaAuto'''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 i6 c6 -> [catIntervalo i1 c1,
     catIntervalo i2 c2, catIntervalo i3 c3, catIntervalo i4 c4,
     catIntervalo i5 c5, catIntervalo i6 c6]


parseFiveNoteConOctavaAuto :: H [Note]
parseFiveNoteConOctavaAuto = parseFiveNoteConOctavaAuto' <*> identifier

parseFiveNoteConOctavaAuto' :: H (String -> [Note])
parseFiveNoteConOctavaAuto' = parseFiveNoteConOctavaAuto'' <*> int

parseFiveNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseFiveNoteConOctavaAuto'' = parseFiveNoteConOctavaAuto''' <*> identifier

parseFiveNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseFiveNoteConOctavaAuto''' = parseFiveNoteConOctavaAuto'''' <*> int

parseFiveNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [Note
   ])
parseFiveNoteConOctavaAuto'''' = parseFiveNoteConOctavaAuto''''' <*>
   identifier

parseFiveNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseFiveNoteConOctavaAuto''''' = parseFiveNoteConOctavaAuto'''''' <*> int

parseFiveNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String -> Int
    -> String -> [Note])
parseFiveNoteConOctavaAuto'''''' = parseFiveNoteConOctavaAuto''''''' <*>
   identifier

parseFiveNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFiveNoteConOctavaAuto''''''' = parseFiveNoteConOctavaAuto'''''''' <*>
   int

parseFiveNoteConOctavaAuto'''''''' :: H (Int -> String -> Int -> String ->
   Int -> String -> Int -> String -> [Note])
```

```
parseFiveNoteConOctavaAuto'''''''' = parseFiveNoteConOctavaAuto''''''''' <*>
    identifier

parseFiveNoteConOctavaAuto''''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> Int -> String -> [Note])
parseFiveNoteConOctavaAuto''''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 i5 c5 -> [catIntervalo i1 c1, catIntervalo
     i2 c2, catIntervalo i3 c3, catIntervalo i4 c4, catIntervalo i5 c5]


--
parseFourNoteConOctavaAuto :: H [Note]
parseFourNoteConOctavaAuto = parseFourNoteConOctavaAuto' <*> identifier

parseFourNoteConOctavaAuto' :: H (String -> [Note])
parseFourNoteConOctavaAuto' = parseFourNoteConOctavaAuto'' <*> int

parseFourNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseFourNoteConOctavaAuto'' = parseFourNoteConOctavaAuto''' <*> identifier

parseFourNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseFourNoteConOctavaAuto''' = parseFourNoteConOctavaAuto'''' <*> int

parseFourNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [Note
   ])
parseFourNoteConOctavaAuto'''' = parseFourNoteConOctavaAuto''''' <*>
   identifier

parseFourNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseFourNoteConOctavaAuto''''' = parseFourNoteConOctavaAuto'''''' <*> int

parseFourNoteConOctavaAuto'''''' :: H (Int -> String -> Int -> String -> Int
    -> String -> [Note])
parseFourNoteConOctavaAuto'''''' = parseFourNoteConOctavaAuto''''''' <*>
   identifier

parseFourNoteConOctavaAuto''''''' :: H (String -> Int -> String -> Int ->
   String -> Int -> String -> [Note])
parseFourNoteConOctavaAuto''''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 i4 c4 -> [catIntervalo i1 c1, catIntervalo i2 c2,
      catIntervalo i3 c3, catIntervalo i4 c4]
```

```
--
parseThreeNoteConOctavaAuto :: H [Note]
parseThreeNoteConOctavaAuto = parseThreeNoteConOctavaAuto' <*> identifier

parseThreeNoteConOctavaAuto' :: H (String -> [Note])
parseThreeNoteConOctavaAuto' = parseThreeNoteConOctavaAuto'' <*> int

parseThreeNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseThreeNoteConOctavaAuto'' = parseThreeNoteConOctavaAuto''' <*>
   identifier

parseThreeNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseThreeNoteConOctavaAuto''' = parseThreeNoteConOctavaAuto'''' <*> int

parseThreeNoteConOctavaAuto'''' :: H (Int -> String -> Int -> String -> [
   Note])
parseThreeNoteConOctavaAuto'''' = parseThreeNoteConOctavaAuto''''' <*>
   identifier

parseThreeNoteConOctavaAuto''''' :: H (String -> Int -> String -> Int ->
   String -> [Note])
parseThreeNoteConOctavaAuto''''' = do
  i1 <- int
  return $ \c1 i2 c2 i3 c3 -> [catIntervalo i1 c1, catIntervalo i2 c2,
     catIntervalo i3 c3]

--
parseDosNoteConOctavaAuto :: H [Note]
parseDosNoteConOctavaAuto = parseDosNoteConOctavaAuto' <*> identifier

parseDosNoteConOctavaAuto' :: H (String -> [Note])
parseDosNoteConOctavaAuto' = parseDosNoteConOctavaAuto'' <*> int

parseDosNoteConOctavaAuto'' :: H (Int -> String -> [Note])
parseDosNoteConOctavaAuto'' = parseDosNoteConOctavaAuto''' <*> identifier

parseDosNoteConOctavaAuto''' :: H (String -> Int -> String -> [Note])
parseDosNoteConOctavaAuto''' = do
  i1 <- int
  return $ \c1 i2 c2-> [catIntervalo i1 c1, catIntervalo i2 c2]

--
parseUnNoteConOctavaAuto :: H [Note]
parseUnNoteConOctavaAuto = parseUnNoteConOctavaAuto' <*> identifier
```

```
parseUnNoteConOctavaAuto' :: H (String -> [Note])
parseUnNoteConOctavaAuto' = do
  i <- int
  return $ \c -> [catIntervalo i c]

catIntervalo :: Int -> String -> Note
catIntervalo i c = intervalo ((T.unpack $ T.strip $ T.pack (show i)) ++ (T.
   unpack $ T.stripStart $ T.pack c)) 0

-- listaDeStringsANote :: [[String]] -> [Note]
-- listaDeStringsANote xs = listaDeListaStringAListaDeNota xs -- [Note]
listaDeStringsANote :: [[Note]] -> [Note]
listaDeStringsANote xs = concat xs -- [Note]

-- listaDeStringsARhythmicPattern :: RhythmicPattern -> [[String]] ->
   RhythmicPattern
listaDeStringsARhythmicPattern :: RhythmicPattern -> [[Note]] ->
   RhythmicPattern
listaDeStringsARhythmicPattern rs xs = do
  let z = zip xs rs -- [([String], INt)]
  listaDeListaDeStringARhythmicP z --
-- let rPat' = listaDeListaDeStringARhythmicP xs [(1, (realToFrac n) / 4)]
    -- [(String, RhythmicPosition)]

-- listaDeStringsANPattern :: NPattern -> [[String]] -> NPattern
listaDeStringsANPattern :: NPattern -> [[Note]] -> NPattern
listaDeStringsANPattern (NPattern1 ns) xs = do
  let z = zip xs ns -- [([String], Int)]
  NPattern1 $ listaDeListaDeStringAN z --[Int]

-- listaDeListaDeStringARhythmicP :: [([String], RhythmicPosition)] ->
   RhythmicPattern
listaDeListaDeStringARhythmicP :: [([Note], RhythmicPosition)] ->
   RhythmicPattern
listaDeListaDeStringARhythmicP xs = do
  let a = concat $ fmap (\x -> listaDeStringARhythmicP x) xs
  fmap snd a

-- listaDeStringARhythmicP :: ([String], RhythmicPosition) -> [(String,
   RhythmicPosition)] -- e.g. [((0,1),"f")]
listaDeStringARhythmicP :: ([Note], RhythmicPosition) -> [(Note,
   RhythmicPosition)] -- e.g. [((0,1),"f")]
listaDeStringARhythmicP (xs, rs) = fmap (\x -> (x, rs)) xs

listaDeListaStringAListaDeNota :: [[String]] -> [Note]
```

```
listaDeListaStringAListaDeNota xs = concat $ fmap listaDeStringAListaDeNota
   xs

listaDeStringAListaDeNota :: [String] -> [Note] -- [intervalo "unisono" 0,
   intervalo "3a" 0, intervalo "5a" 0]
listaDeStringAListaDeNota xs = fmap stringANote xs

-- listaDeListaDeStringAN :: [([String], Int)] -> [Int]
listaDeListaDeStringAN :: [([Note], Int)] -> [Int]
listaDeListaDeStringAN xs = do
  let a = concat $ fmap (\x -> listaDeStringAN x) xs
  fmap snd a

listaDeStringAN :: ([Note], Int) -> [(Note, Int)]-- e.g. [("f", 0)]
listaDeStringAN (xs, n) = fmap (\x -> (x, n)) xs

stringANote :: String -> Note
stringANote s = intervalo s 0

--
praseListaDeListaStringAListaDeAcordes :: H [[Note]]
praseListaDeListaStringAListaDeAcordes = list parseNoteList

-- acompanamiento 2 ["f" "3a" "5a", ]
-- parseStringsAListaDeNotes :: H [String]
parseNoteList :: H [Note]
parseNoteList = parseStringsAListaDeNotes
            <|> parseNotesNoQuotes

parseStringsAListaDeNotes :: H [Note]
parseStringsAListaDeNotes = parseUnStringAListadeNotas
                        <|> parseDosStringsAListadeNotas
                        <|> parseTresStringsAListadeNotas
                        <|> parseCuatroStringsAListadeNotas
                        <|> parseCincoStringsAListadeNotas
                        <|> parseSeisStringsAListadeNotas
                        <|> parseSieteStringsAListadeNotas
                        <|> parseOchoStringsAListadeNotas
                        <|> parseNueveStringsAListadeNotas
                        <|> parseDiezStringsAListadeNotas
                        <|> parseOnceStringsAListadeNotas
                        <|> parseDoceStringsAListadeNotas
```

```
parseDoceStringsAListadeNotas :: H [Note]
parseDoceStringsAListadeNotas = parseDoceStringsAListadeNotas' <*> parseNote

parseDoceStringsAListadeNotas' :: H (Note -> [Note])
parseDoceStringsAListadeNotas' = parseDoceStringsAListadeNotas'' <*>
   parseNote

parseDoceStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseDoceStringsAListadeNotas'' = parseDoceStringsAListadeNotas''' <*>
   parseNote

parseDoceStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseDoceStringsAListadeNotas''' = parseDoceStringsAListadeNotas'''' <*>
   parseNote

parseDoceStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
parseDoceStringsAListadeNotas'''' = parseDoceStringsAListadeNotas''''' <*>
   parseNote

parseDoceStringsAListadeNotas''''' :: H (Note -> Note -> Note -> Note ->
   Note -> [Note])
parseDoceStringsAListadeNotas''''' = parseDoceStringsAListadeNotas'''''' <*>
    parseNote

parseDoceStringsAListadeNotas'''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> [Note])
parseDoceStringsAListadeNotas'''''' = parseDoceStringsAListadeNotas'''''''
   <*> parseNote

parseDoceStringsAListadeNotas''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> [Note])
parseDoceStringsAListadeNotas''''''' = parseDoceStringsAListadeNotas''''''''
    <*> parseNote

parseDoceStringsAListadeNotas'''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> Note -> [Note])
parseDoceStringsAListadeNotas'''''''' = parseDoceStringsAListadeNotas
   ''''''''' <*> parseNote

parseDoceStringsAListadeNotas''''''''' :: H (Note -> Note -> Note -> Note ->
    Note -> Note -> Note -> Note -> Note -> [Note])
parseDoceStringsAListadeNotas''''''''' = parseDoceStringsAListadeNotas
   '''''''''' <*> parseNote
```

```
parseDoceStringsAListadeNotas'''''''''' :: H (Note -> Note -> Note -> Note
   -> Note -> Note -> Note -> Note -> Note -> [Note])
parseDoceStringsAListadeNotas'''''''''' = parseDoceStringsAListadeNotas
   ''''''''''' <*> parseNote

parseDoceStringsAListadeNotas''''''''''' :: H (Note -> Note -> Note -> Note
   -> Note -> Note -> Note -> Note -> Note -> Note -> [Note])
parseDoceStringsAListadeNotas''''''''''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 -> stringsAListadeDoceNotas
     s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12

--
parseOnceStringsAListadeNotas :: H [Note]
parseOnceStringsAListadeNotas = parseOnceStringsAListadeNotas' <*> parseNote

parseOnceStringsAListadeNotas' :: H (Note -> [Note])
parseOnceStringsAListadeNotas' = parseOnceStringsAListadeNotas'' <*>
   parseNote

parseOnceStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseOnceStringsAListadeNotas'' = parseOnceStringsAListadeNotas''' <*>
   parseNote

parseOnceStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseOnceStringsAListadeNotas''' = parseOnceStringsAListadeNotas'''' <*>
   parseNote

parseOnceStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
parseOnceStringsAListadeNotas'''' = parseOnceStringsAListadeNotas''''' <*>
   parseNote

parseOnceStringsAListadeNotas''''' :: H (Note -> Note -> Note -> Note ->
   Note -> [Note])
parseOnceStringsAListadeNotas''''' = parseOnceStringsAListadeNotas'''''' <*>
    parseNote

parseOnceStringsAListadeNotas'''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> [Note])
parseOnceStringsAListadeNotas'''''' = parseOnceStringsAListadeNotas'''''''
   <*> parseNote

parseOnceStringsAListadeNotas''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> [Note])
```

```
parseOnceStringsAListadeNotas'''''' = parseOnceStringsAListadeNotas''''''''
    <*> parseNote

parseOnceStringsAListadeNotas'''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> Note -> [Note])
parseOnceStringsAListadeNotas'''''''' = parseOnceStringsAListadeNotas
   ''''''''' <*> parseNote

parseOnceStringsAListadeNotas''''''''' :: H (Note -> Note -> Note -> Note ->
    Note -> Note -> Note -> Note -> Note -> [Note])
parseOnceStringsAListadeNotas''''''''' = parseOnceStringsAListadeNotas
   '''''''''' <*> parseNote

parseOnceStringsAListadeNotas'''''''''' :: H (Note -> Note -> Note -> Note
   -> Note -> Note -> Note -> Note -> Note -> Note -> [Note])
parseOnceStringsAListadeNotas'''''''''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 -> stringsAListadeOnceNotas s1
     s2 s3 s4 s5 s6 s7 s8 s9 s10 s11

  --

parseDiezStringsAListadeNotas :: H [Note]
parseDiezStringsAListadeNotas = parseDiezStringsAListadeNotas' <*> parseNote

parseDiezStringsAListadeNotas' :: H (Note -> [Note])
parseDiezStringsAListadeNotas' = parseDiezStringsAListadeNotas'' <*>
   parseNote

parseDiezStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseDiezStringsAListadeNotas'' = parseDiezStringsAListadeNotas''' <*>
   parseNote

parseDiezStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseDiezStringsAListadeNotas''' = parseDiezStringsAListadeNotas'''' <*>
   parseNote

parseDiezStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
parseDiezStringsAListadeNotas'''' = parseDiezStringsAListadeNotas''''' <*>
   parseNote

parseDiezStringsAListadeNotas''''' :: H (Note -> Note -> Note -> Note ->
   Note -> [Note])
```

```
parseDiezStringsAListadeNotas''''' = parseDiezStringsAListadeNotas'''''' <*>
    parseNote

parseDiezStringsAListadeNotas'''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> [Note])
parseDiezStringsAListadeNotas'''''' = parseDiezStringsAListadeNotas'''''''
   <*> parseNote

parseDiezStringsAListadeNotas''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> [Note])
parseDiezStringsAListadeNotas''''''' = parseDiezStringsAListadeNotas''''''''
    <*> parseNote

parseDiezStringsAListadeNotas'''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> Note -> [Note])
parseDiezStringsAListadeNotas'''''''' = parseDiezStringsAListadeNotas
   ''''''''' <*> parseNote

parseDiezStringsAListadeNotas''''''''' :: H (Note -> Note -> Note -> Note ->
    Note -> Note -> Note -> Note -> Note -> [Note])
parseDiezStringsAListadeNotas''''''''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 s6 s7 s8 s9 s10 -> stringsAListadeDiezNotas s1 s2 s3
      s4 s5 s6 s7 s8 s9 s10

--
parseNueveStringsAListadeNotas :: H [Note]
parseNueveStringsAListadeNotas = parseNueveStringsAListadeNotas' <*>
   parseNote

parseNueveStringsAListadeNotas' :: H (Note -> [Note])
parseNueveStringsAListadeNotas' = parseNueveStringsAListadeNotas'' <*>
   parseNote

parseNueveStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseNueveStringsAListadeNotas'' = parseNueveStringsAListadeNotas''' <*>
   parseNote

parseNueveStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseNueveStringsAListadeNotas''' = parseNueveStringsAListadeNotas'''' <*>
   parseNote

parseNueveStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
```

```
parseNueveStringsAListadeNotas'''' = parseNueveStringsAListadeNotas''''' <*>
    parseNote

parseNueveStringsAListadeNotas''''' :: H (Note -> Note -> Note -> Note ->
   Note -> [Note])
parseNueveStringsAListadeNotas''''' = parseNueveStringsAListadeNotas'''''''
   <*> parseNote

parseNueveStringsAListadeNotas'''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> [Note])
parseNueveStringsAListadeNotas'''''' = parseNueveStringsAListadeNotas''''''''
    <*> parseNote

parseNueveStringsAListadeNotas''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> [Note])
parseNueveStringsAListadeNotas''''''' = parseNueveStringsAListadeNotas
   '''''''' <*> parseNote

parseNueveStringsAListadeNotas'''''''' :: H (Note -> Note -> Note -> Note ->
    Note -> Note -> Note -> Note -> [Note])
parseNueveStringsAListadeNotas'''''''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 s6 s7 s8 s9 -> stringsAListadeNueveNotas s1 s2 s3 s4
      s5 s6 s7 s8 s9

--

parseOchoStringsAListadeNotas :: H [Note]
parseOchoStringsAListadeNotas = parseOchoStringsAListadeNotas' <*> parseNote

parseOchoStringsAListadeNotas' :: H (Note -> [Note])
parseOchoStringsAListadeNotas' = parseOchoStringsAListadeNotas'' <*>
   parseNote

parseOchoStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseOchoStringsAListadeNotas'' = parseOchoStringsAListadeNotas''' <*>
   parseNote

parseOchoStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseOchoStringsAListadeNotas''' = parseOchoStringsAListadeNotas'''' <*>
   parseNote

parseOchoStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
```

```
parseOchoStringsAListadeNotas'''' = parseOchoStringsAListadeNotas''''' <*>
   parseNote

parseOchoStringsAListadeNotas''''' :: H (Note -> Note -> Note -> Note ->
   Note -> [Note])
parseOchoStringsAListadeNotas''''' = parseOchoStringsAListadeNotas'''''' <*>
    parseNote

parseOchoStringsAListadeNotas'''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> [Note])
parseOchoStringsAListadeNotas'''''' = parseOchoStringsAListadeNotas'''''''
   <*> parseNote

parseOchoStringsAListadeNotas''''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> Note ->[Note])
parseOchoStringsAListadeNotas''''''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 s6 s7 s8 -> stringsAListadeOchoNotas s1 s2 s3 s4 s5
     s6 s7 s8

--

parseSieteStringsAListadeNotas :: H [Note]
parseSieteStringsAListadeNotas = parseSieteStringsAListadeNotas' <*>
   parseNote

parseSieteStringsAListadeNotas' :: H (Note -> [Note])
parseSieteStringsAListadeNotas' = parseSieteStringsAListadeNotas'' <*>
   parseNote

parseSieteStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseSieteStringsAListadeNotas'' = parseSieteStringsAListadeNotas''' <*>
   parseNote

parseSieteStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseSieteStringsAListadeNotas''' = parseSieteStringsAListadeNotas'''' <*>
   parseNote

parseSieteStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
parseSieteStringsAListadeNotas'''' = parseSieteStringsAListadeNotas''''' <*>
    parseNote

parseSieteStringsAListadeNotas''''' :: H (Note -> Note -> Note -> Note ->
   Note -> [Note])
```

```
parseSieteStringsAListadeNotas''''' = parseSieteStringsAListadeNotas''''''
   <*> parseNote

parseSieteStringsAListadeNotas'''''' :: H (Note -> Note -> Note -> Note ->
   Note -> Note -> [Note])
parseSieteStringsAListadeNotas'''''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 s6 s7 -> stringsAListadeSieteNotas s1 s2 s3 s4 s5 s6
      s7

--
parseSeisStringsAListadeNotas :: H [Note]
parseSeisStringsAListadeNotas = parseSeisStringsAListadeNotas' <*> parseNote

parseSeisStringsAListadeNotas' :: H (Note -> [Note])
parseSeisStringsAListadeNotas' = parseSeisStringsAListadeNotas'' <*>
   parseNote

parseSeisStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseSeisStringsAListadeNotas'' = parseSeisStringsAListadeNotas''' <*>
   parseNote

parseSeisStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseSeisStringsAListadeNotas''' = parseSeisStringsAListadeNotas'''' <*>
   parseNote

parseSeisStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
parseSeisStringsAListadeNotas'''' = parseSeisStringsAListadeNotas''''' <*>
   parseNote

parseSeisStringsAListadeNotas''''' :: H (Note -> Note -> Note -> Note ->
   Note -> [Note])
parseSeisStringsAListadeNotas''''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 s6 -> stringsAListadeSeisNotas s1 s2 s3 s4 s5 s6

--

parseCincoStringsAListadeNotas :: H [Note]
parseCincoStringsAListadeNotas = parseCuatroStringsAListadeNotas' <*>
   parseNote

parseCincoStringsAListadeNotas' :: H (Note -> [Note])
```

```
parseCincoStringsAListadeNotas' = parseCincoStringsAListadeNotas'' <*>
   parseNote

parseCincoStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseCincoStringsAListadeNotas'' = parseCincoStringsAListadeNotas''' <*>
   parseNote

parseCincoStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseCincoStringsAListadeNotas''' = parseCincoStringsAListadeNotas'''' <*>
   parseNote

parseCincoStringsAListadeNotas'''' :: H (Note -> Note -> Note -> Note -> [
   Note])
parseCincoStringsAListadeNotas'''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 s5 -> stringsAListadeCincoNotas s1 s2 s3 s4 s5
  --

parseCuatroStringsAListadeNotas :: H [Note]
parseCuatroStringsAListadeNotas = parseCuatroStringsAListadeNotas' <*>
   parseNote

parseCuatroStringsAListadeNotas' :: H (Note -> [Note])
parseCuatroStringsAListadeNotas' = parseCuatroStringsAListadeNotas'' <*>
   parseNote

parseCuatroStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseCuatroStringsAListadeNotas'' = parseCuatroStringsAListadeNotas''' <*>
   parseNote

parseCuatroStringsAListadeNotas''' :: H (Note -> Note -> Note -> [Note])
parseCuatroStringsAListadeNotas''' = do
  s1 <- parseNote
  return $ \s2 s3 s4 -> stringsAListadeCuatroNotas s1 s2 s3 s4

--
parseTresStringsAListadeNotas :: H [Note]
parseTresStringsAListadeNotas = parseTresStringsAListadeNotas' <*> parseNote

parseTresStringsAListadeNotas' :: H (Note -> [Note])
parseTresStringsAListadeNotas' = parseTresStringsAListadeNotas'' <*>
   parseNote

parseTresStringsAListadeNotas'' :: H (Note -> Note -> [Note])
parseTresStringsAListadeNotas'' = do
```

```
  s1 <- parseNote
  return $ \s2 s3 -> stringsAListadeTresNotas s1 s2 s3

--
parseDosStringsAListadeNotas :: H [Note]
parseDosStringsAListadeNotas = parseDosStringsAListadeNotas' <*> parseNote

parseDosStringsAListadeNotas' :: H (Note -> [Note])
parseDosStringsAListadeNotas' = do
  s1 <- parseNote
  return $ \s2 -> stringsAListadeDosNotas s1 s2

--
parseUnStringAListadeNotas :: H [Note]
parseUnStringAListadeNotas = do
  -- s1 <- string
  s1 <- parseNote -- Note
  return $ stringAListadeUnaNota s1
--
-- helper funcs para acompanamiento
-- stringAListadeUnaNota :: String -> [String]
stringAListadeUnaNota :: Note -> [Note]
stringAListadeUnaNota s1 = [s1]

stringsAListadeDosNotas :: Note -> Note -> [Note]
stringsAListadeDosNotas s1 s2 = [s1, s2]

stringsAListadeTresNotas :: Note -> Note -> Note -> [Note]
stringsAListadeTresNotas s1 s2 s3 = [s1, s2, s3]

stringsAListadeCuatroNotas :: Note -> Note -> Note -> Note -> [Note]
stringsAListadeCuatroNotas s1 s2 s3 s4 = [s1, s2, s3, s4]

stringsAListadeCincoNotas :: Note -> Note -> Note -> Note -> Note -> [Note]
stringsAListadeCincoNotas s1 s2 s3 s4 s5 = [s1, s2, s3, s4, s5]

stringsAListadeSeisNotas :: Note -> Note -> Note -> Note -> Note -> Note ->
   [Note]
stringsAListadeSeisNotas s1 s2 s3 s4 s5 s6 = [s1, s2, s3, s4, s5, s6]

stringsAListadeSieteNotas :: Note -> Note -> Note -> Note -> Note -> Note ->
    Note -> [Note]
stringsAListadeSieteNotas s1 s2 s3 s4 s5 s6 s7 = [s1, s2, s3, s4, s5, s6, s7
   ]
```

```
stringsAListadeOchoNotas :: Note -> Note -> Note -> Note -> Note -> Note ->
   Note -> Note -> [Note]
stringsAListadeOchoNotas s1 s2 s3 s4 s5 s6 s7 s8 = [s1, s2, s3, s4, s5, s6,
   s7, s8]

stringsAListadeNueveNotas :: Note -> Note -> Note -> Note -> Note -> Note ->
    Note -> Note -> Note -> [Note]
stringsAListadeNueveNotas s1 s2 s3 s4 s5 s6 s7 s8 s9 = [s1, s2, s3, s4, s5,
   s6, s7, s8, s9]

stringsAListadeDiezNotas :: Note -> Note -> Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> Note -> [Note]
stringsAListadeDiezNotas s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 = [s1, s2, s3, s4,
   s5, s6, s7, s8, s9, s10]

stringsAListadeOnceNotas :: Note -> Note -> Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> Note -> Note -> [Note]
stringsAListadeOnceNotas s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 = [s1, s2, s3,
   s4, s5, s6, s7, s8, s9, s10, s11]

stringsAListadeDoceNotas :: Note -> Note -> Note -> Note -> Note -> Note ->
   Note -> Note -> Note -> Note -> Note -> Note -> [Note]
stringsAListadeDoceNotas s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 = [s1, s2,
   s3, s4, s5, s6, s7, s8, s9, s10, s11, s12]

-- [60 64 67] [59 62 67]
-- ["f" "3a" "5a", "3a" (-1) "5a" (-1) "f" (-1)]
--
-- helper funcs para acompanamiento con octava. e.g. acompanamiento 2 ["3a
   " "5a" ]
-- ?

parseAlternar :: H Layer
parseAlternar = parseAlternar' <*> parseLayer

parseAlternar' :: H (Layer -> Layer)
parseAlternar' = parseAlternar'' <*> parseLayerToLayerFunc

parseAlternar'' :: H ((Layer -> Layer) -> Layer -> Layer)
parseAlternar'' = parseAlternar''' <*> int

parseAlternar''' :: H (Int -> (Layer -> Layer) -> Layer -> Layer)
parseAlternar''' = alternar <$ reserved "alternar"

parseLayerToLayerFunc :: H (Layer -> Layer)
```

```
parseLayerToLayerFunc = parseSeleccionarEstilo'
                      <|> parseSeleccionarSample'
                      <|> parseSeleccionarSamples'
                      <|> parseTonicaYquinta'
                      <|> parseTonicaYquinta2'
                      <|> parseTonicaQoctava'
                      <|> parseTonicaQtercera'
                      <|> parseCambiarNota'
                      <|> parseCambiarNotas'
                      <|> parseCambiarRitmoAuto'
                      <|> parseCambiarRitmosAuto'
                      <|> parseCambiarRitmo'
                      <|> parseCambiarRitmos'
                      <|> parseCambiarIntervalo'
                      <|> parseCambiarIntervalos'
                      <|> parseCambiarIntervaloConOctava'
                      <|> parseCambiarIntervalosConOctava'
                      <|> parseCambiarIntervaloDouble'
                      <|> parseCambiarIntervaloDoubleConOctava'
                      <|> parseCambiarIntervalosDouble'
                      <|> parseCambiarIntervalosDoubleConOctava'
                      <|> parsePreset'
                      <|> parseAlternar'
                      <|> parseCambiarGain'
                      <|> parseCambiarPaneo'
                      <|> parsePunteo'
                      <|> parsePunteos'
                      <|> parseTumbao'
                      <|> parseaTumbaoBajoVoicingSel'
                      <|> parseaTumbaoBajoVoicingYRitmoSel'
                      <|> parseaTumbaoBajoVoicingsYRitmoSel'
                      <|> parseTumbaoCongasGolpesSel'
                      <|> parseTumbaoCongasGolpesYRitmoSel'
                      -- <|> parseTumbaoCongasListaDeGolpesSel'
                      <|> parseTumbaoCongasListaDeGolpesYRitmoSel'
                      <|> parseacompanamiento'
                      <|> parseacompanamientos'
                      <|> parseAcompanamientoConVoicingSel'
                      <|> parseAcompanamientosConVoicingSel'


alternar :: Int -> (Layer -> Layer) -> Layer -> Layer
alternar 0 f x = Layer { getEvents = updatedEv, style = style x}
  where
    f0 = getEvents x
```

```
    s0 = style x
    updatedEv gm s t iw ew = f0 gm s0 t iw ew

alternar 1 f x = Layer { getEvents = updatedEv, style = style (f x)}
  where
    f1 = getEvents (f x)
    s1 = style (f x)
    updatedEv gm s t iw ew = f1 gm s1 t iw ew

alternar n f x = Layer { getEvents = updatedEv, style = style (f x)}
  where
    f0 = getEvents x -- lista orginal de evs
    f1 = getEvents (f x) -- lista nueva de evs
    s0 = style x
    s1 = style (f x)

    updatedEv gm s t iw ew = liftM ordernarEv $ liftM concat $ liftM2 (++)
       es0 es1
      where
        iw' = timeToCount t iw -- Rational
        ew' = timeToCount t ew --

        (w0, w1) = alternarWindows'' (toRational n) (toRational $ compas gm)
           iw' ew' -- find all the windows that comply with certain
           condition within the provided window

        es0 = mapM (\(i,e) -> f0 gm s0 t (countToTime t i) (countToTime t e))
            w0 -- State LayerState [[Events]]
        es1 = mapM (\(i,e) -> f1 gm s1 t (countToTime t i) (countToTime t e))
            w1

data Event' = Event' (UTCTime, Map.Map T.Text H.Datum) deriving (Show, Eq)

instance Ord Event' where
   Event' (u1, _) `compare` Event' (u2, _) = u1 `compare` u2

ordernarEv :: [(UTCTime, Map.Map T.Text H.Datum)] -> [(UTCTime, Map.Map T.
    Text H.Datum)]
ordernarEv evs = do
  let evsToEv = List.sort $ fmap (\e -> Event' e) evs
  fmap (\e -> evToEv e) evsToEv

evToEv :: Event'-> (UTCTime, Map.Map T.Text H.Datum)
evToEv (Event' (utc, d)) = (utc, d)
```

```
-- myEvent' :: Event'
-- myEvent' = Event' ((mytime 0.75), Map.fromList [("s", H.string "test")
   ])
-- --
-- myEvents :: [Event']
-- myEvents = [Event' ((mytime 1.0), Map.fromList [("s", H.string "tres")
   ]), Event' ((mytime 0.75), Map.fromList [("s", H.string "dos")]), Event
   ' ((mytime 0.25), Map.fromList [("s", H.string "uno")])]

alternarWindows n compas iw ew = do
  let n' = n * compas
  let iw' = realToFrac $ floor iw
  let ew' = realToFrac $ floor ew
  let lista = [iw', iw' + compas .. ew']
  let lista' = fmap (\e -> (e, e + (1 * compas))) lista
  let lista'' = drop 1 $ init lista'
  let lista''' = firstItem : lista'' ++ lastItem
      firstItem | (realToFrac $ floor iw) == (realToFrac $ floor ew) = (
         realToFrac iw, realToFrac ew)
               | otherwise = (realToFrac iw, (realToFrac $ floor iw) + (1 *
                  compas))

      lastItem | (realToFrac $ floor ew) == (realToFrac $ floor iw) = []
              | (realToFrac ew) > (realToFrac $ floor ew) = [(realToFrac $
                 floor ew, realToFrac ew)]
              | otherwise = []
  let x = catMaybes $ fmap (\(x, y) -> if (mod' (realToFrac x) (realToFrac n
     ')) /= ((realToFrac n') - compas) then Just (x , y) else Nothing) lista
     '''
  let fx = catMaybes $ fmap (\(x,y) -> if (mod' (realToFrac x) (realToFrac n
     ')) == ((realToFrac n') - compas) then Just (x , y) else Nothing )
     lista'''
  (x, fx)

-- alternarWindows' :: Rational -> Rational -> Rational -> Rational -> ([(
   Rational, Rational)], [(Rational, Rational)])
alternarWindows' n compas iw ew = do
  let n' = n * compas
  let iw' = realToFrac $ floor iw
  let ew' = realToFrac $ floor ew
  let lista = [iw', iw' + compas .. ew']
  let lista' = fmap (\e -> (e, e + (1 * compas))) lista
  let lista'' = drop 1 $ init lista'
  let lista''' = firstItem ++ lista'' ++ lastItem
```

```
    firstItem | (realToFrac $ floor iw) == (realToFrac $ floor ew) = [(
       realToFrac iw, realToFrac ew)]
              | (realToFrac iw) == ((realToFrac $ floor iw) + (1 * compas))
                 = []
              | otherwise = [(realToFrac iw, (realToFrac $ floor iw) + (1 *
                 compas))]

    lastItem | (realToFrac $ floor ew) == (realToFrac $ floor iw) = []
             | (realToFrac ew) > (realToFrac $ floor ew) = [(realToFrac $
                floor ew, realToFrac ew)]
             | otherwise = []
  let x = catMaybes $ fmap (\(x, y) -> if (mod' (realToFrac x) (realToFrac n
     )) /= ((realToFrac n') - compas) then Just (x , y) else Nothing) lista
     '''
  let fx = catMaybes $ fmap (\(x,y) -> if (mod' (realToFrac x) (realToFrac n
     )) == ((realToFrac n') - compas) then Just (x , y) else Nothing )
     lista'''
  (x, fx)

-- alternarWindows'' :: Double -> Double -> Double -> Double -> ([(Double,
    Double)], [(Double, Double)])
alternarWindows'' n compas iw ew = do
  let n' = n * compas
  let iw' = realToFrac $ floor iw
  let ew' = realToFrac $ ceiling ew
  let lista = [iw', (iw' + compas).. ew']
  let lowerFilteredList = Prelude.filter ((<=) iw) lista
  let upwardsFilteredList = Prelude.filter ((>=) ew) lowerFilteredList
  let lista' = fmap (\e -> (e, e + compas)) upwardsFilteredList
  let confirmHead |((length lista') > 0) && (iw < (fst $ head lista')) = ((
     fst $ head lista') - ((fst $ head lista') - iw) , fst $ head lista') :
     lista'
                  | ((length lista') > 0) && (iw == (fst $ head lista')) =
                     lista'
                  | ((length lista') == 0) = [(iw, ew)]
  let confirmLast | (length confirmHead == 1) = [(fst $ head confirmHead, ew
     )]
                  | (length confirmHead > 1) && (ew == (fst $ last confirmHead
                     )) = init confirmHead
                  | (length confirmHead > 1) && (ew > (fst $ last confirmHead)
                     ) = init confirmHead ++ [(snd $ last $ init confirmHead,
                     ew)]
  let x = catMaybes $ fmap (\(a, b) -> compararX (a, b) compas n')
     confirmLast
```

```
  let fx = catMaybes $ fmap (\(a, b) -> compararFx (a, b) compas n')
     confirmLast
  (x, fx)

-- compararX :: (Double, Double) -> Double -> Double -> Maybe (Double,
   Double)
compararX (a,b) compas n
  | (a > 0) && (a < (realToFrac $ floor a) + compas) = if (mod' (realToFrac
     $ floor a) (realToFrac n)) /= ((realToFrac n) - compas) then Just (a ,
     b) else Nothing
  | (a > (realToFrac $ floor a) + compas) && (a < (realToFrac $ ceiling a))
     = if (mod' ((realToFrac $ floor a) + compas) (realToFrac n)) /= ((
     realToFrac n) - compas) then Just (a , b) else Nothing
  | otherwise = if (mod' (realToFrac a) (realToFrac n)) /= ((realToFrac n) -
      compas) then Just (a , b) else Nothing

-- compararFx :: (Double, Double) -> Double -> Double -> Maybe (Double,
   Double)
compararFx (a,b) compas n
  | (a > 0) && (a < (realToFrac $ floor a) + compas) = if (mod' (realToFrac
     $ floor a) (realToFrac n)) == ((realToFrac n) - compas) then Just (a ,
     b) else Nothing
  | (a > (realToFrac $ floor a) + compas) && (a < (realToFrac $ ceiling a))
     = if (mod' ((realToFrac $ floor a) + compas) (realToFrac n)) == ((
     realToFrac n) - compas) then Just (a , b) else Nothing
  | otherwise = if (mod' (realToFrac a) (realToFrac n)) == ((realToFrac n) -
      compas) then Just (a , b) else Nothing

-- test funcs
-- let l = alternar 2 (tonicaYquinta) (seleccionarEstilo cumbia bajo)
-- (runState (getEvents l testgmm cumbia mytempo (mytime 0) (mytime 1))
   emptyLayerState)

-- helper functions

stringList :: H [String]
stringList = list string

rationalList :: H [Rational]
rationalList = list $ rationalOrInteger

intList :: H [Int]
intList = list $ fromIntegral <$> integer

int :: H Int
```

```
int = fromIntegral <$> integer

parseNPattern1 :: H NPattern
parseNPattern1 = do
  ix <- intList
  return $ NPattern1 ix

double :: H Double
double = fromRational <$> rationalOrInteger

doubleList :: H [Double]
doubleList = list double

--
parseListasDeListasDeAtaques :: H [[Rational]]
parseListasDeListasDeAtaques = list parseAtaquesAListaDeAtaques

parseAtaquesAListaDeAtaques :: H [Rational]
parseAtaquesAListaDeAtaques = parseUnAtaqueAListDeAtaques
              <|> parseDosAtaquesAListDeAtaques
              <|> parseTresAtaquesAListDeAtaques
              <|> parseCuatroAtaquesAListDeAtaques
              <|> parseCincoAtaquesAListDeAtaques
              <|> parseSeisAtaquesAListDeAtaques
              <|> parseSieteAtaquesAListDeAtaques
              <|> parseOchoAtaquesAListDeAtaques
              <|> parseNueveAtaquesAListDeAtaques
              <|> parseDiezAtaquesAListDeAtaques
              <|> parseOnceAtaquesAListDeAtaques
              <|> parseDoceAtaquesAListDeAtaques
              <|> parseTreceAtaquesAListDeAtaques
              <|> parseCatorceAtaquesAListDeAtaques
              <|> parseQuinceAtaquesAListDeAtaques
              <|> parseDieciseisAtaquesAListDeAtaques

parseDieciseisAtaquesAListDeAtaques :: H [Rational]
parseDieciseisAtaquesAListDeAtaques = parseDieciseisAtaquesAListDeAtaques'
   <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques' = parseDieciseisAtaquesAListDeAtaques''
    <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [
   Rational])
```

```
parseDieciseisAtaquesAListDeAtaques'' = parseDieciseisAtaquesAListDeAtaques
   ''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques''' :: H (Rational -> Rational ->
   Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques''' = parseDieciseisAtaquesAListDeAtaques
   '''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'''' :: H (Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques'''' =
   parseDieciseisAtaquesAListDeAtaques''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques''''' =
   parseDieciseisAtaquesAListDeAtaques'''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques'''''' =
   parseDieciseisAtaquesAListDeAtaques''''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques''''''' =
   parseDieciseisAtaquesAListDeAtaques'''''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques'''''''' =
   parseDieciseisAtaquesAListDeAtaques''''''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques''''''''' =
   parseDieciseisAtaquesAListDeAtaques'''''''''' <*> rationalOrInteger
```

```
parseDieciseisAtaquesAListDeAtaques'''''''''' :: H (Rational -> Rational ->
    Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques'''''''''' =
   parseDieciseisAtaquesAListDeAtaques'''''''''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'''''''''''' :: H (Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques'''''''''''' =
   parseDieciseisAtaquesAListDeAtaques'''''''''''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'''''''''''''' :: H (Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> [Rational
   ])
parseDieciseisAtaquesAListDeAtaques'''''''''''''' =
   parseDieciseisAtaquesAListDeAtaques'''''''''''''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'''''''''''''''' :: H (Rational -> Rational
    -> Rational -> Rational -> Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> Rational
   -> [Rational])
parseDieciseisAtaquesAListDeAtaques'''''''''''''''' =
   parseDieciseisAtaquesAListDeAtaques'''''''''''''''''' <*> rationalOrInteger

parseDieciseisAtaquesAListDeAtaques'''''''''''''''''' :: H (Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseDieciseisAtaquesAListDeAtaques'''''''''''''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 n16 ->
     dieciseisAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12
     n13 n14 n15 n16

dieciseisAtaquesAListDeAtaques :: Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational]
dieciseisAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13
   n14 n15 n16 = [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13,
   n14, n15, n16]
```

```
-- ("p" "t" "p" (t "a") ...)
parseQuinceAtaquesAListDeAtaques :: H [Rational]
parseQuinceAtaquesAListDeAtaques = parseQuinceAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseQuinceAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques' = parseQuinceAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseQuinceAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques'' = parseQuinceAtaquesAListDeAtaques''' <*>
    rationalOrInteger

parseQuinceAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational
   -> [Rational])
parseQuinceAtaquesAListDeAtaques''' = parseQuinceAtaquesAListDeAtaques''''
   <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational
   -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques'''' = parseQuinceAtaquesAListDeAtaques'''''
    <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques''''' = parseQuinceAtaquesAListDeAtaques
   '''''' <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques'''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques'''''' = parseQuinceAtaquesAListDeAtaques
   ''''''' <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques''''''' = parseQuinceAtaquesAListDeAtaques
   '''''''' <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques'''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational -> [
   Rational])
parseQuinceAtaquesAListDeAtaques'''''''' = parseQuinceAtaquesAListDeAtaques
   ''''''''' <*> rationalOrInteger
```

```
parseQuinceAtaquesAListDeAtaques''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques''''''''' = parseQuinceAtaquesAListDeAtaques
   '''''''''' <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques'''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques'''''''''' =
   parseQuinceAtaquesAListDeAtaques''''''''''' <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques''''''''''' =
   parseQuinceAtaquesAListDeAtaques'''''''''''' <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques'''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> [Rational])
parseQuinceAtaquesAListDeAtaques'''''''''''' =
   parseQuinceAtaquesAListDeAtaques''''''''''''' <*> rationalOrInteger

parseQuinceAtaquesAListDeAtaques''''''''''''' :: H (Rational -> Rational ->
    Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational -> [
   Rational])
parseQuinceAtaquesAListDeAtaques''''''''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 n15 ->
     quinceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13
     n14 n15

quinceAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> [Rational]
quinceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14
   n15 = [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15]
```

```
-- ("p" "t" "p" (t "a") ...)
parseCatorceAtaquesAListDeAtaques :: H [Rational]
parseCatorceAtaquesAListDeAtaques = parseCatorceAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseCatorceAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques' = parseCatorceAtaquesAListDeAtaques'' <*>
    rationalOrInteger

parseCatorceAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational
   ])
parseCatorceAtaquesAListDeAtaques'' = parseCatorceAtaquesAListDeAtaques'''
   <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational
   -> [Rational])
parseCatorceAtaquesAListDeAtaques''' = parseCatorceAtaquesAListDeAtaques''''
    <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational
    -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques'''' = parseCatorceAtaquesAListDeAtaques
   ''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques''''' = parseCatorceAtaquesAListDeAtaques
   '''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques'''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques'''''' = parseCatorceAtaquesAListDeAtaques
   ''''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques''''''' = parseCatorceAtaquesAListDeAtaques
   '''''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques'''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational -> [
   Rational])
parseCatorceAtaquesAListDeAtaques'''''''' =
   parseCatorceAtaquesAListDeAtaques''''''''' <*> rationalOrInteger
```

```
parseCatorceAtaquesAListDeAtaques'''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques'''''''''' =
   parseCatorceAtaquesAListDeAtaques'''''''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques''''''''''' =
   parseCatorceAtaquesAListDeAtaques''''''''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques'''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques'''''''''''' =
   parseCatorceAtaquesAListDeAtaques'''''''''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques''''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques''''''''''''' =
   parseCatorceAtaquesAListDeAtaques''''''''''''' <*> rationalOrInteger

parseCatorceAtaquesAListDeAtaques'''''''''''''' :: H (Rational -> Rational ->
    Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseCatorceAtaquesAListDeAtaques'''''''''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14 ->
     catorceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13
      n14

catorceAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> [Rational]
catorceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 n14
   = [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14]

-- ("p" "t" "p" (t "a") ...)
parseTreceAtaquesAListDeAtaques :: H [Rational]
parseTreceAtaquesAListDeAtaques = parseTreceAtaquesAListDeAtaques' <*>
   rationalOrInteger
```

```
parseTreceAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseTreceAtaquesAListDeAtaques' = parseTreceAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseTreceAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques'' = parseTreceAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseTreceAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
    [Rational])
parseTreceAtaquesAListDeAtaques''' = parseTreceAtaquesAListDeAtaques'''' <*>
    rationalOrInteger

parseTreceAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational
   -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques'''' = parseTreceAtaquesAListDeAtaques'''''
   <*> rationalOrInteger

parseTreceAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques''''' = parseTreceAtaquesAListDeAtaques''''''
    <*> rationalOrInteger

parseTreceAtaquesAListDeAtaques'''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques'''''' = parseTreceAtaquesAListDeAtaques
   ''''''' <*> rationalOrInteger

parseTreceAtaquesAListDeAtaques''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques''''''' = parseTreceAtaquesAListDeAtaques
   '''''''' <*> rationalOrInteger

parseTreceAtaquesAListDeAtaques'''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational])
parseTreceAtaquesAListDeAtaques'''''''' = parseTreceAtaquesAListDeAtaques
   ''''''''' <*> rationalOrInteger
```

```
parseTreceAtaquesAListDeAtaques'''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques''''''''' = parseTreceAtaquesAListDeAtaques
   '''''''''' <*> rationalOrInteger

parseTreceAtaquesAListDeAtaques''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques'''''''''' = parseTreceAtaquesAListDeAtaques
   ''''''''''' <*> rationalOrInteger

parseTreceAtaquesAListDeAtaques'''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> [Rational])
parseTreceAtaquesAListDeAtaques''''''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 ->
     treceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13

treceAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> [Rational]
treceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 n13 = [n1,
    n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13]

-- ("p" "t" "p" (t "a") ...)
parseDoceAtaquesAListDeAtaques :: H [Rational]
parseDoceAtaquesAListDeAtaques = parseDoceAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseDoceAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseDoceAtaquesAListDeAtaques' = parseDoceAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseDoceAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseDoceAtaquesAListDeAtaques'' = parseDoceAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseDoceAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
   [Rational])
parseDoceAtaquesAListDeAtaques''' = parseDoceAtaquesAListDeAtaques'''' <*>
   rationalOrInteger
```

```
parseDoceAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational ->
    Rational -> [Rational])
parseDoceAtaquesAListDeAtaques'''' = parseDoceAtaquesAListDeAtaques''''' <*>
    rationalOrInteger

parseDoceAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseDoceAtaquesAListDeAtaques''''' = parseDoceAtaquesAListDeAtaques''''''
   <*> rationalOrInteger

parseDoceAtaquesAListDeAtaques'''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> [Rational])
parseDoceAtaquesAListDeAtaques'''''' = parseDoceAtaquesAListDeAtaques'''''''
    <*> rationalOrInteger

parseDoceAtaquesAListDeAtaques''''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseDoceAtaquesAListDeAtaques''''''' = parseDoceAtaquesAListDeAtaques
   '''''''' <*> rationalOrInteger

parseDoceAtaquesAListDeAtaques'''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational -> [
   Rational])
parseDoceAtaquesAListDeAtaques'''''''' = parseDoceAtaquesAListDeAtaques
   ''''''''' <*> rationalOrInteger

parseDoceAtaquesAListDeAtaques''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational])
parseDoceAtaquesAListDeAtaques''''''''' = parseDoceAtaquesAListDeAtaques
   '''''''''' <*> rationalOrInteger

parseDoceAtaquesAListDeAtaques'''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseDoceAtaquesAListDeAtaques'''''''''' = parseDoceAtaquesAListDeAtaques
   ''''''''''' <*> rationalOrInteger

parseDoceAtaquesAListDeAtaques''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> Rational -> [Rational])
parseDoceAtaquesAListDeAtaques''''''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 -> doceAtaquesAListDeAtaques
      n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12
```

```
doceAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational ->
    Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational]
doceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 n12 = [n1, n2,
   n3, n4, n5, n6, n7, n8, n9, n10, n11, n12]

-- ("p" "t" "p" (t "a") ...)
parseOnceAtaquesAListDeAtaques :: H [Rational]
parseOnceAtaquesAListDeAtaques = parseOnceAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseOnceAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseOnceAtaquesAListDeAtaques' = parseOnceAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseOnceAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseOnceAtaquesAListDeAtaques'' = parseOnceAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseOnceAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
   [Rational])
parseOnceAtaquesAListDeAtaques''' = parseOnceAtaquesAListDeAtaques'''' <*>
   rationalOrInteger

parseOnceAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational ->
    Rational -> [Rational])
parseOnceAtaquesAListDeAtaques'''' = parseOnceAtaquesAListDeAtaques''''' <*>
    rationalOrInteger

parseOnceAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseOnceAtaquesAListDeAtaques''''' = parseOnceAtaquesAListDeAtaques'''''' 
   <*> rationalOrInteger

parseOnceAtaquesAListDeAtaques'''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> [Rational])
parseOnceAtaquesAListDeAtaques'''''' = parseOnceAtaquesAListDeAtaques'''''''
    <*> rationalOrInteger

parseOnceAtaquesAListDeAtaques''''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseOnceAtaquesAListDeAtaques''''''' = parseOnceAtaquesAListDeAtaques
   '''''''' <*> rationalOrInteger
```

```
parseOnceAtaquesAListDeAtaques''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational -> [
   Rational])
parseOnceAtaquesAListDeAtaques''''''''' = parseOnceAtaquesAListDeAtaques
   ''''''''' <*> rationalOrInteger

parseOnceAtaquesAListDeAtaques'''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational])
parseOnceAtaquesAListDeAtaques'''''''''' = parseOnceAtaquesAListDeAtaques
   '''''''''' <*> rationalOrInteger

parseOnceAtaquesAListDeAtaques''''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> Rational -> [Rational])
parseOnceAtaquesAListDeAtaques''''''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 -> onceAtaquesAListDeAtaques n1
     n2 n3 n4 n5 n6 n7 n8 n9 n10 n11

onceAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational ->
    Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational]
onceAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11 = [n1, n2, n3,
   n4, n5, n6, n7, n8, n9, n10, n11]

-- ("p" "t" "p" (t "a") ...)
parseDiezAtaquesAListDeAtaques :: H [Rational]
parseDiezAtaquesAListDeAtaques = parseDiezAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseDiezAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseDiezAtaquesAListDeAtaques' = parseDiezAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseDiezAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseDiezAtaquesAListDeAtaques'' = parseDiezAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseDiezAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
   [Rational])
parseDiezAtaquesAListDeAtaques''' = parseDiezAtaquesAListDeAtaques'''' <*>
   rationalOrInteger
```

```
parseDiezAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational ->
    Rational -> [Rational])
parseDiezAtaquesAListDeAtaques'''' = parseDiezAtaquesAListDeAtaques''''' <*>
    rationalOrInteger

parseDiezAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseDiezAtaquesAListDeAtaques''''' = parseDiezAtaquesAListDeAtaques''''''
   <*> rationalOrInteger

parseDiezAtaquesAListDeAtaques'''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> [Rational])
parseDiezAtaquesAListDeAtaques'''''' = parseDiezAtaquesAListDeAtaques'''''''
    <*> rationalOrInteger

parseDiezAtaquesAListDeAtaques''''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseDiezAtaquesAListDeAtaques''''''' = parseDiezAtaquesAListDeAtaques
   '''''''' <*> rationalOrInteger

parseDiezAtaquesAListDeAtaques'''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational -> [
   Rational])
parseDiezAtaquesAListDeAtaques'''''''' = parseDiezAtaquesAListDeAtaques
   ''''''''' <*> rationalOrInteger

parseDiezAtaquesAListDeAtaques''''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   Rational -> [Rational])
parseDiezAtaquesAListDeAtaques''''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 n10 -> diezAtaquesAListDeAtaques n1 n2
     n3 n4 n5 n6 n7 n8 n9 n10

diezAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational ->
    Rational -> Rational -> Rational -> Rational -> Rational -> Rational ->
   [Rational]
diezAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 = [n1, n2, n3, n4,
   n5, n6, n7, n8, n9, n10]

-- ("p" "t" "p" (t "a") ...)
parseNueveAtaquesAListDeAtaques :: H [Rational]
parseNueveAtaquesAListDeAtaques = parseNueveAtaquesAListDeAtaques' <*>
   rationalOrInteger
```

```
parseNueveAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseNueveAtaquesAListDeAtaques' = parseNueveAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseNueveAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseNueveAtaquesAListDeAtaques'' = parseNueveAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseNueveAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
    [Rational])
parseNueveAtaquesAListDeAtaques''' = parseNueveAtaquesAListDeAtaques'''' <*>
    rationalOrInteger

parseNueveAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational
   -> Rational -> [Rational])
parseNueveAtaquesAListDeAtaques'''' = parseNueveAtaquesAListDeAtaques'''''
   <*> rationalOrInteger

parseNueveAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseNueveAtaquesAListDeAtaques''''' = parseNueveAtaquesAListDeAtaques''''''
    <*> rationalOrInteger

parseNueveAtaquesAListDeAtaques'''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> [Rational])
parseNueveAtaquesAListDeAtaques'''''' = parseNueveAtaquesAListDeAtaques
   ''''''' <*> rationalOrInteger

parseNueveAtaquesAListDeAtaques''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseNueveAtaquesAListDeAtaques''''''' = parseNueveAtaquesAListDeAtaques
   '''''''' <*> rationalOrInteger

parseNueveAtaquesAListDeAtaques'''''''' :: H (Rational -> Rational ->
   Rational -> Rational -> Rational -> Rational -> Rational -> Rational -> [
   Rational])
parseNueveAtaquesAListDeAtaques'''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 n9 -> nueveAtaquesAListDeAtaques n1 n2 n3
     n4 n5 n6 n7 n8 n9

nueveAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> Rational -> Rational -> [Rational]
nueveAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 n9 = [n1, n2, n3, n4, n5,
    n6, n7, n8, n9]
```

```
-- ("p" "t" "p" (t "a") ...)
parseOchoAtaquesAListDeAtaques :: H [Rational]
parseOchoAtaquesAListDeAtaques = parseOchoAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseOchoAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseOchoAtaquesAListDeAtaques' = parseOchoAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseOchoAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseOchoAtaquesAListDeAtaques'' = parseOchoAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseOchoAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
   [Rational])
parseOchoAtaquesAListDeAtaques''' = parseOchoAtaquesAListDeAtaques'''' <*>
   rationalOrInteger

parseOchoAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational ->
    Rational -> [Rational])
parseOchoAtaquesAListDeAtaques'''' = parseOchoAtaquesAListDeAtaques''''' <*>
    rationalOrInteger

parseOchoAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseOchoAtaquesAListDeAtaques''''' = parseOchoAtaquesAListDeAtaques'''''' 
   <*> rationalOrInteger

parseOchoAtaquesAListDeAtaques'''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> [Rational])
parseOchoAtaquesAListDeAtaques'''''' = parseOchoAtaquesAListDeAtaques'''''''
    <*> rationalOrInteger

parseOchoAtaquesAListDeAtaques''''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> Rational -> [Rational])
parseOchoAtaquesAListDeAtaques''''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 n8 -> ochoAtaquesAListDeAtaques n1 n2 n3 n4 n5
      n6 n7 n8

ochoAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational ->
    Rational -> Rational -> Rational -> Rational -> [Rational]
ochoAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 n8 = [n1, n2, n3, n4, n5, n6,
    n7, n8]
```

```
-- ("p" "t" "p" (t "a") ...)
parseSieteAtaquesAListDeAtaques :: H [Rational]
parseSieteAtaquesAListDeAtaques = parseSieteAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseSieteAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseSieteAtaquesAListDeAtaques' = parseSieteAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseSieteAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseSieteAtaquesAListDeAtaques'' = parseSieteAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseSieteAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
    [Rational])
parseSieteAtaquesAListDeAtaques''' = parseSieteAtaquesAListDeAtaques'''' <*>
    rationalOrInteger

parseSieteAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational
   -> Rational -> [Rational])
parseSieteAtaquesAListDeAtaques'''' = parseSieteAtaquesAListDeAtaques''''' 
   <*> rationalOrInteger

parseSieteAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseSieteAtaquesAListDeAtaques''''' = parseSieteAtaquesAListDeAtaques'''''' 
    <*> rationalOrInteger

parseSieteAtaquesAListDeAtaques'''''' :: H (Rational -> Rational -> Rational
    -> Rational -> Rational -> Rational -> [Rational])
parseSieteAtaquesAListDeAtaques'''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 n7 -> sieteAtaquesAListDeAtaques n1 n2 n3 n4 n5
     n6 n7

sieteAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational
   -> Rational -> Rational -> Rational -> [Rational]
sieteAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 n7 = [n1, n2, n3, n4, n5, n6,
   n7]

-- ("p" "t" "p" (t "a") ...)
parseSeisAtaquesAListDeAtaques :: H [Rational]
parseSeisAtaquesAListDeAtaques = parseSeisAtaquesAListDeAtaques' <*>
   rationalOrInteger
```

```
parseSeisAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseSeisAtaquesAListDeAtaques' = parseSeisAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseSeisAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseSeisAtaquesAListDeAtaques'' = parseSeisAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseSeisAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
   [Rational])
parseSeisAtaquesAListDeAtaques''' = parseSeisAtaquesAListDeAtaques'''' <*>
   rationalOrInteger

parseSeisAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational ->
    Rational -> [Rational])
parseSeisAtaquesAListDeAtaques'''' = parseSeisAtaquesAListDeAtaques''''' <*>
    rationalOrInteger

parseSeisAtaquesAListDeAtaques''''' :: H (Rational -> Rational -> Rational
   -> Rational -> Rational -> [Rational])
parseSeisAtaquesAListDeAtaques''''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 n6 -> seisAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6

seisAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational ->
    Rational -> Rational -> [Rational]
seisAtaquesAListDeAtaques n1 n2 n3 n4 n5 n6 = [n1, n2, n3, n4, n5, n6]

-- ("p" "t" "p" (t "a") ...)
parseCincoAtaquesAListDeAtaques :: H [Rational]
parseCincoAtaquesAListDeAtaques = parseCincoAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseCincoAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseCincoAtaquesAListDeAtaques' = parseCincoAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseCincoAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseCincoAtaquesAListDeAtaques'' = parseCincoAtaquesAListDeAtaques''' <*>
   rationalOrInteger

parseCincoAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational ->
    [Rational])
```

```
parseCincoAtaquesAListDeAtaques''' = parseCincoAtaquesAListDeAtaques'''' <*>
    rationalOrInteger

parseCincoAtaquesAListDeAtaques'''' :: H (Rational -> Rational -> Rational
   -> Rational -> [Rational])
parseCincoAtaquesAListDeAtaques'''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 n5 -> cincoAtaquesAListDeAtaques n1 n2 n3 n4 n5

cincoAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational
   -> Rational -> [Rational]
cincoAtaquesAListDeAtaques n1 n2 n3 n4 n5 = [n1, n2, n3, n4, n5]


-- ("p" "t" "p" $ t "a")
parseCuatroAtaquesAListDeAtaques :: H [Rational]
parseCuatroAtaquesAListDeAtaques = parseCuatroAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseCuatroAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseCuatroAtaquesAListDeAtaques' = parseCuatroAtaquesAListDeAtaques'' <*>
   rationalOrInteger

parseCuatroAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseCuatroAtaquesAListDeAtaques'' = parseCuatroAtaquesAListDeAtaques''' <*>
    rationalOrInteger

parseCuatroAtaquesAListDeAtaques''' :: H (Rational -> Rational -> Rational
   -> [Rational])
parseCuatroAtaquesAListDeAtaques''' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 n4 -> cuatroAtaquesAListDeAtaques n1 n2 n3 n4

cuatroAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> Rational
   -> [Rational]
cuatroAtaquesAListDeAtaques n1 n2 n3 n4 = [n1, n2, n3, n4]

-- ("a" t "t" "a")
parseTresAtaquesAListDeAtaques :: H [Rational]
parseTresAtaquesAListDeAtaques = parseTresAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseTresAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseTresAtaquesAListDeAtaques' = parseTresAtaquesAListDeAtaques'' <*>
   rationalOrInteger
```

```
parseTresAtaquesAListDeAtaques'' :: H (Rational -> Rational -> [Rational])
parseTresAtaquesAListDeAtaques'' = do
  n1 <- rationalOrInteger
  return $ \n2 n3 -> tresAtaquesAListDeAtaques n1 n2 n3

tresAtaquesAListDeAtaques :: Rational -> Rational -> Rational -> [Rational]
tresAtaquesAListDeAtaques n1 n2 n3 = [n1, n2, n3]

-- ("a", t "a")
parseDosAtaquesAListDeAtaques :: H [Rational]
parseDosAtaquesAListDeAtaques = parseDosAtaquesAListDeAtaques' <*>
   rationalOrInteger

parseDosAtaquesAListDeAtaques' :: H (Rational -> [Rational])
parseDosAtaquesAListDeAtaques' = do
  n1 <- rationalOrInteger
  return $ \n2 -> dosAtaquesAListDeAtaques n1 n2

dosAtaquesAListDeAtaques :: Rational -> Rational -> [Rational]
dosAtaquesAListDeAtaques n1 n2 = [n1, n2]

-- (1)
parseUnAtaqueAListDeAtaques :: H [Rational]
parseUnAtaqueAListDeAtaques = do
  n <- rationalOrInteger
  return $ unAtaqueAListDeAtaques n

unAtaqueAListDeAtaques :: Rational -> [Rational]
unAtaqueAListDeAtaques n = [n]

-- the renderer
-- getEvents :: GlobalMaterial -> Style -> Tempo -> BeginWindowTime ->
   EndWindowTime -> State InstrumentState [Event]
-- render :: (GlobalMaterial,Style,Instrument) -> Tempo -> UTCTime ->
   UTCTime -> [(UTCTime,Map Text Datum)]
-- runState :: State s a -> s -> (a, s) -- as soon as the state is
   meaningful I should stop discarding it.
--check Tidal.params for looking at the available params for webdirt
render :: ([Layer], GlobalMaterial) -> Tempo -> UTCTime -> UTCTime -> [Event
   ]
render (ls, gm) tempo iw ew = Prelude.concat $ fmap (\l -> render' (l, gm)
   tempo iw ew) ls
```

```
render' :: (Layer, GlobalMaterial) -> Tempo -> UTCTime -> UTCTime -> [Event]
render' (layer, gm) tempo iw ew = do
   fst $ runState x emptyLayerState --this should be another argument to my
       render function
    where
      x = getEvents layer gm (style layer) tempo iw ew



renderForStandalone :: ([Layer], GlobalMaterial)-> UTCTime -> UTCTime -> ([
   Event], Tempo)
renderForStandalone (ls, gm) iw ew = ((Prelude.concat $ fmap (\l ->
   renderForStandalone' (l, gm) iw ew) ls), (tempoForStandalone gm))


renderForStandalone' :: (Layer, GlobalMaterial) -> UTCTime -> UTCTime -> [
   Event]
renderForStandalone' (layer, gm) iw ew = do
   fst $ runState x emptyLayerState--this should be another argument to my
      render function
    where
      x = getEvents layer gm (style layer) (tempoForStandalone gm) iw ew
```

A.4.4 The Layer module

The Layer module describes the virtual instrument processes.

```
{-# LANGUAGE OverloadedStrings, FlexibleInstances #-}

module Sound.Seis8s.Layer where

import Data.Time
import Data.Fixed
import Data.Tempo
import Sound.OSC as H
import qualified Data.Map as M
import Data.Tuple.Select
import Control.Monad.State
-- import Control.Monad
import qualified Data.Text as T
import qualified Data.List as List

import Sound.Seis8s.Style
```

```
import Sound.Seis8s.Harmony
import Sound.Seis8s.Generic
import Sound.Seis8s.LayerState
import Sound.Seis8s.GlobalMaterial

--Layer type has to look like Instrument
-- an instrument as a layer. layer should have been the fundamental type
   all along.


-- what can be transformed from the instrument based on the Global and
   Instrument material
data Layer = Layer {
  getEvents :: GlobalMaterial -> Style -> Tempo -> BeginWindowTime ->
     EndWindowTime -> State LayerState [Event],
  style :: Style,
  tempo' :: Tempo
}

instance Show Layer where
  show (Layer getEvents futureStyle t) = show futureStyle

--I can make function s from layer to layer that tranform the output (i.e.
    like fmapping over the list of events)
-- or also override the informayioni of the style. e.g every event louder
--the style field is being prepared to later be an argument of getEvents.

mapStyle :: (Style -> Style) -> Layer -> Layer
mapStyle f x = x { style = f (style x)}

mapEvents :: (Event -> Event) -> Layer -> Layer
mapEvents f x = x {getEvents = g}
  where
    --a function calling the old function
    g gm s t iw ew = fmap (fmap f) (getEvents x gm s t iw ew) -- State
       LayerState [Event]
    --list of events as an empty list

-- an event to event function could be one where makes all the events
   louder
-- db :: Rational -> Layer -> Layer
-- db gain = mapEvents (dbEvents gain)
--
-- forwardTime :: NominalDiffTime -> Layer -> Layer
-- forwardTime n = mapEvents (efunc n)
```

```
-- let l = alternar 2 seleccionarEstilo bajo
  -- (runState (getEvents l testgmm cumbia mytempo (mytime 0) (mytime 1))
     emptyLayerStat)
-- convert windows to metric position, and do calcs in metre posiion a
   convert to actual times.
     -- if my condition is true i want to call the getEvents function of x
         and if its false call the getEvents
     --of (f x)
     -- the events could come from f0 or f1
-- from where do i take the tempo?
-- myEvent :: [Event]
-- myEvent = [((mytime 3), M.fromList [("s", string "test")])]
-- dbEvents :: Rational -> Event -> Event
-- dbEvents gain x = --uses dbamp (copy from musicW). call dbam on the
   gain to get a raw amplitude and
-- multiply that by whatever ampltude is on the event to return a new
   event.
  --
-- where :: [Event] -> [Event]
-- State LayerState [Event]-- is monad, when f map over it , you change
   the value of the last type like m a
-- the style doesn't need to be provided in layer, bcs it's implicit in
   the layer
-- now
-- parse style as layer

defTempo = mytempo



type BeginWindowTime = UTCTime
type EndWindowTime = UTCTime
type Event = (UTCTime, M.Map T.Text Datum)

-- ev :: [Event']
-- ev = [Event' (mytime 0, M.fromList [("n",Int32 {d_int32 = 1}),("note",
   Double {d_double = 0.0}),("s",ASCII_String {d_ascii_string = "teclado
   "})])]

emptyLayer :: Layer
emptyLayer = Layer {getEvents = emptyEvents, style = defaultStyle}

emptyEvents _ _ _ _ _ = do
```

```
  return $ []

-- function that generates an instrument
acordeon :: Layer
acordeon = Layer {getEvents = acordeonEvents, style = defaultStyle}

zampoa :: Layer
zampoa = Layer {getEvents = guiraEvents, style = defaultStyle}

cuerda :: Layer
cuerda = Layer {getEvents = cuerdaEvents, style = defaultStyle}

teclado :: Layer
teclado = Layer { getEvents = tecladoEvents, style = defaultStyle}

bajo :: Layer
bajo = Layer {getEvents = bajoEvents, style = defaultStyle, tempo' =
   defTempo}
--
guira :: Layer
guira = Layer {getEvents = guiraEvents, style = defaultStyle}

contras :: Layer
contras = Layer {getEvents = contrasEvents, style = defaultStyle}

tarola :: Layer
tarola = Layer {getEvents = tarolaEvents, style = defaultStyle}

efecto :: Layer
efecto = Layer {getEvents = efectoEvents, style = defaultStyle}

altavoz :: Layer
altavoz = Layer {getEvents = altavozEvents, style = defaultStyle}

congas :: Layer
congas = Layer {getEvents = congasEvents, style = defaultStyle}

clave :: Layer
clave = Layer {getEvents = claveEvents, style = defaultStyle}

jamblock :: Layer
jamblock = Layer {getEvents = jamblockEvents, style = defaultStyle}

extras :: Layer
extras = Layer {getEvents = extrasEvents, style = defaultStyle}
```

```
zampoaEvents gmm style tempo iw ew = do
  let paneo = zampoaPanPattern0 style
  let gain = zampoaGainPattern0 style
  let cutGroup = zampoaCutGroupPattern0 style
  let pitchType = (fst $ zampoaPitchPattern0 style)
  let equateLists' = equateLists (zampoaRhythmPattern0 style) (
     zampoaSampleNPattern0 style) (snd $ zampoaPitchPattern0 style)
  let zampoaRhythmPattern = sel1 equateLists' -- [(Rational, Rational)]
  let zampoaRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) zampoaRhythmPattern

  let zampoaSampleNPattern = sel2 equateLists'
  let zampoaPitchPattern = sel3 equateLists'
  let nPat = List.zip zampoaRhythmPattern' zampoaSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip zampoaRhythmPattern' zampoaPitchPattern -- [(
     RhythmicPosition, (String, Double))]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, (String, Int,
     Double))]
  let zampoaline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) zampoaline -- [UTCTime]
  let instCmap = cmap'' "zampoa" samplePat zampoaline paneo gain cutGroup --
     [Map Text Datum]
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

acordeonEvents gmm style tempo iw ew = do
  let paneo = acordeonPanPattern0 style
  let gain = acordeonGainPattern0 style
  let cutGroup = acordeonCutGroupPattern0 style
  let pitchType = (fst $ acordeonPitchPattern0 style)
  let equateLists' = equateLists (acordeonRhythmPattern0 style) (
     acordeonSampleNPattern0 style) (snd $ acordeonPitchPattern0 style)
  let acordeonRhythmPattern = sel1 equateLists' -- [(Rational, Rational)]
  let acordeonRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational
     (compas gmm), attack * toRational (compas gmm))) acordeonRhythmPattern

  let acordeonSampleNPattern = sel2 equateLists'
  let acordeonPitchPattern = sel3 equateLists'
```

```
  let nPat = List.zip acordeonRhythmPattern' acordeonSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip acordeonRhythmPattern' acordeonPitchPattern -- [(
     RhythmicPosition, (String, Double))]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, (String, Int,
     Double))]
  let acordeonline = if pitchType == "intervalo" then (generateLine pitchPat
      (harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) acordeonline -- [UTCTime
     ]
  let instCmap = cmap'' "acordeon" samplePat acordeonline paneo gain
     cutGroup --[Map Text Datum]
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

cuerdaEvents gmm style tempo iw ew = do
  let paneo = cuerdaPanPattern0 style
  let gain = cuerdaGainPattern0 style
  let cutGroup = cuerdaCutGroupPattern0 style
  let pitchType = (fst $ cuerdaPitchPattern0 style)
  let equateLists' = equateLists (cuerdaRhythmPattern0 style) (
     cuerdaSampleNPattern0 style) (snd $ cuerdaPitchPattern0 style)
  let cuerdaRhythmPattern = sel1 equateLists' -- [(Rational, Rational)]
  let cuerdaRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) cuerdaRhythmPattern

  let cuerdaSampleNPattern = sel2 equateLists'
  let cuerdaPitchPattern = sel3 equateLists'
  let nPat = List.zip cuerdaRhythmPattern' cuerdaSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip cuerdaRhythmPattern' cuerdaPitchPattern -- [(
     RhythmicPosition, (String, Double))]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, (String, Int,
     Double))]
  let cuerdaline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) cuerdaline -- [UTCTime]
  let instCmap = cmap'' "cuerdas" samplePat cuerdaline paneo gain cutGroup
     --Map Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]
```

```
  return events

tecladoEvents gmm style tempo iw ew = do
  let paneo = tecladoPanPattern0 style
  let gain = tecladoGainPattern0 style
  let cutGroupForChords = tecladoCutGroupPattern1 style
  let cutGroupForIntervaloAndMidinote = tecladoCutGroupPattern0 style


  let pitchType = fst $ tecladoPitchPattern0 style
  let equateLists' = equateLists (tecladoRhythmPattern0 style) (
     tecladoSampleNPattern0 style) (snd $ tecladoPitchPattern0 style)
  let tecladoRhythmPattern = sel1 equateLists'
  let tecladoRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) tecladoRhythmPattern

  let tecladoRhythmPattern'' = fmap (\(metre,attack) -> (metre * toRational
     (compas gmm), attack * toRational (compas gmm))) (tecladoRhythmPattern0
      style)

  let tecladoSampleNPattern = sel2 equateLists'
  let tecladoPitchPattern = sel3 equateLists'

  -- let nPat = List.zip tecladoRhythmPattern' tecladoSampleNPattern --[(
     RhythmicPattern, Int)]
  -- let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)] --
     maybe not needed of any of the instruments?

  let pat = List.zip tecladoRhythmPattern' tecladoPitchPattern -- [(
     RhythmicPosition, (String, Double))]
  let pitchPat = pitchPattern pat tempo iw ew
  let tecladoline | pitchType == "intervalo" = (generateLine pitchPat (
     harmony gmm)) -- [(Rational, Pitch )]
                  | pitchType == "midinote" = (generateLineFromMidi pitchPat)
                     -- [(Rational, Pitch)]
                 -- | pitchType == "acorde" = concatChords $ pickChords (
                    rhythmicPattern tecladoRhythmPattern' tempo iw ew) (
                    harmony gmm) --[(Rational, Pitch )]
                  | pitchType == "acorde" = concatChords $ pickChords' (
                     rhythmicPattern tecladoRhythmPattern'' tempo iw ew) (
                     harmony gmm) (snd $ tecladoPitchPattern0 style) --
                     tecladoPitchPattern p
  let time = fmap (\c -> countToTime tempo (fst c)) tecladoline -- [UTCTime]
```

```
  -- let instCmap = cmap'' "teclado" samplePat tecladoline paneo gain--Map
     Text Datum

  -- let nPat' = concat $ replicate (length (rhythmicPattern
     tecladoRhythmPattern'' tempo iw ew)) (tecladoSampleNPattern0 style)
     --[(Rational, Int)]
  -- let samplePat' = zip (fmap (\t -> timeToCount tempo t) time) (concat $
      replicate (length time) (tecladoSampleNPattern0 style))
  let samplePat' = samplePatternRat time (tecladoSampleNPattern0 style)
     tempo
  let tecladoCutGroup | (pitchType == "intervalo") || (pitchType == "
     midinote") = [cutGroupForIntervaloAndMidinote .. (
     cutGroupForIntervaloAndMidinote + (length samplePat'))]
                      | pitchType == "acorde" = [cutGroupForChords .. (
                         cutGroupForChords + (length samplePat'))]

  let instCmap = cmap'''' "teclado" samplePat' tecladoline paneo gain
     tecladoCutGroup --Map Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]
  return events


nTest = [0,0,0,0,0,0]
nTest2 = [0,0,0]

rTest :: [(Rational, Rational)]
rTest = [(0.5,0.125), (0.5,0.125), (0.5,0.125), (0.5, 0.375), (0.5, 0.375),
    (0.5, 0.375)]

rTest2 :: [(Rational, Rational)]
rTest2 = [(0.5,0.125), (0.5, 0.375)]

rTest3 :: [(Rational, Rational)]
rTest3 = [(0.5,0.125), (0.5, 0.375), (0.5,0.125), (0.5, 0.375)]

-- p
-- tecladoSampleNPattern1 = take 6 $ cycle [0],
-- tecladoRhythmPattern1 = [(1,0.25), (1,0.25), (1,0.25), (1, 0.75), (1,
    0.75), (1, 0.75)], -- ie. [   ],
-- tecladoPitchPattern1 = ("intervalo", [intervalo "unisono" 0, intervalo
    "3a" 0, intervalo "5a" 0, intervalo "unisono" 0, intervalo "3a" 0,
    intervalo "5a" 0]), -- not used yet

noteTest = [("unisono",0.0,0.0),("tercera",0.0,0.0),("quinta",0.0,0.0),("
    unisono",0.0,0.0),("tercera",0.0,0.0),("quinta",0.0,0.0)]
```

```
noteTest2 = [("unisono",0.0,0.0),("tercera",0.0,0.0),("quinta",0.0,0.0)]

noteTest3 = [("unisono",0.0,0.0),("tercera",0.0,0.0),("quinta",0.0,0.0)]

bajoEvents gmm style tempo iw ew = do
  let paneo = bassPanPattern0 style
  let gain = bassGainPattern0 style
  let cutGroup = bassCutGroupPattern0 style

  let pitchType = (fst $ bassPitchPattern0 style)
  let equateLists' = equateLists (bassRhythmPattern0 style) (
     bassSampleNPattern0 style) (snd $ bassPitchPattern0 style)
  let bassRhythmPattern = sel1 equateLists'
  let bassRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) bassRhythmPattern

  let bassSampleNPattern = sel2 equateLists'
  let bassPitchPattern = sel3 equateLists'
  let nPat = List.zip bassRhythmPattern' bassSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip bassRhythmPattern' bassPitchPattern --[(RhythmicPattern
     , Double)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let bassline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) bassline -- [UTCTime]
  let instCmap = cmap'' "bajo" samplePat bassline paneo gain cutGroup --Map
     Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

guiraEvents gmm style tempo iw ew = do
  let paneo = guiraPanPattern0 style
  let gain = guiraGainPattern0 style
  let pitchType = fst $ guiraPitchPattern0 style
  let equateLists' = equateLists (guiraRhythmPattern0 style) (
     guiraSampleNPattern0 style) (snd $ guiraPitchPattern0 style)
  let guiraRhythmPattern = sel1 equateLists'
  let guiraRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) guiraRhythmPattern
```

```
  let guiraSampleNPattern = sel2 equateLists'
  let guiraPitchPattern = sel3 equateLists'
  let nPat = List.zip guiraRhythmPattern' guiraSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip guiraRhythmPattern' guiraPitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let guiraline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) --
  let time = fmap (\c -> countToTime tempo (fst c)) guiraline -- [UTCTime]
  let instCmap = cmap'' "guira" samplePat guiraline paneo gain 0 --Map Text
     Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

contrasEvents gmm style tempo iw ew = do
  let paneo = contrasPanPattern0 style
  let gain = contrasGainPattern0 style
  let pitchType = fst $ contrasPitchPattern0 style
  let equateLists' = equateLists (contrasRhythmPattern0 style) (
     contrasSampleNPattern0 style) (snd $ contrasPitchPattern0 style)
  let contrasRhythmPattern = sel1 equateLists'
  let contrasRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) contrasRhythmPattern

  let contrasSampleNPattern = sel2 equateLists'
  let contrasPitchPattern = sel3 equateLists'
  let nPat = List.zip contrasRhythmPattern' contrasSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip contrasRhythmPattern' contrasPitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let contrasline = if pitchType == "intervalo" then (generateLine pitchPat
     (harmony gmm)) else (generateLineFromMidi pitchPat) --
  let time = fmap (\c -> countToTime tempo (fst c)) contrasline -- [UTCTime]
  let instCmap = cmap'' "contratiempos" samplePat contrasline paneo gain 0
     --Map Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

tarolaEvents gmm style tempo iw ew = do
```

```
  let paneo = tarolaPanPattern0 style
  let gain = tarolaGainPattern0 style
  let pitchType = fst $ tarolaPitchPattern0 style
  let equateLists' = equateLists (tarolaRhythmPattern0 style) (
     tarolaSampleNPattern0 style) (snd $ tarolaPitchPattern0 style)
  let tarolaRhythmPattern = sel1 equateLists'
  let tarolaRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) tarolaRhythmPattern

  let tarolaSampleNPattern = sel2 equateLists'
  let tarolaPitchPattern = sel3 equateLists'
  let nPat = List.zip tarolaRhythmPattern' tarolaSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip tarolaRhythmPattern' tarolaPitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let tarolaline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) --
  let time = fmap (\c -> countToTime tempo (fst c)) tarolaline -- [UTCTime]
  let instCmap = cmap'' "tarola" samplePat tarolaline paneo gain 0 --Map
     Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

efectoEvents gmm style tempo iw ew = do
  let paneo = efectoPanPattern0 style
  let gain = efectoGainPattern0 style
  let pitchType = fst $ efectoPitchPattern0 style
  let equateLists' = equateLists (efectoRhythmPattern0 style) (
     efectoSampleNPattern0 style) (snd $ efectoPitchPattern0 style)
  let efectoRhythmPattern = sel1 equateLists'
  let efectoRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) efectoRhythmPattern
  let efectoSampleNPattern = sel2 equateLists'
  let efectoPitchPattern = sel3 equateLists'
  let nPat = List.zip efectoRhythmPattern' efectoSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip efectoRhythmPattern' efectoPitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let efectoline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
```

```
      Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) efectoline -- [UTCTime]
  let instCmap = cmap'' "efecto" samplePat efectoline paneo gain 0 --Map
      Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

altavozEvents gmm style tempo iw ew = do
  let paneo = altavozPanPattern0 style
  let gain = altavozGainPattern0 style
  let pitchType = fst $ altavozPitchPattern0 style
  let equateLists' = equateLists (altavozRhythmPattern0 style) (
      altavozSampleNPattern0 style) (snd $ altavozPitchPattern0 style)
  let altavozRhythmPattern = sel1 equateLists'
  let altavozRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
      compas gmm), attack * toRational (compas gmm))) altavozRhythmPattern

  let altavozSampleNPattern = sel2 equateLists'
  let altavozPitchPattern = sel3 equateLists'
  let nPat = List.zip altavozRhythmPattern' altavozSampleNPattern --[(
      RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip altavozRhythmPattern' altavozPitchPattern --[(
      RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let altavozline = if pitchType == "intervalo" then (generateLine pitchPat
      (harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
      Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) altavozline -- [UTCTime]
  let instCmap = cmap'' "altavoz" samplePat altavozline paneo gain 0 --Map
      Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

extrasEvents gmm style tempo iw ew = do
  let paneo = extrasPanPattern0 style
  let gain = extrasGainPattern0 style
  let pitchType = fst $ extrasPitchPattern0 style
  let equateLists' = equateLists (extrasRhythmPattern0 style) (
      extrasSampleNPattern0 style) (snd $ extrasPitchPattern0 style)
  let extrasRhythmPattern = sel1 equateLists'
  let extrasRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
      compas gmm), attack * toRational (compas gmm))) extrasRhythmPattern
```

```
  let extrasSampleNPattern = sel2 equateLists'
  let extrasPitchPattern = sel3 equateLists'
  let nPat = List.zip extrasRhythmPattern' extrasSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip extrasRhythmPattern' extrasPitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let extrasline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) extrasline -- [UTCTime]
  let instCmap = cmap'' "extras" samplePat extrasline paneo gain 0 --Map
     Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

congasEvents gmm style tempo iw ew = do
  let paneo = congasPanPattern0 style
  let gain = congasGainPattern0 style
  let pitchType = fst $ congasPitchPattern0 style
  let equateLists' = equateLists2 (congasRhythmPattern0 style) (
     congasSampleNPattern0 style) (snd $ congasPitchPattern0 style)
  let congasRhythmPattern = sel1 equateLists'
  let congasRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) congasRhythmPattern
  let congasSampleNPattern = sel2 equateLists'
  let congasPitchPattern = sel3 equateLists'
  let nPat = List.zip congasRhythmPattern' congasSampleNPattern --[(
     RhythmicPattern, (String, Int)]
  -- let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)] --
     debe ser (Rational, (String, Int))
  let samplePat = samplePattern2 nPat tempo iw ew --[(Rational, Int)] --
     debe ser (Rational, (String, Int))
  let pat = List.zip congasRhythmPattern' congasPitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let congasline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) congasline -- [UTCTime]
  let instCmap = cmap''' samplePat congasline paneo gain 0 --Map Text Datum
  -- let instCmap = cmap''' samplePat congasline paneo gain--Map Text Datum
```

```
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

claveEvents gmm style tempo iw ew = do
  let paneo = clavePanPattern0 style
  let gain = claveGainPattern0 style
  let pitchType = fst $ clavePitchPattern0 style
  let equateLists' = equateLists (claveRhythmPattern0 style) (
     claveSampleNPattern0 style) (snd $ clavePitchPattern0 style)
  let claveRhythmPattern = sel1 equateLists'
  let claveRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational (
     compas gmm), attack * toRational (compas gmm))) claveRhythmPattern

  let claveSampleNPattern = sel2 equateLists'
  let clavePitchPattern = sel3 equateLists'
  let nPat = List.zip claveRhythmPattern' claveSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip claveRhythmPattern' clavePitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let claveline = if pitchType == "intervalo" then (generateLine pitchPat (
     harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) claveline -- [UTCTime]
  let instCmap = cmap'' "tarola" samplePat claveline paneo gain 0 --Map Text
      Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events


jamblockEvents gmm style tempo iw ew = do
  let paneo = jamblockPanPattern0 style
  let gain = jamblockGainPattern0 style
  let pitchType = fst $ jamblockPitchPattern0 style
  let equateLists' = equateLists (jamblockRhythmPattern0 style) (
     jamblockSampleNPattern0 style) (snd $ jamblockPitchPattern0 style)
  let jamblockRhythmPattern = sel1 equateLists'
  let jamblockRhythmPattern' = fmap (\(metre,attack) -> (metre * toRational
     (compas gmm), attack * toRational (compas gmm))) jamblockRhythmPattern

  let jamblockSampleNPattern = sel2 equateLists'
  let jamblockPitchPattern = sel3 equateLists'
```

```
  let nPat = List.zip jamblockRhythmPattern' jamblockSampleNPattern --[(
     RhythmicPattern, Int)]
  let samplePat = samplePattern nPat tempo iw ew --[(Rational, Int)]
  let pat = List.zip jamblockRhythmPattern' jamblockPitchPattern --[(
     RhythmicPattern, Int)]
  let pitchPat = pitchPattern pat tempo iw ew --[(Rational, Int)]
  let jamblockline = if pitchType == "intervalo" then (generateLine pitchPat
      (harmony gmm)) else (generateLineFromMidi pitchPat) -- [(Rational, [
     Pitch])]
  let time = fmap (\c -> countToTime tempo (fst c)) jamblockline -- [UTCTime
     ]
  let instCmap = cmap'' "jamblock" samplePat jamblockline paneo gain 0 --Map
      Text Datum
  let events = List.zip time instCmap -- [(UTCTime, Map Text Datum)]

  return events

-- [("quinto", 0), ("quinto", 1) ...]
cmap'''' :: String -> [(Rational, Int)] -> [(Rational, Pitch)] -> Double ->
   Double -> [Int] -> [M.Map T.Text Datum]
cmap'''' sampleName is ps pan gain cutGroup = do
  let is' = fmap (\i -> snd i) is
  let ps' = fmap (\p -> snd p) ps
  fmap (\(i, p, cg) -> cmap' sampleName i p pan gain cg) $ zip3 is' ps'
     cutGroup

cmap''' :: [(Rational, (String, Int))] -> [(Rational, Pitch)] -> Double ->
   Double -> Int -> [M.Map T.Text Datum]
cmap''' ns ps pan gain cutGroup = do
  let fs' = fmap (\f -> fst $ snd f) ns -- string - folder name
  let ns' = fmap (\n -> snd $ snd n) ns -- int
  let ps' = fmap (\p -> snd p) ps
  fmap (\(f, n, p) -> cmap' f n p pan gain cutGroup) $ zip3 fs' ns' ps'

cmap'' :: String -> [(Rational, Int)] -> [(Rational, Pitch)] -> Double ->
   Double -> Int -> [M.Map T.Text Datum]
cmap'' sampleName is ps pan gain cutGroup = do
  let is' = fmap (\i -> snd i) is
  let ps' = fmap (\p -> snd p) ps
  fmap (\(i, p) -> cmap' sampleName i p pan gain cutGroup) $ zip is' ps'

cmap' :: String -> Int -> Pitch -> Double -> Double -> Int -> M.Map T.Text
   Datum
cmap' sampleName sampleIndex pitch pan gain cutGroup = M.fromList [("s",
   string sampleName), ("n", int32 sampleIndex), ("note", double
```

```
   pitchAdjustedOctave), ("pan", double pan), ("gain", double gain), ("cut",
    int32 cutGroup)]
  where pitchAdjustedOctave = pitch - 60

cmap :: String -> Pitch -> M.Map T.Text Datum
cmap sampleName pitch = M.fromList [("s", string sampleName), ("note",
   double pitchAdjustedOctave)]
  where pitchAdjustedOctave = pitch - 60

-- returns the the rhythm, pitch and n lists with equal number of indices
equateLists :: [(Rational, Rational)] -> NPattern -> [(String, Double,
   Double)] -> ([(Rational, Rational)], [Int], [(String, Double, Double)])
equateLists attacks (NPattern1 ns) chi
  | (length attacks == length ns) && (length ns == length chi) = (attacks,
     ns, chi)
  | (length attacks > length ns) && (length attacks == length chi) = (
     attacks, take (length attacks) $ cycle' ns, chi)
  | (length attacks > length ns) && (length attacks > length chi) = (attacks
     , take (length attacks) $ cycle' ns, take (length attacks) $ cycle' chi
     )
  | (length attacks > length ns) && (length attacks > length chi) = (attacks
     , take (length attacks) $ cycle' ns, chi)
  | (length attacks > length ns) && (length attacks < length chi) = (attacks
     , take (length attacks) $ cycle' ns, chi)
  | (length attacks == length ns) && (length ns > length chi) = (attacks, ns
     , take (length ns) $ cycle' chi)
  | (length attacks < length ns) && (length ns == length chi) = (take (
     length ns) $ cycle' attacks, ns, chi)
  -- | (length attacks < length ns) && (length ns > length chi) = (take (
     length ns) $ cycle' attacks, ns, take --??
  | (length attacks < length ns) && (length ns > length chi) = (attacks, ns,
      chi)
  | (length attacks == length ns) && (length ns < length chi) = (take (
     length chi) $ cycle' attacks, take (length chi) $ cycle' ns, chi) --
     genera acordes
  | otherwise = error "case not expected"

equateLists2 :: [(Rational, Rational)] -> NPattern -> [(String, Double,
   Double)] -> ([(Rational, Rational)], [(String, Int)], [(String, Double,
   Double)])
equateLists2 attacks (NPattern2 ns) chi
  | (length attacks == length ns) && (length ns == length chi) = (attacks,
     ns, chi)
  | (length attacks > length ns) && (length attacks == length chi) = (
     attacks, take (length attacks) $ cycle' ns, chi)
```

```
  | (length attacks > length ns) && (length attacks > length chi) = (attacks
     , take (length attacks) $ cycle' ns, take (length attacks) $ cycle' chi
     )
  | (length attacks > length ns) && (length attacks > length chi) = (attacks
     , take (length attacks) $ cycle' ns, chi)
  | (length attacks > length ns) && (length attacks < length chi) = (attacks
     , take (length attacks) $ cycle' ns, chi)
  | (length attacks == length ns) && (length ns > length chi) = (attacks, ns
     , take (length ns) $ cycle' chi)
  | (length attacks < length ns) && (length ns == length chi) = (take (
     length ns) $ cycle' attacks, ns, chi)
  | (length attacks < length ns) && (length ns > length chi) = (attacks,
     take (length attacks) ns, chi)
  | (length attacks < length ns) && (length ns > length chi) = (attacks, ns,
      chi)
  | (length attacks == length ns) && (length ns < length chi) = (take (
     length chi) $ cycle' attacks, take (length chi) $ cycle' ns, chi) --
     genera acordes
  | otherwise = error "case not expected"
-- [0.5, 0.25] [1, 1] [0,1]
-- [0.5, 0.25] [1] [0,1]
-- [0.5, 0.25] [1] [0]
-- [0.25, 0.25] [1,1] [0]
-- [0.5] [1, 1] [0, 1]
-- [0.5] [1, 1] [0]
-- [0.5] [1] [0,1] -- genera acordes
-- [0, 0.125] [1] [0,1, 2]
-- bassSampleNPattern0 = [1]
-- bassRhythmPattern0 = [(1/1, 0/1), (1/1, 1/8)]
--bassPitchPattern0 = [0, 1, 2]
```

A.4.5 The Global Material module

The GlobalMaterial module contains data types and functions related to global modifications of the music in Seis8s like music harmony, tempo, and time signature.

```
module Sound.Seis8s.GlobalMaterial where

import Sound.Seis8s.Harmony
import Sound.Seis8s.Generic

import Data.Tempo
import Data.Time
```

```
--GlobalMaterial -- i.e current events in the global material
-- it doesnt change by itself, only by the user
data GlobalMaterial = GlobalMaterial {
  -- tempo :: Tempo, --look at punctual, timenot, -- e.g. tempo 120 meaning
      120/60 -- Cps, anchorTime(anyTime at the cycle grid)
     numberOfTheCycleAtThatTime
  harmony :: Progression, -- combination of a pitch and a chord type, e.g
     60 major
  compas :: Double,
  tempoForStandalone :: Tempo
} deriving (Show)

defaultGlobalMaterial = GlobalMaterial {harmony = myharmony', compas =
    establecerCompas "partido", tempoForStandalone = mytempo}
-- {harmony = Progression 1 [Chord 60 major (0, 1)], compas =
    establecerCompas "partido"}

myharmony = Progression 0.5 [Chord 60 major (0, 0.5)]
myharmony' = Progression 1 [Chord 62 minor (0, 0.5), Chord 60 major (0.5, 1)
    ]
testgmm = GlobalMaterial {harmony = myharmony', compas = establecerCompas "
    partido" }
--
-- defTempo :: Tempo
-- defTempo = do
-- tNow <- getCurrentTime
-- Tempo { freq = 0.5, time=tNow, Data.Tempo.count=0}

establecerCompas :: String -> Double
establecerCompas "4/4" = 1
establecerCompas "partido" = 0.5
establecerCompas "" = 0.5
establecerCompas _ = 1
-- myTempo = Tempo {freq = 1, time = myTime 0, count = 0}
-- tempo is provided by Estuary in this case.
-- choice 1 is tempo being separate from the language, taking it out from
    the
-- choice 2 is having it there and over
```

A.4.6 The Style module

The Style module contains hard-coded patterns of music styles like cumbia and salsa.

```
module Sound.Seis8s.Style where

import Sound.Seis8s.Generic
import Sound.Seis8s.Harmony
import Sound.Seis8s.Rhythm

import Data.Maybe
import Data.Tempo
import Data.Time
import qualified Data.List

-- style should be a collection of individual records that might be
   accessed or not by the context.
-- The style could have many different fields to access at a time.
   tecladoRhythmPattern4 -- they are defined but no universal. The style
   is the rhythm (and other stuff) of all the instruments
-- 1) A style is a library of musical information, a style has a
   tecladoRhyhtmPattern
type N = (String, Int)
data NPattern = NPattern1 [Int] | NPattern2 [N] -- (FolderDelSample, Int)
type CutGroup = Int

instance Eq NPattern where
  (==) (NPattern1 xs) (NPattern1 ys) = xs == ys
  (==) (NPattern2 xs) (NPattern2 ys) = (fmap snd xs) == (fmap snd ys)

instance Show NPattern where
  show (NPattern1 x) = show x
  show (NPattern2 y) = show y

data Style = Style {
 altavozRhythmPattern0 :: RhythmicPattern,
 altavozSampleNPattern0 :: NPattern,
 altavozPitchPattern0 :: PitchPattern,
 altavozPanPattern0 :: Double,
 altavozGainPattern0 :: Double,

 tecladoSampleNPattern0 :: NPattern,
 tecladoSampleNPattern1 :: NPattern,
 tecladoSampleNPattern2 :: NPattern,
 tecladoSampleNPattern3 :: NPattern,
 tecladoPitchPattern0 :: PitchPattern,
 tecladoPitchPattern1 :: PitchPattern,
 tecladoPitchPattern2 :: PitchPattern,
 tecladoPitchPattern3 :: PitchPattern,
```

```
tecladoRhythmPattern0 :: RhythmicPattern, -- or could this be
   accompaniment (used for various instruments)?-- [(Rational, Rational)]
    -- not universal value, but semiuniversal values.
tecladoRhythmPattern1 :: RhythmicPattern,
tecladoRhythmPattern2 :: RhythmicPattern,
tecladoRhythmPattern3 :: RhythmicPattern,
tecladoPanPattern0 :: Double,
tecladoGainPattern0 :: Double,
tecladoCutGroupPattern0 :: CutGroup,
tecladoCutGroupPattern1 :: CutGroup,



acordeonRhythmPattern0 :: RhythmicPattern,
acordeonSampleNPattern0 :: NPattern,
acordeonPitchPattern0 :: PitchPattern,
acordeonPanPattern0 :: Double,
acordeonGainPattern0 :: Double,
acordeonCutGroupPattern0 :: CutGroup,


zampoaRhythmPattern0 :: RhythmicPattern,
zampoaSampleNPattern0 :: NPattern,
zampoaPitchPattern0 :: PitchPattern,
zampoaPanPattern0 :: Double,
zampoaGainPattern0 :: Double,
zampoaCutGroupPattern0 :: CutGroup,


cuerdaRhythmPattern0 :: RhythmicPattern,
cuerdaSampleNPattern0 :: NPattern,
cuerdaPitchPattern0 :: PitchPattern,
cuerdaPanPattern0 :: Double,
cuerdaGainPattern0 :: Double,
cuerdaCutGroupPattern0 :: CutGroup,


efectoRhythmPattern0 :: RhythmicPattern,
efectoSampleNPattern0 :: NPattern,
efectoPitchPattern0 :: PitchPattern,
efectoPanPattern0 :: Double,
efectoGainPattern0 :: Double,

bassSampleNPattern0 :: NPattern,
```

```
bassSampleNPattern1 :: NPattern,
bassSampleNPattern2 :: NPattern,
bassRhythmPattern0 :: RhythmicPattern,
bassRhythmPattern1 :: RhythmicPattern,
bassRhythmPattern2 :: RhythmicPattern,
bassPitchPattern0 :: PitchPattern, --index
bassPitchPattern1 :: PitchPattern, --index
bassPitchPattern2 :: PitchPattern, --index
bassPanPattern0 :: Double,
bassGainPattern0 :: Double,
bassCutGroupPattern0 :: CutGroup,


guiraRhythmPattern0 :: RhythmicPattern,
guiraSampleNPattern0 ::NPattern,
guiraPitchPattern0 :: PitchPattern,
guiraPanPattern0 :: Double,
guiraGainPattern0 :: Double,

guiraRhythmPattern1 :: RhythmicPattern,
guiraSampleNPattern1 ::NPattern,
guiraPitchPattern1 :: PitchPattern,
guiraPanPattern1 :: Double,
guiraGainPattern1 :: Double,

contrasRhythmPattern0 :: RhythmicPattern,
contrasSampleNPattern0 :: NPattern,
contrasPitchPattern0 :: PitchPattern,
contrasPanPattern0 :: Double,
contrasGainPattern0 :: Double,

tarolaRhythmPattern0 :: RhythmicPattern,
tarolaSampleNPattern0 :: NPattern,
tarolaPitchPattern0 :: PitchPattern,
tarolaPanPattern0 :: Double,
tarolaGainPattern0 :: Double,

claveRhythmPattern0 :: RhythmicPattern,
claveSampleNPattern0 :: NPattern,
clavePitchPattern0 :: PitchPattern,
claveRhythmPattern1 :: RhythmicPattern,
claveSampleNPattern1 :: NPattern,
clavePitchPattern1 :: PitchPattern,
claveRhythmPattern2 :: RhythmicPattern,
claveSampleNPattern2 :: NPattern,
```

```
 clavePitchPattern2 :: PitchPattern,
 clavePanPattern0 :: Double,
 claveGainPattern0 :: Double,

 jamblockRhythmPattern0 :: RhythmicPattern,
 jamblockSampleNPattern0 :: NPattern,
 jamblockPitchPattern0 :: PitchPattern,
 jamblockPanPattern0 :: Double,
 jamblockGainPattern0 :: Double,

 congasRhythmPattern0 :: RhythmicPattern,
 congasSampleNPattern0 :: NPattern,
 congasPitchPattern0 :: PitchPattern,
 congasPanPattern0 :: Double,
 congasGainPattern0 :: Double,

 extrasRhythmPattern0 :: RhythmicPattern,
 extrasSampleNPattern0 :: NPattern,
 extrasPitchPattern0 :: PitchPattern,
 extrasPanPattern0 :: Double,
 extrasGainPattern0 :: Double

} deriving (Eq, Show)


 -- type PitchPostion = (Rational, Pitch)
 -- type PitchPattern = [PitchPostion]
 -- type RhythmicPosition = (Rational,Rational)
 -- type RhythmicPattern = [RhythmicPosition]

defaultStyle :: Style
defaultStyle = Style {

   altavozRhythmPattern0 = [(1, 0)],
   altavozSampleNPattern0 = NPattern1 [0],
   altavozPitchPattern0 = ("midinote", [("mn", 60, 0)]),
   altavozPanPattern0 = 0.5,
   altavozGainPattern0 = 1,


   tecladoSampleNPattern1 = NPattern1 [0],
   tecladoSampleNPattern0 = NPattern1 [0],
   tecladoRhythmPattern0 = [(1, 0)], -- ie. [   ],
   tecladoRhythmPattern1 = [(1, 0)], -- ie. [   ],
   tecladoPitchPattern0 = ("intervalo", [("unisono" , 0, 0)]),
```

```
tecladoPitchPattern1 = ("intervalo", [("unisono" , 0, 0)]), -- not used
   yet

tecladoSampleNPattern2 = NPattern1 $ take 10 $ cycle [0],
tecladoRhythmPattern2 = [(2, 0), (2, 0.25), (2, 0.25), (2, 0.5), (2,
   0.875), (2, 1.25), (2, 1.25), (2, 1.375), (2, 1.625), (2, 1.625)],
tecladoPitchPattern2 = ("intervalo", [intervalo "unisono" 0, intervalo "3
   a" 0, intervalo "5a" 0, intervalo "unisono" 0, intervalo "unisono" 0,
   intervalo "3a" 0, intervalo "5a" 0, intervalo "unisono" 0, intervalo "
   3a" 0, intervalo "5a" 0]),

tecladoSampleNPattern3 = NPattern1 $ take 14 $ cycle [0],
tecladoRhythmPattern3 = [(2, 0), (2, 0), (2, 0.25), (2, 0.25), (2, 0.5),
   (2, 0.5), (2, 0.875), (2, 0.875), (2, 1.25), (2, 1.25), (2, 1.375),
   (2, 1.375), (2, 1.625), (2, 1.625)],
tecladoPitchPattern3 = ("intervalo", [intervalo "unisono" 0, intervalo "
   unisono" 1, intervalo "3a" 0, intervalo "5a" 0, intervalo "unisono" 0,
    intervalo "unisono" 1, intervalo "unisono" 0, intervalo "unisono" 1,
   intervalo "3a" 0, intervalo "5a" 0, intervalo "unisono" 0, intervalo "
   unisono" 1, intervalo "3a" 0, intervalo "5a" 0]),

tecladoPanPattern0 = 0.5,
tecladoGainPattern0 = 0.75,

tecladoCutGroupPattern0 = 400,
tecladoCutGroupPattern1 = 500,

acordeonRhythmPattern0 = [(1,0)],
acordeonSampleNPattern0 = NPattern1 [0],
acordeonPitchPattern0 = ("intervalo", [("unisono", 0, 0)]),
acordeonPanPattern0 = 0.5,
acordeonGainPattern0 = 1,

acordeonCutGroupPattern0 = 300,

zampoaRhythmPattern0 = [(1,0)],
zampoaSampleNPattern0 = NPattern1 [0],
zampoaPitchPattern0 = ("intervalo", [("unisono", 0, 0)]),
zampoaPanPattern0 = 0.5,
zampoaGainPattern0 = 1,

zampoaCutGroupPattern0 = 200,

cuerdaRhythmPattern0 = [(1,0)],
cuerdaSampleNPattern0 = NPattern1 [0],
```

```
cuerdaPitchPattern0 = ("intervalo", [("unisono", 0, 0)]),
cuerdaPanPattern0 = 0.5,
cuerdaGainPattern0 = 0.75,

cuerdaCutGroupPattern0 = 100,


bassSampleNPattern0 = NPattern1 [0],
bassSampleNPattern1 = NPattern1 [0],
bassSampleNPattern2 = NPattern1 [0], --index
bassRhythmPattern0 = [(1, 0)], --i.e. [    ],
bassRhythmPattern1 = [], --i.e. [   ],
bassRhythmPattern2 = [],
bassPitchPattern0 = ("intervalo", [("unisono", 0, 0)]), -- int
bassPitchPattern1 = ("intervalo", [("unisono", 0, 0)]), --interval
bassPitchPattern2 = ("intervalo", [("unisono", 0, 0)]), --index
bassPanPattern0 = 0.5,
bassGainPattern0 = 1,

bassCutGroupPattern0 = 2,


guiraRhythmPattern0 = [(1,0)],
guiraSampleNPattern0 = NPattern1 [0],
guiraPitchPattern1 = ("midinote", [("mn", 60, 0)]),
guiraPanPattern1 = 0.5,
guiraGainPattern1 = 1,

guiraRhythmPattern1 = [(1,0)],
guiraSampleNPattern1 = NPattern1 [0],
guiraPitchPattern0 = ("midinote", [("mn", 60, 0)]),
guiraPanPattern0 = 0.5,
guiraGainPattern0 = 1,

contrasRhythmPattern0 = [(1, 0)],
contrasSampleNPattern0 = NPattern1 [0],
contrasPitchPattern0 =("midinote", [("mn", 60, 0)]),
contrasPanPattern0 = 0.5,
contrasGainPattern0 = 1,

tarolaRhythmPattern0 = [(1, 0)],
tarolaSampleNPattern0 = NPattern1 [0],
tarolaPitchPattern0 = ("midinote", [("mn", 60, 0)]),
tarolaPanPattern0 = 0.5,
tarolaGainPattern0 = 1,
```

```
   efectoRhythmPattern0 = [(1, 0)],
   efectoSampleNPattern0 = NPattern1 [0],
   efectoPitchPattern0 = ("intervalo", [("unisono", 0, 0)]),
   efectoPanPattern0 = 0.5,
   efectoGainPattern0 = 1,

   extrasRhythmPattern0 = [(1, 0)],
   extrasSampleNPattern0 = NPattern1 [0],
   extrasPitchPattern0 = ("intervalo", [intervalo "unisono" 0]),
   extrasPanPattern0 = 0.5,
   extrasGainPattern0 = 1,

  congasRhythmPattern0 = [(1, 0)],
  congasSampleNPattern0 = NPattern2 [("quinto", 0)],
  congasPitchPattern0 = ("intervalo", [intervalo "unisono" 0]),
  congasPanPattern0 = 0.5,
  congasGainPattern0 = 1,



  jamblockRhythmPattern0 = [(1, 0)],
  jamblockSampleNPattern0 = NPattern1 [0],
  jamblockPitchPattern0 = ("midinote", [("mn", 60, 0)]),
  jamblockPanPattern0 = 0.5,
  jamblockGainPattern0 = 1,

  claveRhythmPattern0 = [(1, 0)],
  claveSampleNPattern0 = NPattern1 [0],
  clavePitchPattern0 = ("midinote", [("mn", 84, 0)]),
  claveRhythmPattern1 = [(2, 0.25), (2, 0.5), (2, 1), (2, 1.375), (2, 1.75)
     ], -- clave 2 3
  claveSampleNPattern1 = NPattern1 $ take 5$ cycle [0],
  clavePitchPattern1 = ("midinote", take 5 $ cycle [("mn", 84, 0)]),
  claveRhythmPattern2 = [(2, 0), (2, 0.375), (2, 0.75), (2, 1.25), (2, 1.5)
     ], -- clave 3 2
  claveSampleNPattern2 = NPattern1 $ take 5$ cycle [0],
  clavePitchPattern2 = ("midinote", take 5 $ cycle [("mn", 84, 0)]),
  clavePanPattern0 = 0.5,
  claveGainPattern0 = 1
}

--hh
--metal
```

```
-- where cumbia is a collection of all , but no information from the gmm
   harmonies or pitches. Might be information on how to pick harmonies.
   resist the temptation to customize, if I need new information I should
   make a new field to it.
-- cumbia is a collection of knowledges (this would change). Try not to be
    too specific or universal

-- user to access the presets or only have one preset that can be modified
    with functions
-- maybe have two, functions that transform them
-- options: override information, field tranformation, tranform all the
   rhythm at the same time?,
-- eg. remove any attacks but the fundamental, where multiple fields get
   transformed,
-- but still be particular to the style, so it will still be specific

-- supercollider is modular that expects information from the user.

cumbia :: Style
cumbia = Style {
    altavozRhythmPattern0 = [(1, 0)],
    altavozSampleNPattern0 = NPattern1 [0],
    altavozPitchPattern0 = ("midinote", [("mn", 60, 0)]),
    altavozPanPattern0 = 0.5,
    altavozGainPattern0 = 1,

    tecladoRhythmPattern0 = [(1, 0.75)], -- ie. [   ],
    tecladoSampleNPattern0 = NPattern1 [0],
    tecladoPitchPattern0 = ("acorde", [intervalo "unisono" 0, intervalo "3a"
        0, intervalo "5a" 0]), -- not used yet

    tecladoSampleNPattern1 = NPattern1 [0, 0, 0],-- NPattern1 [0, 0, 0, 1,
       1, 1]
    tecladoRhythmPattern1 = [(1,0.25), (1, 0.75)], -- ie. [   ],
    tecladoPitchPattern1 = ("acorde", [intervalo "unisono" 0, intervalo "3a"
        0, intervalo "5a" 0]),

    tecladoSampleNPattern2 = NPattern1 $ take 10 $ cycle [0],
    tecladoRhythmPattern2 = [(2, 0), (2, 0.25), (2, 0.25), (2, 0.5), (2,
       0.875), (2, 1.25), (2, 1.25), (2, 1.375), (2, 1.625), (2, 1.625)],
    tecladoPitchPattern2 = ("intervalo", [intervalo "unisono" 0, intervalo "
       3a" 0, intervalo "5a" 0, intervalo "unisono" 0, intervalo "unisono"
       0, intervalo "3a" 0, intervalo "5a" 0, intervalo "unisono" 0,
       intervalo "3a" 0, intervalo "5a" 0]),
```

```
tecladoSampleNPattern3 = NPattern1 $ take 14 $ cycle [0],
tecladoRhythmPattern3 = [(2, 0), (2, 0), (2, 0.25), (2, 0.25), (2, 0.5),
    (2, 0.5), (2, 0.875), (2, 0.875), (2, 1.25), (2, 1.25), (2, 1.375),
   (2, 1.375), (2, 1.625), (2, 1.625)],
tecladoPitchPattern3 = ("intervalo", [intervalo "unisono" 0, intervalo "
   unisono" 1, intervalo "3a" 0, intervalo "5a" 0, intervalo "unisono"
   0, intervalo "unisono" 1, intervalo "unisono" 0, intervalo "unisono"
   1, intervalo "3a" 0, intervalo "5a" 0, intervalo "unisono" 0,
   intervalo "unisono" 1, intervalo "3a" 0, intervalo "5a" 0]),

-- tecladoPitchPattern0 = ("voicing", [0, 0, 0]), -- not used yet
tecladoPanPattern0 = 0.5,
tecladoGainPattern0 = 0.75,

tecladoCutGroupPattern0 = 400,

tecladoCutGroupPattern1 = 500,


acordeonRhythmPattern0 = [(1,0)],
acordeonSampleNPattern0 = NPattern1 [0],
acordeonPitchPattern0 = ("intervalo", [intervalo "unisono" 0]), -- or
   double? (nota [0, 2, 3] cumbia) cuerda
acordeonPanPattern0 = 0.5,
acordeonGainPattern0 = 1,

acordeonCutGroupPattern0 = 300,

zampoaRhythmPattern0 = [(1,0)],
zampoaSampleNPattern0 = NPattern1 [0],
zampoaPitchPattern0 = ("intervalo", [intervalo "unisono" 0]), -- or
   double? (nota [0, 2, 3] cumbia) cuerda
zampoaPanPattern0 = 0.5,
zampoaGainPattern0 = 1,

zampoaCutGroupPattern0 = 200,


cuerdaRhythmPattern0 = [(1,0)],
cuerdaSampleNPattern0 = NPattern1 [0],
cuerdaPitchPattern0 = ("intervalo", [intervalo "unisono" 0]), -- or
   double? (nota [0, 2, 3] cumbia) cuerda
cuerdaPanPattern0 = 0.5,
cuerdaGainPattern0 = 1,
```

```
cuerdaCutGroupPattern0 = 100,


bassRhythmPattern0 = [(1, 0), (1, 0.5), (1, 0.75)], --i.e. [   ],
bassSampleNPattern0 = NPattern1 [0, 0, 0],
bassPitchPattern0 = ("intervalo", [intervalo "unisono" 0, intervalo "3a"
    0, intervalo "5a" 0]), -- index from list of pitches i.e. [60, 64,
   67]
bassRhythmPattern1 = [(1, 0), (1, 0.5)], --i.e. [    ],
bassSampleNPattern1 = NPattern1 [0, 0],
bassPitchPattern1 = ("intervalo", [intervalo "unisono" 0, intervalo "5a"
    0]),
bassPanPattern0 = 0.5,
bassGainPattern0 = 1,

bassCutGroupPattern0 = 2,


-- bassRhythmPattern2 = [(8, 0), (8, 0.5), (8, 0.75), (8, 1), (8, 1.5),
    (8, 1.75), (8, 2), (1, 2.5), (8, 2.75), (8, 3), (8, 3.5), (8, 3.75)
   , (8, 4), (8, 4.5), (8, 4.75), (8, 5), (8, 5.5), (8, 5.75), (8, 6),
   (8, 6.5), (8, 6.75), (8, 7), (8, 7.25), (8, 7.5), (8, 7.75)], --i.e.
    [
-- bassSampleNPattern2 = take 25 $ cycle [0],
-- bassPitchPattern2 = [0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0,
    1, 2, 0, 1, 2, 0, 2, 0, 2],
-- tonicayquinta2 $ cumbia bajo
-- Aqui, escuchars tres notas distinas, la tnica, la quinta y la quinta
    una octava abajo (i.e. ms grave).
bassRhythmPattern2 = [(1, 0), (1, 0.5), (1, 0.75)],
bassSampleNPattern2 = NPattern1 [0, 0, 0],
bassPitchPattern2 = ("intervalo", [intervalo "unisono" 0, intervalo "5a"
    0, intervalo "5a" (-1)]), -- index from list of pitches i.e. [60,
   64, 67]

-- bassRhythmPattern2 = [(4, 0), (4, 0.5), (4, 0.75), (4, 1), (4, 1.5),
    (4, 1.75), (4, 2), (4, 2.5), (4, 2.75), (4, 3), (4, 3.25), (4, 3.5)
   , (4, 3.75)],
-- bassSampleNPattern2 = take 13 $ cycle [0],
-- bassPitchPattern2 = ("intervalo", [intervalo "unisono", intervalo "3
   a", intervalo "5a", intervalo "unisono", intervalo "3a", intervalo
   "5a", intervalo "unisono", intervalo "3a", intervalo "5a", intervalo
    "unisono", intervalo "5a", intervalo "unisono", intervalo "5a"]),
```

```
guiraRhythmPattern0 = [(1, 0), (1, 0.25), (1, 0.375), (1, 0.5), (1,
   0.75), (1, 0.875)], --i.e. [   ]
guiraSampleNPattern0 = NPattern1 [0, 1, 2, 0, 1, 2],
guiraPitchPattern0 = ("midinote", [("mn", 60, 0), ("mn", 60, 0), ("mn",
   60, 0), ("mn", 60, 0), ("mn", 60, 0), ("mn", 60, 0)]),

guiraRhythmPattern1 = [(1, 0), (1, 0.5)], --i.e. [   ]
guiraSampleNPattern1 = NPattern1 $ take 2 $ cycle [0],
guiraPitchPattern1 = ("midinote", take 2 $ cycle [("mn", 60, 0)]),

guiraPanPattern0 = 0.5,
guiraGainPattern0 = 0.8,


guiraPanPattern1 = 0.5,
guiraGainPattern1 = 0.5,

contrasRhythmPattern0 = [(1, 0), (1, 0.25), (1, 0.5), (1, 0.75)],
contrasSampleNPattern0 = NPattern1 [0, 1, 0, 1, 0, 1, 0, 1],
contrasPitchPattern0 = ("midinote", [("mn", 60, 0), ("mn", 60, 0), ("mn"
   , 60, 0), ("mn", 60, 0), ("mn", 60, 0), ("mn", 60, 0)]),
contrasPanPattern0 = 0.5,
contrasGainPattern0 = 1,

tarolaRhythmPattern0 = [(1, 0.375), (1, 0.75)],
tarolaSampleNPattern0 = NPattern1 [0, 0],
tarolaPitchPattern0 = ("midinote", [("mn", 60, 0), ("mn", 60, 0)]),
tarolaPanPattern0 = 0.5,
tarolaGainPattern0 = 1,

efectoRhythmPattern0 = [(1, 0)],
efectoSampleNPattern0 = NPattern1 [0],
efectoPitchPattern0 = ("intervalo", [intervalo "unisono" 0]),
efectoPanPattern0 = 0.5,
efectoGainPattern0 = 1,

congasRhythmPattern0 = [(1, 0), (1, 0.25), (1, 0.5), (1, 0.75)],
-- congasSampleNPattern0 = [1, 2, 1, 2],
congasSampleNPattern0 = NPattern2 [("quinto", 0), ("quinto", 1), ("
   quinto", 0), ("quinto", 1)],
congasPitchPattern0 = ("midinote", take 4 $ cycle [("mn", 60, 0)]),
congasPanPattern0 = 0.5,
congasGainPattern0 = 1,

claveRhythmPattern0 = [(1, 0), (1, 0.5)],
```

```
    claveSampleNPattern0 = NPattern1 $ take 2 $ cycle [0],
    clavePitchPattern0 = ("midinote", take 2 $ cycle [("mn", 84, 0)]),
    claveRhythmPattern1 = [(2, 0.25), (2, 0.5), (2, 1), (2, 1.375), (2,
       1.75)], -- clave 2 3
    claveSampleNPattern1 = NPattern1 $ take 5$ cycle [0],
    clavePitchPattern1 = ("midinote", take 5 $ cycle [("mn", 84, 0)]),
    claveRhythmPattern2 = [(2, 0), (2, 0.375), (2, 0.75), (2, 1.25), (2,
       1.5)], -- clave 3 2
    claveSampleNPattern2 = NPattern1 $ take 5$ cycle [0],
    clavePitchPattern2 = ("midinote", take 5 $ cycle [("mn", 84, 0)]),
    clavePanPattern0 = 0.5,
    claveGainPattern0 = 1,

    jamblockRhythmPattern0 = [(1, 0), (1, 0.5)],
    jamblockSampleNPattern0 = NPattern1 $ take 2 $ cycle [0],
    jamblockPitchPattern0 = ("midinote", take 2 $ cycle [("mn", 60, 0)]),
    jamblockPanPattern0 = 0.5,
    jamblockGainPattern0 = 1,

    extrasRhythmPattern0 = [(1, 0)],
    extrasSampleNPattern0 = NPattern1 [0],
    extrasPitchPattern0 = ("intervalo", [intervalo "unisono" 0]),
    extrasPanPattern0 = 0.5,
    extrasGainPattern0 = 1
  }


-- 1. When to play

rhythmicPattern :: RhythmicPattern -> (Tempo -> UTCTime -> UTCTime -> [
    Rational])
rhythmicPattern xs t iw ew = Data.List.sort $ concat $ fmap (\(x, y) ->
    findBeats t iw ew x y) xs

samplePatternRat :: [UTCTime] -> NPattern -> Tempo -> [(Rational, Int)]
samplePatternRat times (NPattern1 xs)tempo = do
  let times' = (fmap (\t -> timeToCount tempo t) times)
  let nPat = (concat $ replicate (length times) xs)
  zip times' nPat

samplePattern2 :: [(RhythmicPosition, (String, Int))] -> Tempo -> UTCTime ->
     UTCTime -> [(Rational, (String, Int))]
samplePattern2 xs t iw ew = Data.List.sort $ concat $ fmap (\x ->
    samplePattern2' x t iw ew) xs
```

```
samplePattern2' :: (RhythmicPosition, (String, Int)) -> Tempo -> UTCTime ->
   UTCTime -> [(Rational, (String, Int))]
samplePattern2' (xs, (s, i)) t iw ew = fmap (\attack -> (attack, (s, i)))
   attacks
  where
    attacks = findBeats t iw ew (fst xs) (snd xs)

samplePattern :: [(RhythmicPosition, Int)] -> Tempo -> UTCTime -> UTCTime ->
    [(Rational, Int)]
samplePattern xs t iw ew = Data.List.sort $ concat $ fmap (\x ->
   samplePattern' x t iw ew) xs

samplePattern' :: (RhythmicPosition, Int) -> Tempo -> UTCTime -> UTCTime ->
   [(Rational, Int)]
samplePattern' (xs, sampleNumber) t iw ew = fmap (\attack -> (attack,
   sampleNumber)) attacks
  where
    attacks = findBeats t iw ew (fst xs) (snd xs)

pitchPattern :: [(RhythmicPosition, Note)] -> Tempo -> UTCTime -> UTCTime ->
    [(Rational, Note)]
pitchPattern xs t iw ew = Data.List.sort $ concat $ fmap (\x -> pitchPattern
   ' x t iw ew) xs

pitchPattern' :: (RhythmicPosition, Note) -> Tempo -> UTCTime -> UTCTime ->
   [(Rational, Note)]
pitchPattern' (xs, (relacion, midiOintervalo, octava)) t iw ew = fmap (\
   attack -> (attack, (relacion, midiOintervalo, octava))) attacks
  where
    attacks = findBeats t iw ew (fst xs) (snd xs)



--webdirt uses end uses the % dependant on the length of the sample, cut
   is used for drums (might be useful)
--we might need to add the duration parameters
```

A.4.7 The Rhythm module

The Rhythm module contains data types relative to rhythmic patterns.

```
module Sound.Seis8s.Rhythm where
```

```
type Metre = Rational
type Attack = Rational
type Start = Rational
type End = Rational
type RhythmicPosition = (Metre, Attack)
type ChordPosition = (Start, End)
type RhythmicPattern = [RhythmicPosition]
```

A.4.8 The Harmony module

The Harmony module contains data types and functions related to music harmony.

```
module Sound.Seis8s.Harmony where

import Sound.Seis8s.Generic
import Sound.Seis8s.Rhythm


import Data.Time
import Data.Tempo
import Data.Maybe
import Data.Function
import Data.List as List
import Data.Tuple.Select

type Relacion = String -- "segunda" "absoluto"
type Octava = Double
type Note = (Relacion, Double, Octava)
type PitchType = String -- intervalo o midinote
type PitchPattern = (PitchType, [Note])

data Chord = Chord Pitch ChordType ChordPosition deriving (Show)
type Pitch = Double
type ChordType = [Double]
-- data Harmony = Harmony Chord ChordPosition ChordPosition deriving (Show
   )-- :: (Rational,Rational)
data Progression = Progression Metre [Chord] deriving (Show)

generateLineFromMidi :: [(Rational, (String, Double, Double))] -> [(Rational
   , Pitch)]
generateLineFromMidi xs = fmap generateLineFromMidi' xs
```

```
generateLineFromMidi' :: (Rational, (String, Double, Double)) -> (Rational,
   Pitch)
generateLineFromMidi' (attack, (midiIdentifier, midinote, octava)) = (attack
   , midinote)

generateLine :: [(Rational, (String, Double, Octava))] -> Progression -> [(
   Rational, Pitch)]
generateLine attacksAndIntervals (Progression metre chords) = sort $ concat
   $ fmap (generateNotesFromChord attacksAndIntervals metre) chords

generateNotesFromChord :: [(Rational, (String, Double, Octava))] -> Metre ->
    Chord -> [(Rational, Pitch)]
generateNotesFromChord attacksAndIntervals metre chord = concat $ fmap (\x
   -> generateSingleNoteFromChord x metre chord) attacksAndIntervals

generateSingleNoteFromChord :: (Rational, (String, Double, Octava)) -> Metre
    -> Chord -> [(Rational, Pitch)]
generateSingleNoteFromChord (attack, (tipo, interval, octava)) metre (Chord
   p t (start, end))
  |compareRationalWChordRange attack metre (start, end) = attackAndNote
     attack note
  | otherwise = []
  where
    note = getNoteInChord (Chord p t (start, end)) (tipo, interval, octava)
       -- Maybe Pitch

attackAndNote :: Rational -> Maybe Pitch -> [(Rational, Pitch)]
attackAndNote attack (Just note) = [(attack, note)]
attackAndNote _ Nothing = []

-- functions to pick a chord that has a voicing

pickChords' :: [Rational] -> Progression -> [(String, Double, Octava)] -> [(
   Rational, [Pitch])]
pickChords' attacks (Progression metre chords) pitchPattern = concat $ fmap
   (\a -> pickChordsWithVoicingFromRational' (Progression metre chords) a
   pitchPattern) attacks

pickChordsWithVoicingFromRational' :: Progression -> Rational -> [(String,
   Double, Octava)] -> [(Rational, [Pitch])]
pickChordsWithVoicingFromRational' (Progression metre chords) attack
   pitchPattern = do
   let acordes = acordesConVoicing $ getNotesInChords (Progression metre
      chords) pitchPattern--[([Pitch], (Rational, Rational))]
   concat $ fmap (pickSingleChordFromRational' attack metre) acordes
```

```
pickSingleChordFromRational' :: Rational -> Metre -> ([Pitch], (Rational,
   Rational)) -> [(Rational, [Pitch])]
pickSingleChordFromRational' attack metre (acorde, (start, end))
  | compareRationalWChordRange attack metre (start, end) = [(attack, acorde)
     ]
  | otherwise = []

getNotesInChords :: Progression -> [(String, Double, Octava)] -> [([Pitch],
   (Start, End))]
getNotesInChords (Progression metre chords) intervalos = fmap (\c ->
   getNotesInChord c intervalos) chords

getNotesInChord :: Chord -> [(String, Double, Octava)] -> ([Pitch], (Start,
   End))
getNotesInChord (Chord root chordType (start, end)) intervalos = do
  let acorde = catMaybes $ fmap (\i -> getNoteInChord (Chord root chordType
     (start, end)) i) intervalos
  (acorde, (start, end))

getNoteInChord :: Chord -> (String, Double, Octava) -> Maybe Pitch
getNoteInChord (Chord root chordType (start, end)) (tipo, interval, octava)
      | tipo == "segunda" = (+) <$> Just root <*> ((+) <$> Just (12 * octava
         ) <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "tercera" = (+) <$> Just root <*> ((+) <$> Just (12 * octava
         ) <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "cuarta" = (+) <$> Just root <*> ((+) <$> Just (12 * octava)
          <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "quinta" = (+) <$> Just root <*> ((+) <$> Just (12 * octava)
          <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "sexta" = (+) <$> Just root <*> ((+) <$> Just (12 * octava)
         <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "septima" = (+) <$> Just root <*> ((+) <$> Just (12 * octava
         ) <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "novena" = (+) <$> Just root <*> ((+) <$> Just (12 * octava)
          <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "oncena" = (+) <$> Just root <*> ((+) <$> Just (12 * octava)
          <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | tipo == "trecena" = (+) <$> Just root <*> ((+) <$> Just (12 * octava
         ) <*> (intervaloDisponible chordType (tipo, interval, octava)))
      | otherwise = (+) <$> Just root <*> ((+) <$> Just (12 * octava) <*>
         Just interval)

 -- [Double]
intervaloDisponible :: ChordType -> (String, Double, Octava) -> Maybe Double
```

```
intervaloDisponible cht (tipo, _, _)
      |cht == major && tipo == "segunda" = Just $ sel2 (intervalo "2maj" 0)
      |cht == major7 && tipo == "segunda" = Just $ sel2 (intervalo "2maj" 0)
      |cht == minor && tipo == "segunda" = Nothing
      |cht == minor7 && tipo == "segunda" = Nothing
      |cht == dom && tipo == "segunda" = Just $sel2 (intervalo "2maj" 0)
      |cht == sus2 && tipo == "segunda" = Just $sel2 (intervalo "2maj" 0)
      |cht == sus4 && tipo == "segunda" = Nothing
      |cht == fifths && tipo == "segunda" = Just $ sel2 (intervalo "2maj" 0)
      |cht == dim && tipo == "segunda" = Nothing
      |cht == aug && tipo == "segunda" = Nothing

      |cht == major && tipo == "tercera" = Just $ sel2 (intervalo "3maj" 0)
      |cht == major7 && tipo == "tercera" = Just $ sel2 (intervalo "3maj" 0)
      |cht == minor && tipo == "tercera" = Just $ sel2 (intervalo "3m" 0)
      |cht == minor7 && tipo == "tercera" = Just $ sel2 (intervalo "3m" 0)
      |cht == dom && tipo == "tercera" = Just $ sel2 (intervalo "3maj" 0)
      |cht == sus2 && tipo == "tercera" = Nothing
      |cht == sus4 && tipo == "tercera" = Nothing
      |cht == fifths && tipo == "tercera" = Nothing
      |cht == dim && tipo == "tercera" = Just $ sel2 (intervalo "3m" 0)
      |cht == dim7 && tipo == "tercera" = Just $ sel2 (intervalo "3m" 0)
      |cht == semidim && tipo == "tercera" = Just $ sel2 (intervalo "3m" 0)
      |cht == aug && tipo == "tercera" = Just $ sel2 (intervalo "3maj" 0)

      |cht == major && tipo == "cuarta" = Just $ sel2 (intervalo "4justa" 0)
      |cht == major7 && tipo == "cuarta" = Just $ sel2 (intervalo "4justa" 0)
      |cht == minor && tipo == "cuarta" = Just $ sel2 (intervalo "4justa" 0)
      |cht == minor7 && tipo == "cuarta" = Just $ sel2 (intervalo "4justa" 0)
      |cht == dom && tipo == "cuarta" = Nothing
      |cht == sus2 && tipo == "cuarta" = Nothing -- ?
      |cht == sus4 && tipo == "cuarta" = Nothing -- ?
      |cht == fifths && tipo == "cuarta" = Nothing
      |cht == dim && tipo == "cuarta" = Just $ sel2 (intervalo "4justa" 0)
      |cht == dim7 && tipo == "cuarta" = Just $ sel2 (intervalo "4justa" 0)
      |cht == semidim && tipo == "cuarta" = Just $ sel2 (intervalo "4justa"
         0)
      |cht == aug && tipo == "cuarta" = Just $ sel2 (intervalo "4justa" 0)

      |cht == major && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
      |cht == major7 && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
      |cht == minor && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
      |cht == minor7 && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
      |cht == dom && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
      |cht == sus2 && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
```

```
|cht == sus4 && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
|cht == fifths && tipo == "quinta" = Just $ sel2 (intervalo "5justa" 0)
|cht == dim && tipo == "quinta" = Just $ sel2 (intervalo "5b" 0)
|cht == dim7 && tipo == "quinta" = Just $ sel2 (intervalo "5b" 0)
|cht == semidim && tipo == "quinta" = Just $ sel2 (intervalo "5b" 0)
|cht == aug && tipo == "quinta" = Just $ sel2 (intervalo "5aug" 0)

|cht == major && tipo == "sexta" = Just $ sel2 (intervalo "6maj" 0)
|cht == major7 && tipo == "sexta" = Just $ sel2 (intervalo "6maj" 0)
|cht == minor && tipo == "sexta" = Nothing
|cht == minor7 && tipo == "sexta" = Nothing
|cht == dom && tipo == "sexta" = Just $ sel2 (intervalo "6maj" 0)
|cht == sus2 && tipo == "sexta" = Nothing -- ?
|cht == sus4 && tipo == "sexta" = Nothing -- ?
|cht == fifths && tipo == "sexta" = Nothing
|cht == dim && tipo == "sexta" = Just $ sel2 (intervalo "6b" 0)
|cht == dim7 && tipo == "sexta" = Just $ sel2 (intervalo "6b" 0)
|cht == semidim && tipo == "sexta" = Just $ sel2 (intervalo "6b" 0)
|cht == aug && tipo == "sexta" = Nothing

|cht == major && tipo == "septima" = Just $ sel2 (intervalo "7maj" 0)
|cht == major7 && tipo == "septima" = Just $ sel2 (intervalo "7maj" 0)
|cht == minor && tipo == "septima" = Just $ sel2 (intervalo "7m" 0)
|cht == minor7 && tipo == "septima" = Just $ sel2 (intervalo "7m" 0)
|cht == dom && tipo == "septima" = Just $ sel2 (intervalo "7m" 0)
|cht == sus2 && tipo == "septima" = Nothing -- ?
|cht == sus4 && tipo == "septima" = Nothing -- ?
|cht == fifths && tipo == "septima" = Nothing
|cht == dim && tipo == "septima" = Just $ sel2 (intervalo "7bb" 0) --
   m7b5
|cht == dim7 && tipo == "septima" = Just $ sel2 (intervalo "7bb" 0)
|cht == semidim && tipo == "septima" = Just $ sel2 (intervalo "7m" 0)
|cht == aug && tipo == "septima" = Just $ sel2 (intervalo "7m" 0)

|cht == major && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
|cht == major7 && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
|cht == minor && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
|cht == minor7 && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
|cht == dom && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
|cht == sus2 && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0) --
    ?
|cht == sus4 && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0) --
    ?
|cht == fifths && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
```

```
      |cht == dim && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0) --
         m7b5
      |cht == dim7 && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
      |cht == semidim && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)
      |cht == aug && tipo == "novena" = Just $ sel2 (intervalo "9maj" 0)

      |cht == major && tipo == "oncena" = Just $ sel2 (intervalo "11aug" 0)
      |cht == major7 && tipo == "oncena" = Just $ sel2 (intervalo "11aug" 0)
      |cht == minor && tipo == "oncena" = Just $ sel2 (intervalo "11justa" 0)
      |cht == minor7 && tipo == "oncena" = Just $ sel2 (intervalo "11justa"
         0)
      |cht == dom && tipo == "oncena" = Just $ sel2 (intervalo "11aug" 0)
      |cht == sus2 && tipo == "oncena" = Just $ sel2 (intervalo "11b" 0)
      |cht == sus4 && tipo == "oncena" = Just $ sel2 (intervalo "11b" 0)
      |cht == fifths && tipo == "oncena" = Just $ sel2 (intervalo "11justa"
         0)-- ?
      |cht == dim && tipo == "oncena" = Just $ sel2 (intervalo "11justa" 0)--
          m7b5
      |cht == dim7 && tipo == "oncena" = Just $ sel2 (intervalo "11justa" 0)
      |cht == semidim && tipo == "oncena" = Just $ sel2 (intervalo "11justa"
         0)
      |cht == aug && tipo == "oncena" = Just $ sel2 (intervalo "11aug" 0)

      |cht == major && tipo == "trecena" = Just $ sel2 (intervalo "13maj" 0)
      |cht == major7 && tipo == "trecena" = Just $ sel2 (intervalo "13maj" 0)
      |cht == minor && tipo == "trecena" = Nothing
      |cht == minor7 && tipo == "trecena" = Nothing
      |cht == dom && tipo == "trecena" = Just $ sel2 (intervalo "13b" 0)
      |cht == sus2 && tipo == "trecena" = Nothing
      |cht == sus4 && tipo == "trecena" = Just $ sel2 (intervalo "13b" 0)
      |cht == fifths && tipo == "trecena" = Nothing -- ?
      |cht == dim && tipo == "trecena" = Just $ sel2 (intervalo "13b" 0)--
         m7b5
      |cht == dim7 && tipo == "trecena" = Just $ sel2 (intervalo "13b" 0)
      |cht == semidim && tipo == "trecena" = Just $ sel2 (intervalo "13b" 0)
      |cht == aug && tipo == "trecena" = Nothing

      |otherwise = Nothing

  -- (!!) (generateChord (Chord root chordType)) (round interval)

-- generates a list of chords according to a given harmony and a given
   rhythm pattern
-- generatechords :: [Rational] -> Progression -> [(Rational, [Pitch])]
generatechords :: [Rational] -> Progression -> [(Rational, [Pitch])]
```

```
generatechords attacks (Progression metre chords) = concat $ fmap (
   generateChordsFromRational (Progression metre chords)) attacks

generateChordsFromRational :: Progression -> Rational -> [(Rational, [Pitch
   ])]
generateChordsFromRational (Progression metre chords) attack = concat $ fmap
    (generateSingleChordFromRational attack metre) chords

generateSingleChordFromRational :: Rational -> Metre -> Chord -> [(Rational,
    [Pitch])]
generateSingleChordFromRational attack metre (Chord p chordType (start, end)
   )
  | compareRationalWChordRange attack metre (start, end) = [(attack,
     generateChord (Chord p chordType (start, end)))]
  | otherwise = []

-- return a list of pitches from one chord
generateChord :: Chord -> [Pitch]
generateChord (Chord root chordType (start, end)) = fmap ((+) root)
   chordType

pickChords :: [Rational] -> Progression -> [(Rational, [Pitch])]
pickChords attacks (Progression metre chords) = concat $ fmap (
   pickChordsWithVoicingFromRational (Progression metre chords)) attacks

pickChordsWithVoicingFromRational :: Progression -> Rational -> [(Rational,
   [Pitch])]
pickChordsWithVoicingFromRational (Progression metre chords) attack = do
   let acordes = acordesConVoicing $ generateChordsFromProg (Progression
      metre chords) --[([Pitch], (Rational, Rational))]
   concat $ fmap (pickSingleChordFromRational attack metre) acordes

pickSingleChordFromRational :: Rational -> Metre -> ([Pitch], (Rational,
   Rational)) -> [(Rational, [Pitch])]
pickSingleChordFromRational attack metre (acorde, (start, end))
  | compareRationalWChordRange attack metre (start, end) = [(attack, acorde)
     ]
  | otherwise = []

generateChordsFromProg :: Progression -> [([Pitch], (Rational, Rational))]
generateChordsFromProg (Progression metre chords) = fmap (\c ->
   generateChordFromProg c) chords

generateChordFromProg :: Chord -> ([Pitch], (Rational, Rational))
generateChordFromProg (Chord root chordType (start, end)) = do
```

```
  let acorde = fmap ((+) root) chordType
  (acorde, (start,end))

compareRationalWChordRange :: Rational -> Metre -> ChordPosition -> Bool
compareRationalWChordRange attack metre (startOffset, endOffset) = do
  let attackInMetre = attack / metre
  let attackInMetre' = fract attackInMetre
  let startInMetre = startOffset / metre
  let endInMetre = endOffset / metre
  let b | startOffset <= endOffset = (attackInMetre' >= startInMetre) && (
     attackInMetre' < endInMetre)
       | otherwise = (attackInMetre' < startInMetre) && (attackInMetre' >=
          endInMetre)
  b
-- armonia 1 [c 0 5] => Progression 0.5 [Chord 60 major (0, 0.25)]
-- attack 1 es 0.5
-- compas gmm = compas "partido" (i.e. 0.5)
-- por lo tanto compareRationalWChordRange' 0.5 0.5 0.5 (0, 0.25)
-- compareRationalWChordRange' 1 1 1 (0, 0.5)
-- not in use
compareRationalWChordRange' :: Rational -> Metre -> Double -> ChordPosition
   -> Bool
compareRationalWChordRange' attack metre compas (startOffset, endOffset) =
   do
  let attack' = attack / toRational compas
  let metre' = metre / toRational compas
  let startOffset' = startOffset / toRational compas
  let endOffset' = endOffset / toRational compas
  let attackInMetre = attack' / metre' -- 1/1 = 1
  let attackInMetre' = fract attackInMetre -- 1.0 -> 0
  let startInMetre = startOffset' / metre' -- 0/1 = 0
  let endInMetre = endOffset' / metre' -- 0.5/1 = 0.5
  let b | startOffset' <= endOffset' = (attackInMetre' >= startInMetre) && (
     attackInMetre' < endInMetre)
       | otherwise = (attackInMetre' < startInMetre) && (attackInMetre' >=
          endInMetre)
  b


concatChord :: (Rational, [Pitch]) -> [(Rational, Pitch)]
concatChord (attack, ps) = fmap (\p -> (attack, p)) (snd (attack, ps))

concatChords :: [(Rational, [Pitch])] -> [(Rational, Pitch)] -- eg. for [60,
    64, 67] three events all whith the same time all with different
   pitches
```

```
concatChords attackAndChords = concat $ fmap (\x -> concatChord x)
   attackAndChords


-- a simple voicing

c1 = Chord 60 major (0, 0.5)
c2 = Chord 64 minor (0.5, 1)

--generateLine xs 1 myharmony = [(9 % 20,60.0),(9 % 20,64.0),(9 % 20,67.0)
   ]
agruparNotas :: [(Rational, Pitch)] -> [[(Rational, Pitch)]]
agruparNotas xs = groupBy ((==) `on` fst) xs
-- 1. se recibe como [(Rational, Pitch)]
-- 2. lo transformamos a [[(Rational, Pitch)]]
-- 3. lo regresamos a [(Rational, Pitch)]
-- [(0, 59), (0, 62), (0, 67), (0.5, 60), (0.5, 64), (0.5, 67)]
-- generateChordsFromProg :: Progression -> [([Pitch], (Rational, Rational
   ))]
--
--
acordesConVoicing :: [([Pitch], (Rational, Rational))] -> [([Pitch], (
   Rational, Rational))]
acordesConVoicing prog
  | length prog == 0 = []
  | length prog == 1 = prog
  | length prog == 2 = [a, ab]
  | length prog == 3 = [a, ab, bc]
  | length prog == 4 = [a, ab, bc, cd]
  | length prog == 5 = [a, ab, bc, cd, de]
  | length prog == 6 = [a, ab, bc, cd, de, ef]
  | length prog == 7 = [a, ab, bc, cd, de, ef, fg]
  | length prog == 8 = [a, ab, bc, cd, de, ef, fg, gh]
  | length prog == 9 = [a, ab, bc, cd, de, ef, fg, gh, hi]
  | length prog == 10 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij]
  | length prog == 11 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk]
  | length prog == 12 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl]
  | length prog == 13 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm]
  | length prog == 14 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
     mn]
  | length prog == 15 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
     mn, no]
  | length prog == 16 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
     mn, no, op]
```

```
| length prog == 17 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq]
| length prog == 18 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr]
| length prog == 19 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs]
| length prog == 20 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st]
| length prog == 21 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu]
| length prog == 22 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv]
| length prog == 23 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw]
| length prog == 24 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx]
| length prog == 25 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy]
| length prog == 26 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy, yz]
| length prog == 27 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy, yz, za]
| length prog == 28 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy, yz, za, ab']
| length prog == 29 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy, yz, za, ab', bc']
| length prog == 30 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy, yz, za, ab', bc', cd']
| length prog == 31 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy, yz, za, ab', bc', cd',
   de']
| length prog == 32 = [a, ab, bc, cd, de, ef, fg, gh, hi, ij, jk, kl, lm,
   mn, no, op, pq, qr, rs, st, tu, uv, vw, wx, xy, yz, za, ab', bc', cd',
   de', ef']
  where
    a = (!!) prog 0
    ab = acordeConVoicing a ((!!) prog 1)
    bc = acordeConVoicing ab ((!!) prog 2)
    cd = acordeConVoicing bc ((!!) prog 3)
    de = acordeConVoicing cd ((!!) prog 4)
    ef = acordeConVoicing de ((!!) prog 5)
    fg = acordeConVoicing ef ((!!) prog 6)
    gh = acordeConVoicing fg ((!!) prog 7)
    hi = acordeConVoicing gh ((!!) prog 8)
    ij = acordeConVoicing hi ((!!) prog 9)
```

```
      jk = acordeConVoicing ij ((!!) prog 10)
      kl = acordeConVoicing jk ((!!) prog 11)
      lm = acordeConVoicing kl ((!!) prog 12)
      mn = acordeConVoicing lm ((!!) prog 13)
      no = acordeConVoicing mn ((!!) prog 14)
      op = acordeConVoicing no ((!!) prog 15)
      pq = acordeConVoicing op ((!!) prog 16)
      qr = acordeConVoicing pq ((!!) prog 17)
      rs = acordeConVoicing qr ((!!) prog 18)
      st = acordeConVoicing rs ((!!) prog 19)
      tu = acordeConVoicing st ((!!) prog 20)
      uv = acordeConVoicing tu ((!!) prog 21)
      vw = acordeConVoicing uv ((!!) prog 22)
      wx = acordeConVoicing vw ((!!) prog 23)
      xy = acordeConVoicing wx ((!!) prog 24)
      yz = acordeConVoicing xy ((!!) prog 25)
      za = acordeConVoicing yz ((!!) prog 26)
      ab' = acordeConVoicing za ((!!) prog 27)
      bc' = acordeConVoicing ab' ((!!) prog 28)
      cd' = acordeConVoicing bc' ((!!) prog 29)
      de' = acordeConVoicing cd' ((!!) prog 30)
      ef' = acordeConVoicing de' ((!!) prog 31)
--
-- -- [(9 % 20,60.0),(9 % 20,64.0),(9 % 20,67.0)]
-- -- [([Pitch], (Rational, Rational))] -- ([60, 64, 67], (0, 1))

acordeConVoicing :: ([Pitch], (Rational, Rational)) -> ([Pitch], (Rational,
   Rational)) -> ([Pitch], (Rational, Rational))
acordeConVoicing prev sig = do
  let notasComunesPrev = notasComunes (fst prev) (fst sig) -- [(0, 60), (0,
     64), (0, 67)] [(0.25, 64), (0.25, 67), (0.25, 71)] = [(0.25, 64),
     (0.25, 67)]
  let notasComunesSig = notasComunes (fst sig) (fst prev)
  let notasNoComunesPrev = notasNoComunes' (fst prev) notasComunesPrev --
     [(0, 60)]
  let notasNoComunesSig = notasNoComunes' (fst sig) notasComunesSig --
     [(0.5, 71)]
  let dists = distancias'' notasNoComunesPrev notasNoComunesSig -- [(Double,
      Pitch)]
  let notasCercanas' = notasCercanas (length notasNoComunesSig) dists --
     [71] -- [Pitch]
  let listaFinal = List.sort $ notasComunesPrev ++ notasCercanas'
  (listaFinal, snd sig)
```

```
-- notasNoComunes' :: [(Rational, Pitch)] -> [(Rational, Pitch)] -> [(
   Rational, Pitch)]
notasNoComunes' :: [Pitch] -> [Pitch] -> [Pitch]
notasNoComunes' xs ys = catMaybes $ fmap (\x -> notasNoComunes x ys) xs

-- [([Pitch], (Rational, Rational))]
-- notasNoComunes :: (Rational, Pitch) -> [(Rational, Pitch)] -> Maybe (
   Rational, Pitch)
notasNoComunes :: Pitch -> [Pitch] -> Maybe Pitch
notasNoComunes x xs
  | elem x xs = Nothing
  | otherwise = Just x

-- 71 [64, 67, 71] [64, 67]

-- 1. encontrar las notas en comun entre ambas listas
-- notasComunes :: [Pitch] -> [Pitch] -> [Pitch]
notasComunes :: [Pitch] -> [Pitch] -> [Pitch]
notasComunes prev sig = concat $ fmap (\p -> notaComun' p sig) prev

-- notaComun' :: (Rational, Pitch) -> [(Rational, Pitch)] -> [(Rational,
   Pitch)]
notaComun' :: Pitch -> [Pitch] -> [Pitch]
notaComun' prev sig = do
  let hacerListasDeListas' = hacerListasDeListas sig
  concat $ fmap (\s -> notaComun prev s) hacerListasDeListas'

-- notaComun :: (Rational, Pitch) -> [(Rational, Pitch)] -> [(Rational,
   Pitch)]
notaComun :: Pitch -> [Pitch] -> [Pitch]
notaComun prev sig = filter ((==) prev) sig

-- hacerListasDeListas :: [(Rational, Pitch)] -> [[(Rational, Pitch)]]
hacerListasDeListas :: [Pitch] -> [[Pitch]]
hacerListasDeListas xs = fmap (\x -> hacerListaDeListas x) xs

-- hacerListaDeListas :: (Rational, Pitch) -> [(Rational, Pitch)]
hacerListaDeListas :: Pitch -> [Pitch]
hacerListaDeListas x = do
  let xs = List.sort $ [x, (x - 12) .. 0] ++ [(x + 12), (x + 24) .. 127] --
     [Pitch]
  xs

-- (0, 60) [(0.5, 65), (0.5, 69), (0.5, 72)]
-- (0, 60) [[ ], [], []]
```

```
-- 2. Escoger todos los 3 valores con distancia ms corta de la lista.
distancias :: Pitch -> Pitch -> [(Double, Pitch)]
distancias prev sig = do
  let saltoSig = (sig - 12)
  let sig' = (sig + 12)
  let saltoSig' = (sig + 12) + 12
  let sigs = List.sort $ [sig, saltoSig .. 0] ++ [sig', saltoSig' .. 127] --
      [Pitch]
  let sigs' = catMaybes $ fmap (\s -> if (s >= (prev - 12)) && (s <= (prev +
      12)) then Just s else Nothing) sigs -- [(Rational, Pitch)]
  let sigsYdist = fmap (\d -> (abs $ prev - d, d)) sigs'
  sigsYdist

distancias' :: Pitch -> [Pitch] -> [(Double, Pitch)]
distancias' prev sig = concat $ fmap (\s -> distancias prev s) sig

distancias'' :: [Pitch] -> [Pitch] -> [(Double, Pitch)]
distancias'' prev sig = List.sort $ concat $ fmap (\p -> distancias' p sig)
   prev

-- uniq which removes duplicates from a list (found here https://
   codereview.stackexchange.com/questions/150533/filter-duplicate-elements
   -in-haskell)
uniqPitch :: Eq b => [(a,b)] -> [(a,b)]
uniqPitch [] = []
uniqPitch ((x, y):xs) = (x,y) : uniqPitch (filter ((/= y) . snd) xs)

notasCercanas :: Int -> [(Double, Pitch)] -> [Pitch]
notasCercanas tamano notas = do
  let lista1 = take tamano $ uniqPitch notas
  List.sort $ fmap snd lista1




-- pitches

c :: Double
c = 60

cs :: Double
cs = 61

d :: Double
```

```
d = 62

ds :: Double
ds = 63

e :: Double
e = 64

f :: Double
f = 65

fs :: Double
fs = 66

g :: Double
g = 67

gs :: Double
gs = 68

a :: Double
a = 69

as :: Double
as = 70

b :: Double
b = 71

-- intervalos para generar acordes

major :: ChordType
major = [0, 4, 7]

minor:: ChordType
minor = [0, 3, 7]

major7 :: ChordType
major7 = [0, 4, 7, 11]

minor7 :: ChordType
minor7 = [0,3,7,10]

dom :: ChordType
dom = [0,4,7,10]
```

```
fifths :: ChordType
fifths = [0, 7]

sus4 :: ChordType
sus4 = [0,5,7]

sus2 :: ChordType
sus2 = [0, 2, 7]

aug :: ChordType
aug = [0, 4, 8]

dim :: ChordType
dim = [0, 3, 6]

semidim :: ChordType --m7b5
semidim = [0, 3, 6, 10]

dim7 :: ChordType
dim7 = [0, 3, 6, 9]

-- opcion 1: funcion octava
-- octavar -1 (intervalo "5a") -- returns ("quinta", (-1*12) (snd
   intervalo "5a"))
-- entonces => octavar (intervalo "5a", "4a", "3a") que seria equivalente
   a intervalo "5a" + 1, etc...
-- opcion 2: incorporarlo en la funcion intervalo
--intervalo "t" -1

-- una lista de intervalos
intervaloDouble :: Double -> Double -> (String, Double, Octava)
intervaloDouble intervalo octava = ("libre", intervalo, octava)

intervalo :: String -> Double -> (String, Double, Octava)
intervalo "unisono" octava = ("unisono", 0, octava)

intervalo "f" octava = ("unisono", 0, octava)

intervalo "1a" octava = ("unisono", 0, octava)

intervalo "2a" octava = ("segunda", 0, octava)

intervalo "2m" octava = ("segundaMenor", 1, octava)
```

```
intervalo "2maj" octava = ("segundaMayor", 2, octava)

intervalo "3a" octava = ("tercera", 0, octava)

intervalo "3m" octava = ("terceraMenor", 3, octava)

intervalo "3maj" octava = ("terceraMayor", 4, octava)

intervalo "4a" octava = ("cuarta", 0, octava)

intervalo "4justa" octava = ("cuartaJusta", 5, octava)

intervalo "4aug" octava = ("cuartaAug", 6, octava)

intervalo "5a" octava = ("quinta", 0, octava)

intervalo "5b" octava = ("quintaBemol", 6, octava)

intervalo "5justa" octava = ("quintaJusta", 7, octava)

intervalo "5aug" octava = ("quintaAug", 8, octava)

intervalo "6a" octava = ("sexta", 0, octava)

intervalo "6m" octava = ("sextaMenor", 8, octava)

intervalo "6maj" octava = ("sextaMayor", 9, octava)

intervalo "7a" octava = ("septima", 0, octava)

intervalo "7bb" octava = ("septimaMenor", 8, octava)

intervalo "7m" octava = ("septimaMenor", 10, octava)

intervalo "7maj" octava = ("septimaMayor", 11, octava)

intervalo "8a" octava = ("octava", 12, octava)

intervalo "9a" octava = ("novena", 0, octava)

intervalo "9m" octava = ("novenaMenor", 13, octava)

intervalo "9maj" octava = ("novenaMayor", 14, octava)

intervalo "11a" octava = ("oncena", 0, octava)
```

```
intervalo "11b" octava = ("oncenaBemol", 16, octava)

intervalo "11justa" octava = ("oncenaJusta", 17, octava)

intervalo "11aug" octava = ("oncenaAug", 18, octava)

intervalo "13a" octava = ("trecena", 0, octava)

intervalo "13b" octava = ("trecenaBemol", 20, octava)

intervalo "13maj" octava = ("trecenaMayor", 21, octava)

intervalo _ octava = ("nada", 0, octava)
```