Modelling Concurrent Systems with Interval Processes

MODELLING CONCURRENT SYSTEMS WITH INTERVAL PROCESSES

BY

MOHAMMED A. ALQARNI, M.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2016) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE:	Modelling Concurrent Systems with Interval Processes
AUTHOR:	Mohammed A. Alqarni M.Sc. (Computational Sciences) Laurentian University, Sudbury, ON, Canada
SUPERVISOR:	Ryszard Janicki (Professor)
NUMBER OF PAGES:	xi, 110

To my wife Nada and my son Ali, I dedicate this thesis.

Abstract

Standard operational semantics of the majority of concurrency models is defined in terms of either sequences or step sequences, while standard concurrent history semantics is usually defined in terms of partial orders, stratified order structures (or structures equivalent to them as net processes).

It is commonly assumed (first argued by N. Wiener in 1914) that any system run (execution) that can be observed by a single observer must be an interval order of event occurrences.

However, generating interval orders directly is problematic for most models of concurrency, as the only feasible sequence representation of interval order is by using Fishburn Theorem (1970) and appropriate sequences of beginnings and endings of events involved. It was shown by Janicki and Koutny in 1997 that concurrent histories involving interval orders can be represented by interval order structures, but how these interval order structures could be derived for particular concurrent systems was not clear.

My original contribution to knowledge is defining an interval order semantics for Petri Nets with Inhibitor Arcs. We start with introducing operational interval order semantics, and then we generalize the concept of net process to represent the set of equivalent executions modelled by interval orders.

Next we will show that our interval processes correspond to appropriate interval order structures. Finally, we will prove that our model is equivalent to that of Janicki and Yin (2015) where novel *interval traces* are used to represent equivalent executions.

We will also demonstrate that our model covers simpler cases where sequences or step sequences were used to represent system runs.

Acknowledgements

First and foremost I would like to thank my advisor Professor Ryszard Janicki. It has been an honor to be his Ph.D. student. I appreciate all his contributions of time, and ideas to make my Ph.D. experience productive and stimulating. His expertise and keen logic have been of great value to me. His advice on both research as well as on my career have been priceless.

I would also like to thank Professor Sanzheng Qiao, and Professor Michael Soltys for serving as my advisory committee members.

Furthermore, I would like pay tribute to the late King Abdullah of Saudi Arabia for giving me and many other of my fellow countrymen and women the opportunity to seek post secondary education all over the world. If it was not for his scholarship program, I would not have gotten this far. May he rest in peace. For that, I acknowledge the financial support provided by the Ministry of Education in Saudi Arabia through The Saudi Arabian Cultural Bureau in Canada.

Last but not least, my deepest gratitude and love belong to my wife Nada, whose support and patience were with me all these years, and my little son Ali, who gave me a different perspective of life.

Contents

Abstract **iii**

Acknowledgements **v**

1 Introduction and Motivation **1**

1.1 Motivation 6

1.2 Organization of Thesis 7

2 Mathematical Foundation **9**

2.1 Partial, Total, Stratified and Interval Orders 9

3 Concurrent Histories and Order Structures **13**

3.1 Concurrent Histories 13

3.2 Stratified and Interval Order Structures 16

4 Petri Nets **21**

4.1 Elementary Nets 21

4.1.1 Formal Definition of Elementary Nets 22

4.1.2 Operational Semantics of Elementary Nets 25

4.1.3 Fundamental Situations in Dynamic Systems Modeling 27

4.1.4 Contact Freeness 29
4.2 Elementary Nets with Inhibitor Arcs 31
4.3 Activator Nets 34

5 Interval Elementary Net with Inhibitor Arcs 37

6 Process Semantics 44

6.1 Semantical Framework for Process Semantics 45
6.2 Processes of Elementary Nets 49
6.3 Processes of Elementary Nets with Inhibitor Arcs 53
6.4 Processes and Concurrent Histories 57

7 Trace Semantics 62

7.1 Mazurkiewicz Traces 62
7.2 Comtraces 67
7.3 Interval Traces 70
7.3.1 Interval Traces Construction 70
7.3.2 Interval Traces Semantics for Elementary Net with Inhibitor Arcs 73

8 Interval Processes and Interval Order Structures 75

9 Interval Processes and Interval Traces 88

10 Interval Semantic for Other Petri Nets 91

10.1 Interval-Timed Petri Nets 92
10.2 Interval Semantic for Activator Nets 97

11 Conclusion 100

Bibliography **104**

List of Figures

1.1 Inhibitor nets $\mathtt{N}$ and $\mathtt{N}_{io}$ and all their behaviours involving one occurrence of a, b and c. The net $\mathtt{N}$ generates $<_1^{\mathtt{N}}, <_2^{\mathtt{N}}, <_3^{\mathtt{N}}, <_4^{\mathtt{N}}$, and two concurrent histories, while $\mathtt{N}_{io}$ generates only an interval order $<_4^{\mathtt{N}}$. Partial orders are represented by Hasse diagrams. All orders except $<_4^{\mathtt{N}}$ are stratified. 7

2.1 Various types of partial orders (represented as Hasse diagrams), the total order defined by the sequence $aaba$, and the stratified order defined by the step sequence $x = \{a, b\}\{a, c\}\{a\}\{b, c\}$. 10

4.1 A simple example of an elementary net. 24

4.2 An illustration of local changes of states and firing rules. 25

4.3 Concurrency of transition t_1 and t_2. 28

4.4 Causality of transition t_1 and t_2. 28

4.5 Conflict between transition t_1 and t_2. 29

4.6 An example of an elementary net with inhibitor arc. 32

4.7 An example of an activator net. 35

5.1 ENI $\mathtt{N}$ from Figure 1.1 and its interval representation $\mathcal{N}$. 39

6.1 Semantical framework for process semantics from Kleijn and Koutny (2004) (right) and complete semantical framework for process semantics from Juhás et al. (2007) 47
6.2 A step by step construction of a process for EN from Figure 4.1 generated by $x = \{t_1t_4t_5t_6\}$. 52
6.3 Complement closed ENI. 54
6.4 A step by step construction of a process for ENI from Figure 6.3 generated by $y = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4, t_5\}\}$. 56
6.5 Direct acyclic graph associated with the process in Figure 6.2(e) and its transitive closure. 57
6.6 Rules of deriving a partial order from an ENI process generated by a firing sequence x. 58
6.7 Direct acyclic graph associated with the activator occurrence net in Figure 6.4(e) and its transitive closure assuming it was generated by $x = \{t_1t_2t_3t_1t_5t_4\}$. 59
6.8 Rules of deriving a stratified order from an activator occurrence net (process) generated by a step sequence y. 60
6.9 Stratified order structure associated with the activator occurrence net in Figure 6.4(e). 61
7.1 The partial order $\lessdot_{[s]}^{trace}$ generated by the trace $[s]$ where $s = abcbca$ and $ind = \{(b, c), (c, b)\}$. 66
7.2 An example of relation *sim* (simultaneity), and congruence relation *ser* (serializability). 69
8.1 Process construction for $\mathcal{N}$ (interval representation of N from Figure 5.1). 76

8.2 An example of a process $\mathcal{P}_x^{\mathsf{iseq}}$, the directed acyclic graph $\preccurlyeq_x^{init}$, the partial order $\preccurlyeq_x^{proc}$, the relations $\prec_x$, $\sqsubset_x$ and the interval order structure $S^x = (\{a^1, b^1, c^1\}, \prec_x, \sqsubset_x)$. The net here is $\mathcal{N}$ from Figure 5.1, and $x = BaBcEaEcBbEb$. 77

8.3 An example of an ENI that generates *only* interval orders. Our method results in the process $\mathcal{P}_z^{\mathsf{iseq}}$ and the interval order $\preccurlyeq_z$, which is isomorphic to $<_4^{\mathtt{N}}$ of Figure 1.1, while all techniques based on either firing sequences or firing step sequences produce *empty* set. 83

8.4 An example of an ENI, its interval representation, processes and concurrent histories they generate. The process $\mathsf{P}_x^{\mathsf{seq}}$ generates a concurrent history $\{<_2^{\mathsf{N}}, <_3^{\mathsf{N}}\}$ while the process $\mathcal{P}_y^{\mathsf{iseq}}$ generates $\{<_2^{\mathsf{N}}, <_3^{\mathsf{N}}, <_4^{\mathsf{N}}\}$. . . 87

10.1 An interval-timed Petri net (ITPN). 93

10.2 An initial part of an arbitrary process of the ITPN of Figure 10.1 . . 97

10.3 The splitting of transition a (left) into a^- and a^- (right). 98

Chapter 1

Introduction and Motivation

Concurrent systems have become the norm, due to technological development and our demand for computing power. Such systems can be characterized by having a number of different activities being executed at the same time. Moreover, there is usually a number of distributed components which need to communicate with each other in concurrent systems. Thus, they are complex in nature. Due to this complexity, the history of concurrency theory research consists of both: the construction of languages and models to make this complexity manageable, and the development of theories for describing and reasoning about interacting components. One of the most prominent development is *Petri Nets*.

Petri nets introduced by Carl Adam Petri in his dissertation "Communication with Automata", see Petri (1962, 1966). Petri nets provide both comprehensible graphical constructs to model concurrent systems and a rich set of analysis techniques to reason about those systems; therefore, it appeals to both practitioners and theorist. In its most common formulation, a Petri net consists of places, or local states, and transitions effecting the change of local states. The latter is possible if, for a given

transition, a specified set of local states is currently active, or marked in Petri net terminology. Petri nets (the basic model and other Petri net classes) have been used to model a variety of dynamic event-driven systems like computers networks (Marsan et al. (1986)), manufacturing plants (Venkatesh et al. (1994)), real-time computing systems (Tsai et al. (1995)), and workflow (Van Der Aalst and Van Hee (2004); Chuang et al. (2002)) to name just a few examples. This wide spectrum of applications is accompanied by a wide spectrum of different classes of Petri nets that has made such applications possible. Such classes were built upon the basic notion of a net defined by Petri and went to introduce concepts that were not present in the basic net; take Petri nets with inhibitor or activator arcs, Timed Petri nets, and colored Petri nets for example. The differences between those classes are generally in the relation between transitions, and how expressive the nets are. Although some classes of Petri nets will be touched upon, *Elementary Petri nets with inhibitor arcs* will be the primary model in this thesis. Petri nets with inhibitor arcs are in one hand very simple as they are just classical nets accompanied with inhibitor arcs (allowing to test for the absence of a token) and on the other hand they are far more expressive than the classical model. Requiring a transitions executability to be also dependent on some specific local states *not* being marked is perhaps the most natural extension of the basic net model. As stated in Peterson (1981), 'Petri nets with inhibitor arcs are intuitively the most direct approach to increasing the modelling power of Petri nets.' In general, Petri net with inhibitor arcs can simulate the computations of Turing machines (i.e. test for the absence of a token can be translated to 'test for zero'). Therefore, several important decision problems like reachability and liveness which are decidable for basic Petri nets are undecidable for Petri nets with inhibitor arcs Hack (1979). 'Test for zero'

means that inhibitor arcs are well suited to model situations involving testing for a specific condition, rather than producing and consuming resources. Therefore, Petri nets with inhibitor arcs have been found useful in area like communication protocols and performance analysis.

Once a system is model as a Petri net, the simplest form to describe the behaviours of that net is using operational semantics (i.e. sequential semantics) which can be viewed in two ways based on how transitions' occurrences are captured. If interleaved concurrency is the goal, then *firing sequences* are used, however, if true concurrency is of interest, then *firing step sequences* are used. As their names suggest, a firing sequence is simply a sequence of transitions that fire one after another while firing step sequences are sequences of sets (possibly singleton) of transitions that fire concurrently.

Although firing step sequences provide some insights about the relationships between transitions (which ones can occur concurrently) of the systems, it is still of a sequential flavor and lacking the ability to convey some important information (such as causality between transitions). Therefore, operational semantics, in both forms (firing sequences and step sequences), cannot provide enough information that allows for determining what systems runs/executions are equivalent, despite the importance of this equivalence in the concurrent processing.

To address the shortcomings of operational semantics, different semantics have been developed with the goal of obtaining more information about the relation between transitions of a given net. One example is *concurrent histories* (also called *non-sequential execution histories*) which are defined as abstractions of equivalent

runs. Those abstractions are modelled based on the assumptions made about system runs/executions. They could be partial orders (as in Goltz and Reisig (1983); Nielsen et al. (1990)), stratified orders structures (as in Janicki and Koutny (1991, 1995); Kleijn and Koutny (2004); Rozenberg and Engelfriet (1998)), or interval orders structures (as in Janicki and Koutny (1991); Janicki and Yin (2015)).

Although concurrent histories semantics yield better insights about systems, it is important to note that, unlike validity of operational semantics which is usually obvious, the validity of concurrent history/behaviour semantics is often not. It relies on the validity of the definition of a concurrent history/behaviour, which is often not trivial and may involve complex reasoning (cf. Baldan et al. (2004); Janicki (2008); Janicki et al. (2010); Nielsen et al. (1990)). On the other hand, the *process semantics* (in the sense of Busi and Pinna (1999); Kleijn and Koutny (2008); Nielsen et al. (1990); Rozenberg and Engelfriet (1998)), does not usually require much validation as intuitively it is just a set of system unfoldings, so it is as natural as any operational semantics (c.f. Kleijn and Koutny (2004); Nielsen et al. (1990); Vogler et al. (1998)). Hence it can be used as a benchmark for validity of other types of history/behaviour semantics, they just have to be equivalent to the process semantics (c.f. Busi and Pinna (1999); Kleijn and Koutny (2004, 2008)).

That being said, recording observations as firing sequences and step sequences (and consequentially abstractions generated by them) is not the most precise way when it comes to defining semantics based on observations. It is commonly assumed (first argued in Wiener (1914) and formally analyzed in details in Janicki and Koutny (1993)) that observations that are observed by a single observer *must* be interval orders of event occurrences. This means that the most precise observational semantics

is defined in terms of interval orders. Moreover, representing observations as interval orders allows to capture behaviours that neither of the standard semantics can really describe. However generating interval orders directly is problematic for most models of concurrency, as the only feasible *sequence representation* of interval order is by using Fishburn Theorem Fishburn (1970) and appropriate sequences of *beginnings* and *endings* of events involved. It was shown in Janicki and Koutny (1997) that concurrent histories involving interval orders can be represented by *interval order structures* (proposed in Janicki and Koutny (1991); Lamport (1986)), but how these interval order structures could be derived for particular concurrent systems was not clear.

The main contributions of our research include the following. First we define an *interval process semantics* for Petri Nets with Inhibitor Arcs. It is achieved by modifying and extending the ideas of Kleijn and Koutny (2004) so we can generate processes that are equivalent to interval orders when their sequence representation derived from Fishburn Theorem is used. While various process semantics for Petri Nets with inhibitor arcs have been proposed in the past, the most representative and prolific are probably Busi and Pinna (1999); Janicki and Koutny (1995); Juhás et al. (2007); Kleijn and Koutny (2004); Montanari and Rossi (1995); Vogler et al. (1998); Winkowski (1998), *they all assume that the operational semantics is defined in terms of sequences* Montanari and Rossi (1995); Vogler et al. (1998); Winkowski (1998) *or step sequences* Busi and Pinna (1999); Janicki and Koutny (1995); Juhás et al. (2007); Kleijn and Koutny (2004). None of these models is able to deal with observations (system runs) that are neither step sequences nor semantically equivalent to any step sequence. We then show that such processes can be interpreted as appropriate

interval order structures of Janicki and Koutny (1991). Finally, we will show that our interval processes correspond uniquely to appropriate *interval traces* of Janicki and Yin (2015).

The initial results of this thesis have been published in Alqarni and Janicki (2015), Alqarni and Janicki (2016a), and Alqarni and Janicki (2016b).

1.1 Motivation

Consider the elementary net with inhibitor arcs $\mathtt{N}$ in Figure 1.1 together with its concurrent histories $hist_1^{\mathtt{N}}$ and $hist_2^{\mathtt{N}}$ involving one occurrence of each transition a, b and c (elementary net with inhibitor arcs will be formally defined in Section 4.2 and concurrent histories is the focus of Chapter 3).

Standard operational semantics (firing sequences and step sequences) and stander concurrent semantics (partial orders and stratified order) can model $<_1^{\mathtt{N}}, <_2^{\mathtt{N}}, <_3^{\mathtt{N}}$ generated by $\mathtt{N}$. Intuitively, $<_1^{\mathtt{N}}$ is when transition a occurs, then b followed by c, $<_2^{\mathtt{N}}$ is when transition c occurs, then a followed by b, and $<_3^{\mathtt{N}}$ is when both transition a and c can occur concurrently then b occurs. However, neither can model $<_4^{\mathtt{N}}$ which can also be considered as a possible run/execution that involves occurrences of a, b and c, for both $\mathtt{N}$ and $\mathtt{N}_{\mathtt{io}}$. Intuitively, if the events a, b and c are *not instantaneous*, one can imagine a situation where b follows a but c *overlaps* with both a and b. This means that for the net $\mathtt{N}$ and $\mathtt{N}_{io}$ of Figure 1.1, one can, intuitively, 'hold on' to a token taken from s_2 until a token taken from s_1 is placed in s_5.

Moreover, we would like to point out that for the net N_{io} from Figure 1.1 the set of all firing sequences that start from the marking $\{s_1, s_2\}$ and end at the marking $\{s_4, s_5\}$ is *empty* and the set of all firing step sequences that start from the

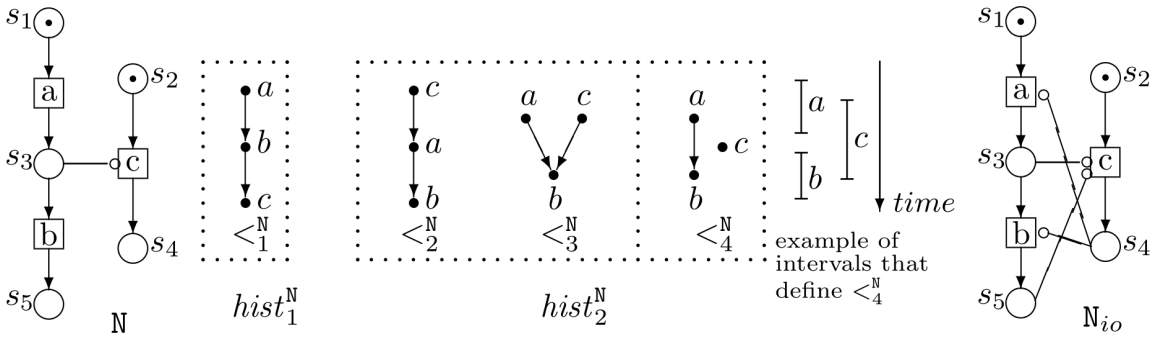


Figure 1.1: Inhibitor nets $\mathtt{N}$ and $\mathtt{N}_{io}$ and all their behaviours involving one occurrence of a, b and c. The net $\mathtt{N}$ generates $<_1^{\mathtt{N}}, <_2^{\mathtt{N}}, <_3^{\mathtt{N}}, <_4^{\mathtt{N}}$, and two concurrent histories, while $\mathtt{N}_{io}$ generates only an interval order $<_4^{\mathtt{N}}$. Partial orders are represented by Hasse diagrams. All orders except $<_4^{\mathtt{N}}$ are stratified.

marking $\{s_1, s_2\}$ and end at the marking $\{s_4, s_5\}$ is also *empty*, and the *only* observation/system run that starts from the marking $\{s_1, s_2\}$ and ends at the marking $\{s_4, s_5\}$ is the interval order $<_4^N$.

We will define a tool that, unlike standard semantics, allows to generate not only $<_1^{\mathtt{N}}, <_2^{\mathtt{N}}, <_3^{\mathtt{N}}$ but also the interval order $<_4^{\mathtt{N}}$.

1.2 Organization of Thesis

This thesis consist of ten chapters of which this the first. In the second chapter we provide some mathematical basics needed through out the thesis. Then, a brief overview of concurrent histories and order structures is given in chapter three. Chapter four is devoted for the background knowledge of Petri nets and given how big its body of knowledge, we only included the essentials. Then, we introduce one of our contributions which is *interval elementary net with inhibitor arcs* in chapter five. Chapters six and seven are about different semantics of Petri nets, namely process

semantics and trace semantics respectively. We then define our own process semantics for interval elementary net with inhibitor arcs in chapter eight under the title *interval processes and interval order structures*. In chapter nine we overview some of the other classes of Petri nets where the notion of interval orders has been used before the thesis is concluded in chapter ten.

Chapter 2

Mathematical Foundation

2.1 Partial, Total, Stratified and Interval Orders

In this chapter, a short introduction to partial orders (c.f. Fishburn (1985)), as they are the principal tool to describe executions and operational semantics of concurrent systems.

Definition 1. *A relation* $< \subseteq X \times X$ *is a (*strict*)* ***partial order*** *if it is irreflexive and transitive, i.e. for all* $a, c, b \in X$, $a \not< a$ *and* $a < b < c \implies a < c$. *We also define:* $a \frown_< b \stackrel{df}{\iff} \neg(a < b) \wedge \neg(b < a) \wedge a \neq b$, *and*

$$a <\!\!\frown b \stackrel{df}{\iff} a < b \vee a \frown_< b.$$

Note that $a \frown_< b$ *means* a *and* b *are* incomparable *(w.r.t.* $<$*) elements of* X. □

Let $<$ be a partial order on a set X. Then:

1. $<$ is ***total*** if $\frown_< = \emptyset$. In other words, for all $a, b \in X$, $a < b \ \vee \ b < a \ \vee \ a = b$;
2. $<$ is ***stratified*** if $a \frown_< b \frown_< c \Rightarrow a \frown_< c \ \vee \ a = c$, i.e., the relation $\frown_< \cup id_X$ is an equivalence relation on X;

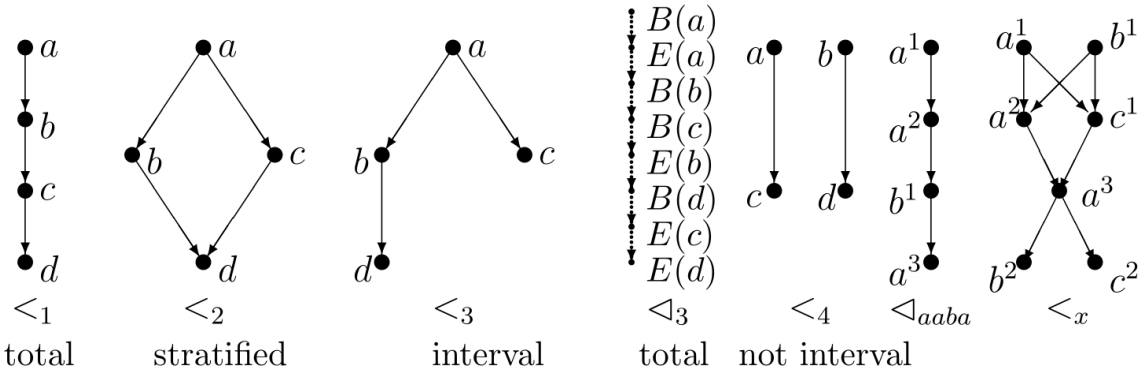


Figure 2.1: Various types of partial orders (represented as Hasse diagrams), the total order defined by the sequence *aaba*, and the stratified order defined by the step sequence $x = \{a, b\}\{a, c\}\{a\}\{b, c\}$.

3. $<$ is ***interval*** if for all $a, b, c, d \in X$, $a < c \wedge b < d \Rightarrow a < d \vee b < c$. In other words, $<$ is interval if all its four element restrictions are different from $<_4$ in Figure 2.1.

It is clear from these definitions that every total order is stratified and every stratified order is interval. Partial orders are usually represented as *Hasse diagrams*[1]. Figure 2.1 illustrates the above definitions. Every finite total order is uniquely represented by a sequence. For example the order $<_1$ of Figure 2.1 is represented by a sequence *abcd*. Similarly, every stratified order is uniquely represented by a step sequence. For example the order $<_2$ of Figure 2.1 is represented by a step sequence $\{a\}\{b, c\}\{d\}$. The opposite is also true as every sequence uniquely defines a total order if it is enumerated elements, and every step sequence uniquely defines a stratified order if it is enumerated elements.

Each sequence of events represents a total order of *enumerated events* in a natural way. For precise definitions see for example Janicki and Koutny (1995). Here we will

[1] A Hasse diagram of a partial order $<$ is the smallest relation R such that the transitive closure of R, i.e. R^+, is equal to $<$ (c.f. Fishburn (1985)).

be using the following notation.

Notation 1. *1. For each set of events* Σ*, let* $\widehat{\Sigma} = \{a^i \mid a \in \Sigma, i \geq 1\}$ *denote the set of* enumerated *events generated by* Σ*.*

2. For each sequence $x \in \Sigma^*$ *and each step sequence* $z \in (2^{\Sigma})^*$*, let* $\hat{x} \in \widehat{\Sigma}^*$ *and* $\hat{z} \in (2^{\widehat{\Sigma}})^*$ *denote their* enumerated representations*.*

For example, if $x = abbaa$ *then* $\hat{x} = a^1b^1b^2a^2a^3$*, and if* $z = \{a, b\}\{a, b, c\}\{a\}$ *then* $\hat{z} = \{a^1, b^1\}\{a^2, b^2, c^1\}\{a^3\}$*.*

3. For every sequence $x \in \Sigma^*$*,* $\lhd_x$ *is the* total order *defined by the enumerated sequence* $\hat{x}$*. For example:* $\lhd_{abbaa} = a^1 \to b^1 \to b^2 \to a^2 \to a^3$*.*

4. For every step sequence $z \in (2^{\Sigma})^*$*,* $\lhd_z$ *is the* stratified order *defined by the enumerated step sequence* $\hat{z}$*.*

For example: $\lhd_{\{a,b\}\{a,b,c\}\{a\}} = \{a^1, b^1\} \to \{a^2, b^2, c^1\} \to \{a^3\}$*.* □

The two orders on the far right of Figure 2.1 illustrate points (3) and (4) of the notation presented above. For the interval orders, the name and intuition follows from Fishburn's Theorem:

Theorem 1 (Fishburn (1970)). *A partial order* $<$ *on* X *is* ***interval*** *iff there exists a total order* $\lhd$ *on some* T *and two mappings* $B, E : X \to T$ *such that for all* $x, y \in X$*,*

1. $B(x) \lhd E(x)$*,*

2. $x < y \iff E(x) \lhd B(y)$*.* □

Usually $B(x)$ is interpreted as the beginning and $E(x)$ as the end of an *interval* x. The intuition of Fishburn's theorem is also illustrated in Figure 2.1 with $<_3$ and

$\lhd_3$. For all $x, y \in \{a, b, c, d\}$, we have $B(x) \lhd_3 E(x)$ and $x <_3 y \iff E(x) \lhd_3 B(y)$. For better readability we will skip parentheses in $B(x)$ and $E(x)$ in the future. Note that the interval order $<_3$ is (*not* uniquely) represented by a sequence that represents $\lhd_3$, i.e. $BaEaBbBcEbBdEcEd$. Fishburn's Theorem will be essential in interval semantics of inhibitor nets.

We will say that a total order $\lhd$ on X *extends* a partial order $<$ on X, if for all $x, y \in X$ $x < y \Rightarrow x \lhd y$, and for every partial $<$, $\mathsf{total}(<)$ denotes the set of all *total extensions* of $<$.

By Szpilrajn's Theorem Szpilrajn (1930), we know that every partial order $<$ is uniquely represented by the set $\mathsf{total}(<)$. Szpilrajn's Theorem can be stated as follows:

Theorem 2 (Szpilrajn (1930)). *For every partial order $<$,*

$$< \; = \bigcap_{\lhd \in \mathsf{total}(<)} \lhd,$$

i.e. each partial order is the intersection of all its total extensions. □

Chapter 3

Concurrent Histories and Order Structures

An important model for abstracting non-sequential behaviour of concurrent systems has been developed and proved to be sound in Janicki and Koutny (1993, 1997). In principle, the model assumes that concurrent behaviour is fully described by a triple $(X, \prec, \sqsubset)$, where X is the set of event occurrences, $\prec$ is *causality* (i.e. an abstraction of "earlier than"), and $\sqsubset$ is *weak causality* (i.e. an abstraction of "no later than"); both relations are on X.

3.1 Concurrent Histories

The abstraction model is based in three fundamental concepts: observations of concurrent behaviours, concurrent histories, and paradigms of concurrency. Following, we will briefly discuss those concepts.

A run *(observation, instance of concurrent behaviours)* is an abstract model of

the execution of a concurrent system. In Janicki and Koutny (1993), it was argued that *an observation must be an initially finite order that is either total, stratified, or interval.*

A *Concurrent history* is a complete set of equivalent runs. To elaborate, let's assume that all possible runs (observations) are total orders. A set $\Delta = \{abc, cba\}$ is not a concurrent history. This is because the intersection of the runs abc and cba (denoted by $<_\Delta$ or $<_{\{abc,cba\}}$) is the empty set which means that there is no causal relation between event a, b and c. This implies that bca is a possible run, for instance, but $bca \notin \Delta$; a contradiction. Now, let the Δ^{cl} be the set of all total extensions of $<_\Delta$ (i.e. $\{abc, bac, acb, bca, cab, cba\}$). $<_\Delta$ is complete and thus can be considered a concurrent history.

The above was in the case of all runs being total orders. However, when it is not the case, then a definition of concurrent histories require using more complex analysis of the runs. The set $<_\Delta$ can be viewed as an *invariant* characterizing the set Δ (i.e. all elements of Δ adheres to the ordering relation defined by $<_\Delta$). A concurrent history is a set of partial orders (of an appropriate type) with common domain that is fully characterized by all its *relational invariants* Janicki and Koutny (1993).

A *relational invariant* over a set of partial order Δ is any relation $R \subseteq X \times X$ defined by a formula of the type

$$(x, y) \in R \Longleftrightarrow \forall o \in \Delta.\ \phi_R(x, y, o),$$

where $\phi_R(x, y, o)$ is any propositional formula built from atoms $x \xrightarrow{o} y, y \xrightarrow{o} x, x \xleftrightarrow{o} y$ and $True$; for instance, $\phi_R(x, y, o) = x \xrightarrow{o} y \vee x \xleftrightarrow{o} y$. Note that $o \in O$ where O denote any partial order (i.e. total, stratified, interval, partial) and

$\longrightarrow_o$ instead of $<$ while $\longleftrightarrow_o = \sim$.

Let $RInv(\Delta)$ denote the set of all relational invariants generated by Δ, let O be a class of partial orders and let $\Delta \subseteq O$ be a set of partial orders with a common domain X.

We define the *closure of* Δ *with respect to* $RInv(\Delta)$ (Δ_O^{cl}) as the set of all partial orders in O with the domain X that satisfy:

$$o \in \Delta_O^{cl} \iff (\forall R \in RInv(\Delta). \forall x, y \in X. (x, y) \in R \Rightarrow \phi_R(x, y, o)).$$

Given all the above, a formal definition follows.

Definition 2 (Janicki and Koutny (1993)). *A set of runs* Δ *is a concurrent history in* O *iff* $\Delta = \Delta_O^{cl}$. □

In Janicki and Koutny (1993), it was shown that $RInv(\Delta)$ consists, at most, of eight different relations of which, at most, two are independent (i.e. cannot be calculated from each other using standard set theory operator union, intersection, and complement).

Lemma 1 (Janicki and Koutny (1993)).

1. $RInv(\Delta) = \{\emptyset, \Diamond_\Delta, \sqsubset_\Delta, \sqsubset_\Delta^{-1}, <_\Delta, <_\Delta^{-1}, \bowtie_\Delta, X \times X\}$.
2. $RInv(\Delta)$ *is the smallest set of relations containing* $\{\Diamond_\Delta, \sqsubset_\Delta\}$ *and closed under union, intersection, and complement and their inverse operators.*
3. $<_\Delta = \Diamond_\Delta \cap \sqsubset_\Delta$, $\bowtie_\Delta = \sqsubset_\Delta \cap \sqsubset_\Delta^{-1}$. □

The relation $<_\Delta$ is causality abstracting the "earlier than" relation ($x <_\Delta y$ means x is performed earlier than y in all observation from Δ). The relation $\sqsubset_\Delta$ is weak

causality which abstracts the "no later than relation" ($x \sqsubset_\Delta y$ means that x is performed no later than y in all observations in Δ; i.e. x is either performed earlier than or simultaneously with y). The relation $\lozenge_\Delta$ is commutativity which can be seen as an abstraction of "interleaving" or "not simultaneously" relation ($x \lozenge_\Delta y$ means that x and y are not performed simultaneously in any observation from Δ). Lastly, the relation $\bowtie_\Delta$ is synchronization and can be seen as an abstraction of simultaneity ($x \bowtie_\Delta y$ means that x and y are performed simultaneously in all observation from Δ).

3.2 Stratified and Interval Order Structures

When the system runs are represented by stratified or interval orders, or when we want to express not only "earlier than" but also "no later than" relationship, partial orders alone are not enough, we need to use pairs or relations called order structures (c.f. Janicki (2008); Janicki et al. (2010); Kleijn and Koutny (2004)).

In this section, we will define two concrete classes of structures which are used to model concurrent histories.

Definition 3 (Gaifman and Pratt (1987); Janicki and Koutny (1991)). *A **stratified order structure** is a relational structure $S = (X, \prec, \sqsubset)$ such that for all $a, b, c \in X$;*

S1. $a \not\sqsubset a$

S2. $a \prec b \Rightarrow a \sqsubset b$

S3. $a \sqsubset b \sqsubset c \wedge a \neq c \Rightarrow a \sqsubset c$

S4. $a \sqsubset b \prec c \wedge a \prec b \sqsubset c \Rightarrow a \prec c$ □

We will say that a *stratified order* $<$ on X *extends* the structure S, if $\prec \subseteq <$ and

$\sqsubset \subseteq <^{\frown}$. The set of all such extensions of S will be denoted by $strat(S)$. If $<$ is a stratified order on X, then the triple $(X, <, <^{\frown})$ is a stratified order structure. The axioms $S1$–$S4$ (see Definition 3) can be seen as an abstraction and generalization of the relationship between $<$ and $<^{\frown}$ when $<$ is a stratified order (c.f. Janicki and Koutny (1993, 1997)).

Theorem 2 (Szpilrajn Theorem) states that each partial order is uniquely represented by its set of total extensions. We have a similar relationship between stratified order structures and stratfied orders.

Theorem 3 (Janicki and Koutny (1997)). *For each stratified order structure* $S = (X, \prec, \sqsubset)$*, we have*

$$S = \Big(X, \bigcap_{< \in \mathsf{Strat}(S)} <, \bigcap_{< \in \mathsf{Strat}(S)} <^{\frown}\Big),$$

i.e. S is entirely defined by the set of all its extensions. □

The above theorem is a generalization of Szpilrajn's Theorem to stratified order structures (c.f. Janicki (2008); Kleijn and Koutny (2008)). It is interpreted as the proof of the claim that stratified order structures uniquely represent sets of equivalent system runs, provided that the system's operational semantics can be fully described in terms of stratified orders (see Janicki (2008); Janicki and Koutny (1997); Janicki et al. (2010); Kleijn and Koutny (2008) for details).

The formalism provided by interval order structures is more general than those provided by partial orders and stratified order structures. Interval order structures models concurrent behaviour that neither partial orders nor stratified order structures can model.

Definition 4 (Janicki and Koutny (1991); Lamport (1986)). *An **interval order structure** is a relational structure $S = (X, \prec, \sqsubset)$ such that for all $a, b, c, d \in X$:*

I1. $a \not\sqsubset a$

I2. $a \prec b \Rightarrow a \sqsubset b$

I3. $a \prec b \prec c \Rightarrow a \prec c$

I4. $a \prec b \sqsubset c \wedge a \sqsubset b \prec c \Rightarrow a \sqsubset c$

I5. $a \prec b \sqsubset c \prec d \Rightarrow a \prec d$

I6. $a \sqsubset b \prec c \sqsubset d \Rightarrow a \sqsubset d \vee a = d$ $\square$

Note that *every stratified order structure is also an interval order structure.*

We will say that an *interval order* $<$ on X *extends* the structure S, if $\prec \subseteq <$ and $\sqsubset \subseteq <^{\frown}$. The set of all such extensions of S will be denoted by $interv(S)$. If $<$ is an interval order on X, then the triple $(X, <, <^{\frown})$ is an interval order structure. The axioms $I1$–$I4$ can be seen as an abstraction and generalization of the relationship between $<$ and $<^{\frown}$ when $<$ is an interval order (c.f. Janicki and Koutny (1993, 1997)).

Theorem 3 states that each stratified order structure order is uniquely represented by its set of stratified extensions. We have the similar relationship between interval order structures and interval orders.

Theorem 4 (Janicki and Koutny (1997)). *For each interval order structure $S = (X, \prec, \sqsubset)$, we have*

$$S = \Big(X, \bigcap_{< \in \mathsf{Interv}(S)} <, \bigcap_{< \in \mathsf{Interv}(S)} <^{\frown}\Big),$$

i.e. S is entirely defined by the set of all its extensions.

The above theorem is a generalization of Szpilrajn's Theorem to interval order structures, similarly as Theorem 3 is a generalization of Szpilrajn's Theorem to stratified order structures. It shows that if the system's operational semantics is fully

described in terms of interval orders, then the interval order structures uniquely represent sets of equivalent system runs (see Janicki (2008); Janicki and Koutny (1997) for details).

In both order structures the relations $\prec$ and $\sqsubset$ are called *causality* and *weak causality* respectively, and in both models $\prec$ is an abstraction of "earlier than" relation while $\sqsubset$ is an abstraction of "no later than" relation. In both models $\prec$ is always a partial order, while $\sqsubset$ does not have to be. The fundamental difference is that for Stratified Order Structures the system runs/executions are assumed to be modeled by at most stratified orders, while for Interval Order Structures the system runs/executions are assumed to be modeled by general interval orders.

For interval order structures we have the following equivalent of Fishburn Theorem which is one of the main tool that will be used in our model Abraham et al. (1990).

Theorem 5 (Abraham et al. (1990)).
A triple $S = (X, \prec, \sqsubset)$ is an interval order structure if and only if there exists a partial order $<$ on some Y and two mappings $B, E : X \to Y$ such that $B(X) \cap E(X) = \emptyset$ and for each $x, y \in X$:

1. $Bx < Ex$,
2. $x \prec y \iff Ex < By$,
3. $x \sqsubset y \iff Bx < Ey$ □

The partial order $<$ from Theorem 5 is not unique and does not need to be interval. For more on the theory of order structures and their applications, the reader is referenced to Janicki (2008); Janicki et al. (2010); Janicki and Koutny (1997).

There is a connection between order structures (Section 3.2) and concurrent histories. Formally, for every set of observations (on the same set of event occurrences X) Δ, define: $\prec_\Delta = \bigcap_{<\in\Delta} <$ and $\sqsubset_\Delta = \bigcap_{<^\frown\in\Delta} \sqsubset$. Also, define $S_\Delta = \{X, \prec_\Delta, \sqsubset_\Delta\}$. Then, it was shown in Janicki and Koutny (1993) that

- if Δ consists of only total orders, then Δ is a concurrent history if and only if $\Delta = total(\prec_\Delta)$,
- if Δ consists of only stratified orders, then Δ is a concurrent history if and only if $\Delta = Strat(S_\Delta)$, and
- if Δ consists of only interval orders, then Δ is a concurrent history if and only if $\Delta = Interv(\prec_\Delta)$.

For example, lets $\Delta = \{<_1^{\mathtt{N}}, <_2^{\mathtt{N}}, <_3^{\mathtt{N}}\}$ and $\Delta' = \{<_1^{\mathtt{N}}, <_2^{\mathtt{N}}, <_3^{\mathtt{N}}, <_4^{\mathtt{N}}\}$ where $<_1^{\mathtt{N}}, <_2^{\mathtt{N}}, <_3^{\mathtt{N}}$, and $<_4^{\mathtt{N}}$ are from Figure 1.1. Then, Δ and Δ' are concurrent histories as $Strat(S_\Delta) = \Delta$ and $Interv(S_{\Delta'}) = \Delta'$.

Chapter 4

Petri Nets

In this chapter an overview of all different nets that are used in our research is given to a varying degree of details. It starts with an informal introduction to Elementary nets (EN) which form the most fundamental class of Petri nets. Then, a more formal and detailed overview of the elementary nets with Inhibitor arcs (ENI) is given. In addition, we will briefly introduce activator nets.

4.1 Elementary Nets

Elementary nets (EN) are the most fundamental class of Petri nets. The basic idea of a net-based model is that it consists of a net (structure) and a set of "token game" rules. The net describes the static structure of the concurrent system while the token game describe the dynamic behaviour of such a system. Different classes of Petri nets differs in the underlying structure, the dynamic rules, or in both of them.

The token game rules are primarily concerned with how to get from one global state of the net to the other. Therefore, those rules define the potential state space of

the system. However, in order to get the actual state space, an initial global state of the system must be specified. Then, starting from this initial global state, the state space can then be obtained based on the token game rules. Hence, a given elementary net system is given by a net and an initial global state (called initial configuration or marking).

A defining characteristic of Petri nets is that it is the local states that form a global state and it is through local transition(s) acting on those local states that a change from global state to another global state can occur. In other words, subset of the local states forming the global space is replaced by another subset of local states as a result of local transition firing.

The local states of an EN are called *places* and local transition are called *transitions*. The net is a finite directed graph with its nodes being places (drawn as circles) and transitions (drawn as boxes) while its edges assigning inputs and outputs to each transition. Thus, the net defines which local states are replaced by other local states for each transition. The actual global state is indicated by putting tokens (drawn as thick dots) in the places forming the local state and the token game is played by moving those token based on the transitions that are firing.

Following, we formalize the above concepts.

4.1.1 Formal Definition of Elementary Nets

The static structure of an EN system (i.e. the underlying net) is the main part of the system. There are a number of possible equivalent definitions of EN that differs mainly in notations; following is one of them.

Definition 5. *An **elementary net (EN)** is a tuple $EN = (P, T, F)$ such that*

1. *P and T are finite disjoint sets of places and transitions;*
2. *$F \subseteq (P \times T) \cup (T \times P)$ is the flow relation of EN ($dom(F) \cup range(F) = P \cup T$);*
3. *$C_{init} \subseteq P$ is the initial marking/configuration of EN (generally, any $C \subseteq P$ is a marking);* □

The net in Figure 4.1 is an example of an EN. Each transition $t \in T$ has a neighbouring places with which it is connected by incoming and outgoing arrows. It is those neighbours which determine the firing rules of a transition as well as the effect of such a firing. For every $x \in P \cup T$, we define its input set (or preset) ${}^\bullet x = \{y \mid (y, x) \in F\}$ and its output set (or post set) $x^\bullet = \{y \mid (x, y) \in F\}$. A transition with no input places is called a *source transition*, and one with no output places is called a *sink transition*. Moreover, a net is said to have a *self-loop* if for a pair of a place p and a transition t, p is both an input place and an output place of t. A net is said to be pure if it has no self-loops. The input and output definitions are extended to subsets of transitions as for any subset $U \subset T : {}^\bullet U = \bigcup_{t \in U} {}^\bullet t, U^\bullet = \bigcup_{t \in U} t^\bullet$. For EN, it is assumed that for every $t \in T, {}^\bullet t \neq \emptyset \neq t^\bullet$ and ${}^\bullet t \cap t^\bullet = \emptyset$.

For `EN` in Figure 4.1, the input and output of any transition or place can be defined, for example, ${}^\bullet t_1 = \{s_1, s_2\}$, and ${}^\bullet s_3 = \{t_1\}$, etc; while $t_1^\bullet = \{s_3, s_4\}$, and $s_3^\bullet = \{t_4, t_2\}$, etc.

While the structure of EN describe its static features, the dynamics of an EN is defined through the set of *markings or configurations* (i.e. a subset of places representing conditions holding at any given global situation). Those conditions are represented as tokens.

Once a marking is known, the firing rules of a transition can be defined. The most basic rule for a transition to fire is that its input has a token and its output does not

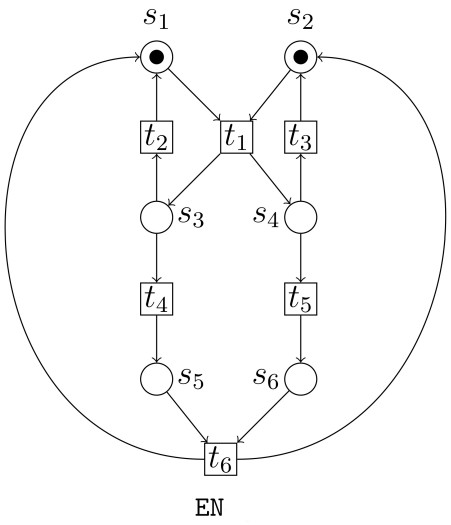


Figure 4.1: A simple example of an elementary net.

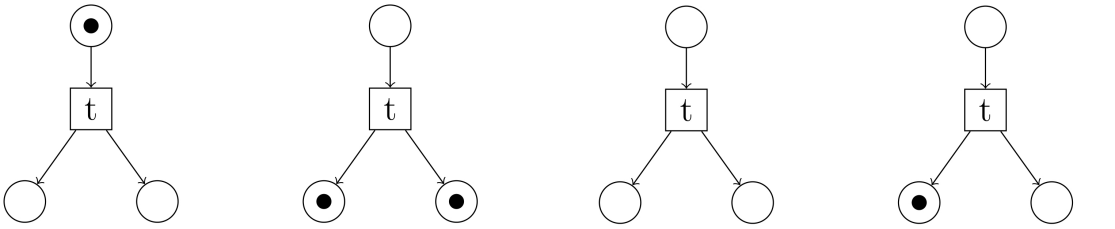
have a token. When a transition has fired, it is said that it has consumed a token from its input place(s) and has produced token(s) in its output place(s). Formally;

Definition 6. *A **transition** t **is enabled** (i.e. can fire) at a marking C, if*

1. ${}^\bullet t \in C$; *2.* $t^\bullet \notin C$

The notation $C[t\rangle$ is used to denote that transition t is enabled at marking C. We will also write $C[t_1 \dots t_n\rangle C'$ if $C[t_1\rangle C_1 \dots C_{n-1}[t_n\rangle C'$ for some configurations $C_1, \dots, C_{n-1}$. From Definition 6 and given the initial marking $C_{init} = \{s_1, s_2\}$ for EN (Figure 4.1), one can see the following $C_{init}[a\rangle$ and $C_{init}[c\rangle$ (i.e. transitions a and c are enabled at C_{init}). Note that a source transition is always enabled, and that the firing of a sink transition consumes tokens, but does not produce tokens. When a transition (or more than one transition) occur, the global state of the entire system is changed. This is captured through the new configuration, denoted as C', which the firing(s) lead to. C' is defined simply as $C' = (C \setminus {}^\bullet t) \cup t^\bullet$ showing which tokens have

been consumed and which ones have been produced. See Figure 4.2 for an illustration of firing rules and local changes of states.



(a) t is enabled ($[t\rangle$). (b) t has fired. (c) t is not enabled. (d) t is not enabled.

Figure 4.2: An illustration of local changes of states and firing rules.

4.1.2 Operational Semantics of Elementary Nets

The rules of transitions occurrences can be generalized to be applied not only to individual transitions but also sequences of transitions as well as sequences of set of concurrently enabled transitions. The *operational semantics* of EN is defined through the token game which simulates the occurrence of transitions and the changes of tokens in places. There are two standard operational semantics for ENI, one in terms of **firing sequences** and another in terms of **firing step sequences**.

Definition 7. *A **firing sequence** of an EN is any sequence of transitions $t_1, \ldots, t_n$ for which there are configurations $C_1, \ldots, C_n$ satisfying:*

$$C_{init}[t_1\rangle C_1[t_2\rangle C_2 \ldots [t_n\rangle C_n. \qquad \square$$

The definition above can be generalized to sequences of sets of transitions occurring simultaneously. Let $U \subseteq T$ be a non-empty set such that for all distinct $t_1, t_2 \in U$:

$$(t_1^\bullet \cup {}^\bullet t_1) \cap (t_2^\bullet \cup {}^\bullet t_2) = \emptyset.$$

Then U is enabled at a marking C if ${}^\bullet U \subseteq C$ and $U^\bullet \cap C = \emptyset$. We also denote this by $C[U\rangle C'$ where, $C' = (C \setminus {}^\bullet U) \cup U^\bullet$.

Definition 8. *A **firing step sequence** is a sequence of sets (or steps) $U_1, \ldots, U_n$ for which there are configuration $C_1, \ldots, C_n$ satisfying:*

$$C_{init}[U_1\rangle C_1[U_2\rangle C_2 \ldots [U_n\rangle C_n. \qquad \square$$

Note that, firing sequence can be seen as a special case of the more general definition of firing step sequences (i.e. each step has only one transition).

For `EN` in Figure 4.1, we can define a firing sequences (among infinitely many) $x = \{t_1 t_4 t_5 t_6 t_1\}$ as follows:

$$C_{init}[t_1\rangle\{s_3, s_4\}[t_4\rangle\{s_4, s_5\}[t_5\rangle\{s_5, s_6\}[t_6\rangle\{s_1, s_2\}$$

Additionally, a step firing sequence (again among infinitely many)
$y = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4, t_5\}\}$ as follows:

$$C_{init}[\{t_1\}\rangle\{s_3, s_4\}[\{t_2, t_3\}\rangle\{s_1, s_2\}[\{t_1\}\rangle\{s_3, s_4\}[\{t_4, t_5\}\rangle\{s_5, s_6\}$$

Generally speaking, the theory of Petri nets does not assume anything about the nature of transitions; thus, they can be treated as instantaneous and non-instantaneous (i.e. last for a duration of time) Petri (1996); Reisig (2013); Desel and Reisig (1998). If firing sequences semantics is of interest like in the popular paper Murata (1989) then the distinction between instantaneous or non-instantaneous transitions is often negligible. However, such distinction cannot be neglected in the case of step sequence semantics Baldan et al. (2004); Janicki and Koutny (1995); Kleijn and Koutny (2008) or interval orders Janicki and Koutny (1997).

4.1.3 Fundamental Situations in Dynamic Systems Modeling

Having defined the behaviours of concurrent systems as firing sequences and firing step sequences allow us to discuss some of the complex relationship between transitions. This will be illustrated in the study the three fundamental relationships which could hold between transitions t_1 and t_2 at any configuration of EN; namely, *concurrency, causality, and conflict.*

Concurrency: the first observation is that when, at some configuration, two transitions can occur simultaneously, it does not mean that they have to. They can still occur one after another in *any order* which in turns mean that if they can occur in any order, they must be concurrently enabled and non-interfering. This is formulated as what is called "diamond property" which is formally defined as follows:

Fact 1. *Let $N = (P, T, F, C_{init})$ be an EN system, let $C, C' \subseteq P$, and let $U \subseteq T$. Let $\{U_1, U_2\}$ be partitions of U, i.e. $U = U_1, U_2, U_1 \cap U_2 = \emptyset$ and $U_1, U_2 \neq \emptyset$. If $C[U\rangle C'$, then there exist a marking $C'' \subseteq P$ such that $C[U_1\rangle C''$ and $C''[U_2\rangle D$.* □

The reason for its name is that in diagrams (reachability graphs for example), this case yield a diamond shape. For more on the diamond properties, readers can refer to Hoodgeboom and Rozenberg (1991) Rozenberg and Engelfriet (1998) and Kleijn and Koutny (2008). Moreover, the notion of concurrency between two transition t_1 and t_2 is illustrated in Figure 4.3.

It is worth mentioning here that the case in Figure 4.3 is a part of a bigger picture where there may be more than two transitions which can occur concurrently. Figure 4.3 also lacks the illustration of how ${}^\bullet t_1, {}^\bullet t_2$ and $t_1^\bullet, t_2^\bullet$ fits at a given configuration but it shows the essence of concurrency.

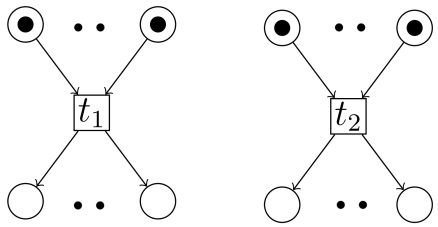
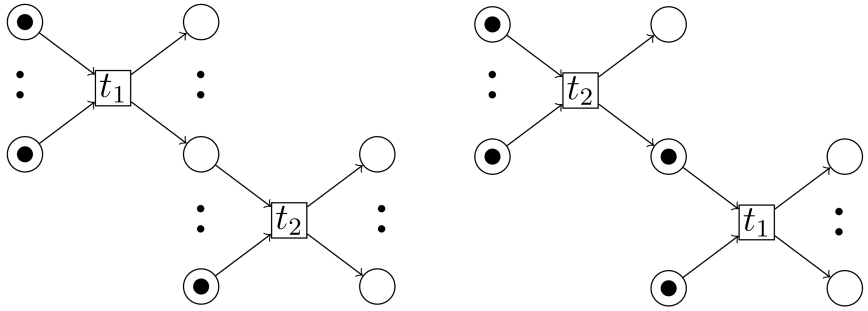


Figure 4.3: Concurrency of transition t_1 and t_2.

Causality: the relation between two transitions is said to be causal, if one transition must occur in order for the other to be enabled. Intuitively, this can happen in one of two cases: the first case is when the first transition t_1 produces a token which populates the input place of a transition t_2 given that there is only one input place and/or all other places have tokens on them (Figure 4.4(a)); the second case is when transition t_1 consumes a token that populates one of t_2 output places (Figure 4.4(b)).

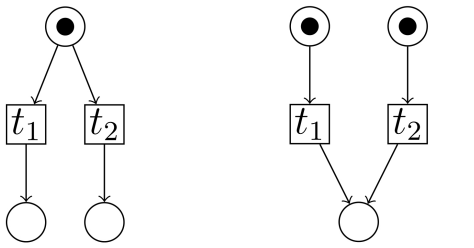


(a) $t_1^\bullet \cap {}^\bullet t_2 \neq \emptyset$. (b) ${}^\bullet t_1 \cap t_2^\bullet \neq \emptyset$.

Figure 4.4: Causality of transition t_1 and t_2.

Conflict: two transitions, t_1, t_2 are said to be in conflict when at some configuration C, we have $C[t_1\rangle$ and $C[t_2\rangle$ but not $C[\{t_1, t_2\}\rangle$. This can be the result of either t_1 and t_2 share a common input place so the firing of either one consumes a token; therefore, disabling the other or that t_1 and t_2 share a common output place and again firing one disables the other. This is presented in Figure 4.5. A conflict is

called input-conflict (or backward conflict) when ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$, and is called output-conflict (or forward conflict) when $t_1^\bullet \cap t_2^\bullet \neq \emptyset$. Both types of conflicts can also be present together.



(a) ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$. (b) $t_1^\bullet \cap t_2^\bullet \neq \emptyset$.

Figure 4.5: Conflict between transition t_1 and t_2.

It is interesting to take a close look at concurrency and conflict in the sense that in both cases, we have t_1 and t_2 enabled at some configuration. However, In case of concurrency t_1 and t_2 are independent, while in the case of conflict they are not. This leads to nondeterministic choice between transitions in case of conflict which is not present in concurrency. Intuitively, EN where there is no choice are easier to understand and work with.

Definition 9. *An EN is* ***conflict-free*** *if for every reachable configuration C and every transitions t_1 and t_2, it is the case that if $C[t_1\rangle$ and $C[t_2\rangle$, then $C[\{t_1, t_2\}\rangle$.* □

Conflict-freeness is an important characteristic that will be used later in this research.

4.1.4 Contact Freeness

Equivalence notion is important to any theory of systems, especially when some modification of the system are desirable but its behaviours must not change. This is true

for elementary nets theory as well. Those changes to define "better" equivalent systems results in what is called normal forms. Although there are a number of normal forms, we only define the one that is used later in our research (when defining the notion of non-sequential processes generated by EN) and that is *c*ontact-free. The importance of such a normal form arises from the fact that one often deals with only contact-free nets in the theory of elementary nets Pomello et al. (1992). In addition, for every EN, there exist an equivalence contact-free EN system Rozenberg and Thiagarajan (1986).

Definition 10. *An EN is* ***contact-free*** *if for every reachable configuration C and every transition t, it is the case that ${}^\bullet t \subseteq C$ implies $t^\bullet \cap C = \emptyset$.* □

In other words, while in the definition of elementary nets presented earlier, the firing rule for a transition checks explicitly for its output(s) emptiness, checking its input(s) is enough when dealing with contact-free nets. Transforming an EN to a contact-free net is simple and require only introducing **complementary places** to the original net.

Definition 11 (Goltz and Reisig (1983)). *Places $p, q \in P$ are* ***complementary*** *(p is a complement of q and vice versa) if $p \neq q$, ${}^\bullet p = q^\bullet$ and $p^\bullet = {}^\bullet q$, and $|C_{init} \cap \{p, q\}| = 1$.* □

If p and q are complementary we will write $p = \widetilde{q}, q = \widetilde{p}$, and clearly $p = \widetilde{\widetilde{p}}, q = \widetilde{\widetilde{q}}$.

Thus far, we have covered the basics of the basic class of Petri nets and we will now introduce a simple yet very powerful extension to Petri nets, namely inhibitor arcs.

4.2 Elementary Nets with Inhibitor Arcs

Elementary Nets with inhibitor Arcs (ENI) differs from the classical elementary nets only in the introduction of *inhibitor arcs* which was introduced in Agerwala and Flynn (1973). Inhibitor arcs (drawn as an arc with a circle head) allows to check for the *absence* of a token in the place to which it is connected. ENI extends EN in the way that notion and notations concerning the structure and the token game with only one notation introduced as a result of introducing inhibitor arcs. That is $^\circ t$ which denotes the set of all places $p \in P$ where the presence of a token "inhabits" the firing of a transition t to which it is connected.

Definition 12. *An **elementary net system with inhibitor arcs (ENI)** is a tuple $\{P, T, F, C_{init}, I\}$ such that:*

- *P and T are finite disjoint sets of places and transitions;*
- *$F \subseteq (P \times T) \cup (T \times P)$ is the flow relation of EN ($dom(F) \cup range(F) = P \cup T$);*
- *$C_{init} \subseteq P$ is the initial marking/configuration of EN (generally, any $C \subseteq P$ is a marking); and*
- *$I \subseteq P \times T$ is the set of inhibitor arcs.* □

Note that the first four components are the underlying elementary net (see Defintion 5).

For example, consider the ENI in Figure 4.6 which is the same as the EN in Figure 4.1 with the addition of an inhibitor arc between s_5 and transition t_5. This means when there is a token in place s_5, transition t_5 is not enabled. The inhibitor

arc defines the *"no later than"* relation and in this example, t_5 must occur no later than t_4 as the firing of t_4 inhibits the firing of t_5.

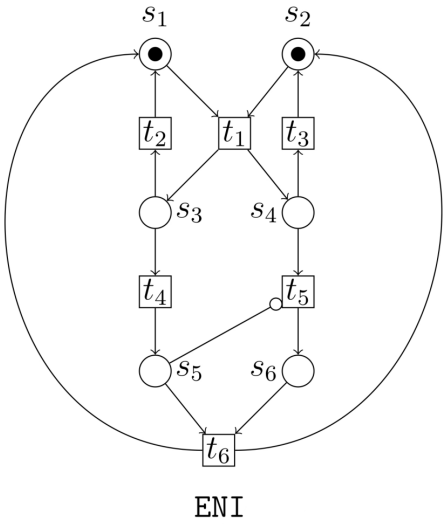


Figure 4.6: An example of an elementary net with inhibitor arc.

Intuitively the transition firing rules and subsequently the definition of firing sequences and firing step sequence of ENI extends that of EN while addressing the addition of inhibitor arcs ($^\circ t$ denotes the set of places concocted to t with inhibitor arcs). A transition t_1 is enabled at marking C if $^\bullet t_1 \subseteq C$ and $(^\circ t_1 \cup t^\bullet) \cap C = \emptyset$. Similarly to EN, there are two standard operational semantics for ENI, one in terms of **firing sequences** and another in terms of **firing step sequences** (c.f. Chiola et al. (1991); Janicki and Koutny (1995)).

A ***firing sequence*** of an ENI is any sequence of transitions $t_1, \ldots, t_n$ for which there are markings $C_1, \ldots, C_n$ satisfying:

$$C_{init}[t_1\rangle C_1[t_2\rangle C_2 \ldots [t_n\rangle C_n.$$

The definition above can be generalized to sequences of sets of transitions occurring

simultaneously. Let $U \subseteq T$ be a non-empty set such that for all distinct $t_1, t_2 \in U$:

$$(t_1^\bullet \cup {}^\bullet t_1) \cap (t_2^\bullet \cup {}^\bullet t_2) = \emptyset.$$

Then U is enabled at a marking C if ${}^\bullet U \subseteq C$ and $(U^\bullet \cup {}^\circ U) \cap C = \emptyset$. We also denote this by $C[U\rangle C'$ where, $C' = (C \setminus {}^\bullet U) \cup U^\bullet$.

A ***firing step sequence*** is a sequence of sets (or steps) $U_1, \ldots, U_n$ for which there are markings $C_1, \ldots, C_n$ satisfying:

$$C_{init}[U_1\rangle C_1[U_2\rangle C_2 \ldots [U_n\rangle C_n.$$

Examining the firing sequence $x = \{t_1 t_4 t_5 t_6 t_1\}$ and the firing step sequences $y = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4, t_5\}\}$ defined for the `EN` in Figure 4.1 shows that x is not a firing sequence of the `ENI` in Figure 4.6 while the firing step sequence y is a step sequence of `ENI`. This is because when the transition t_4 fires, it populates its output place s_5 with a token which then inhibits the firing of t_5. This means that t_5 must fire "no later than" t_4 which is possible to happen when either t_5 fires before or concurrently with t_4. The latter is the case in y where $\{t_4, t_5\}$ occurs in one step.

Firing sequence semantics is sometimes called 'a-posteriori' while firing step sequence semantics is sometimes called 'a-priori' (see Chiola et al. (1991); Janicki and Koutny (1995)). It is often assumed that if the events (transitions) are interpreted as representations of activities whose completion takes some time, then 'a-priori' model is frequently preferable, however if the events (transitions) are instantaneous, i.e. their occurrence takes zero time, then simultaneous executions must be excluded Janicki (2008); Janicki and Koutny (1995), so only firing sequence approach remains.

Assume that when a non-instantaneous transition t begins its firing, the tokens

that it consumes disappear for consumption by other transitions but continue to inhibit other transitions, if connected to an inhibitor arc. Such interpretation allows possible executions where transitions overlap, so they can be interpreted as interval orders. This assumption is crucial to the development of the *interval elementary net with inhibitor arcs* which will be the focus of Chapter 5.

4.3 Activator Nets

Activator arcs (also called 'read', or 'contextual' arcs Baldan et al. (2004); Montanari and Rossi (1995)), formally introduced in Janicki and Koutny (1995); Montanari and Rossi (1995), are conceptually orthogonal to the inhibitor arcs, they allow a transition to check for a *presence* of a token.

Definition 13. *An **Activator Net** is a tuple* $\{P, T, F, C_{init}, A\}$ *such that:*

- P *and* T *are finite disjoint sets of places and transitions;*
- $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation of EN* ($dom(F) \cup range(F) = P \cup T$)*;*
- $C_{init} \subseteq P$ *is the initial marking/configuration of EN (generally, any* $C \subseteq P$ *is a marking); and*
- $A \subseteq P \times T$ *is the set of activator arcs.* □

For example, consider the net in Figure 4.7 which is the same as the ENI in Figure 4.6 with the an activator arc replacing the inhibitor arc between s_5 and transition t_5. This means that in order for t_5 to fire, a token must be presented in s_5. Formally, the set of activator places (places to which a transition is connected via an activator

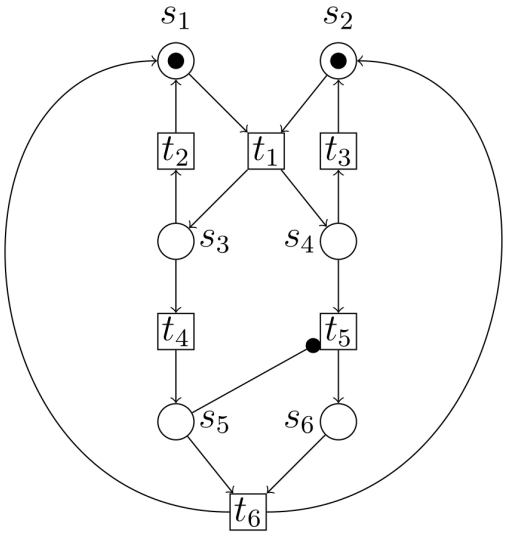


Figure 4.7: An example of an activator net.

arc) is denoted by ${}^{\blacklozenge}t$. Thus a transition t_1 is enabled at marking C if, in addition to rules defined in Definition 6, ${}^{\blacklozenge}t_1 \in C$.

Moreover, the firing sequence and step sequences are defined accordingly. A ***firing sequence*** of an activator net is any sequence of transitions $t_1, \ldots, t_n$ for which there are markings $C_1, \ldots, C_n$ satisfying:

$$C_{init}[t_1\rangle C_1[t_2\rangle C_2 \ldots [t_n\rangle C_n.$$

The definition above can be generalized to sequences of sets of transitions occurring simultaneously. Let $U \subseteq T$ be a non-empty set such that for all distinct $t_1, t_2 \in U$:

$$(t_1^{\bullet} \cup {}^{\bullet}t_1) \cap (t_2^{\bullet} \cup {}^{\bullet}t_2) = \emptyset.$$

Then U is enabled at a marking C if ${}^{\bullet}U \subseteq C$, ${}^{\blacklozenge}U \subseteq C$, and $U^{\bullet} \cap C = \emptyset$. We also denote this by $C[U\rangle C'$ where, $C' = (C \setminus {}^{\bullet}U) \cup U^{\bullet}$.

In this chapter, a general introduction covering the basic of Petri net (more specifically elementary net with and without inhibitor arcs) was provided. We have included what is necessary for this thesis to be self contained, and given the large body of literature on Petri nets, it is minimal.

It is also worth mentioning here that we only introduce the elementary class of Petri nets. There are higher classes of Petri nets where they differ in the number of tokens allowed in every place (e.g. Place/Transition Petri nets). There are also classes of Petri nets where individual tokens can be distinguished; this class is called Colored Petri nets.

Chapter 5

Interval Elementary Net with Inhibitor Arcs

This chapter contains the first part of our contribution. We defined an *interval representation* of ENI that allows for recording observation as interval orders.

Since every interval order of events can be represented by some total order (i.e. an appropriate sequence) of event beginnings and ends (Theorem 1 by Fishburn), if we figure out how a given inhibitor net can generate appropriate sequences of event beginnings and ends, we might be able to describe all interval orders the net generates.

In the approach used in our research, we assume that the events (transitions) in the ENI systems are not instantaneous. On the contrary, they are interpreted as representations of activities whose completion takes some time. However, their beginnings and ends are instantaneous.

If inhibitor arcs are not involved (i.e. the case of elementary nets), to represent transitions by their beginnings and ends, we might just replace each transition $\boxed{t}$ by the net $\boxed{Bt} \rightarrow \textcircled{t} \rightarrow \boxed{Et}$ as proposed for example in Zuberek (1980) for Timed Petri

nets. However, adding inhibitor arcs presents a challenge which will be dealt with in this chapter.

The basic idea of defining the set of firing interval sequences i.e. sequences of beginnings and ends (will be formally defined shortly) for a given inhibitor net N is briefly presented in Figure 5.1 by the transformation of the net N into the net $\mathcal{N}$.

In Figure 5.1, while the inhibitor arc (s_3, Bc) in the net $\mathcal{N}$ is obvious, the inhibitor arc (b, Bc) is not. It is true that $\mathcal{N}_{invalid}$ (the far right net in Figure 5.1) represents a more natural transformation. However, since a token in s_3 of N prevents c from being enabled while a token in s_3 of $\mathcal{N}_{invalid}$ prevents the *starting* of c from being enabled, $\mathcal{N}_{invalid}$ does not work. Take for example the interval sequences $BaEaBbBcEcEb$ and $BaEaBbBcEbEc$, although they are valid sequences for $\mathcal{N}_{invalid}$ from $\{s_1, s_2\}$ to $\{s_4, s_5\}$, they define a stratified order corresponding to the step sequence $\{a\}\{b, c\}$ which cannot be generated by N. Moreover, the interval sequences $BcBbEcEb$ and $BcBbEbEc$ that also generate the stratified order corresponding to the step sequence $\{a\}\{b, c\}$ are *not* firing sequences of $\mathcal{N}_{invalid}$.

One the other hand, $\mathcal{N}$ (the middle net in Figure 5.1) seems to have all the desired properties as the inhibitor arc (b, Bc) prevents the execution of Bc before Eb (assuming that Ea has occur and Bc has not). In other order, after a fires, if c has not started firing yet, then c cannot start until b ends; this is what we want as it confirms with the semantics of the original ENI.

We *assume* that the net $\mathcal{N}$ fully describes the behaviour of the net N and later provide formal justification of this claim.

Below we provide a formal transformation of elementary nets with inhibitor arch into their interval representations.

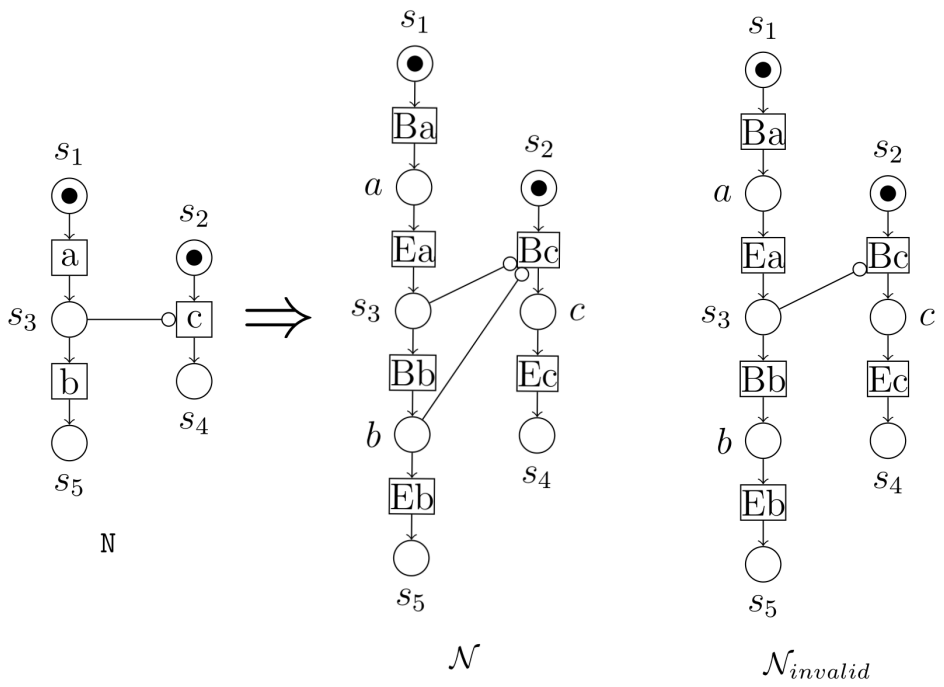


Figure 5.1: ENI N from Figure 1.1 and its interval representation $\mathcal{N}$.

Definition 14 (Transforming ENI into its interval representation).

Let $\mathsf{N} = (P, T, F, I, C_{init})$ *be an ENI system. We define* $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$*, its* ***interval representation*** *as follows:*

1. $\mathcal{P} = P \cup T$
2. $\mathcal{T} = \{Bt \mid t \in T\} \cup \{Et \mid t \in T\}$
3. $\forall\ p \in P.\ t \in T.\ \ (p,t) \in F \iff (p, Bt) \in \mathcal{F}$
4. $\forall\ p \in P.\ t \in T.\ \ (t,p) \in F \iff (Et, p) \in \mathcal{F}$
5. $\forall\ t \in T.\ \ (Bt, t), (t, Et) \in \mathcal{F}$
6. $\forall\ p \in P.\ t \in T.\ \ (p,t) \in I \iff (p, Bt) \in \mathcal{I} \wedge (\forall r \in p^{\bullet}.\ (r, Bt) \in \mathcal{I}).$ □

Note that each of the sequences $BaBcEaBbEbEc$, $BaBcEaBbEcEb\,BcBaEaBbEbEc$, and $BcBaEaBbEcEb$, for example, are *firing sequences* of $\mathcal{N}$, and each of them represents the *interval order* $<_4^{\mathtt{N}}$ from Figure 1.1 via Fishburn Theorem (Theorem 1). *This means that event b follows event a and event c overlaps both events a and b in the original net* $\mathtt{N}$.

Directly from the above definition we have the following convenient result.

Fact 2. *Let* $\mathsf{N} = (P, T, F, I, C_{init})$ *be an ENI system and* $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$ *its interval representation. Then for each* $t \in T$ *we have:* ${}^\bullet Bt = {}^\bullet t$, $Bt^\bullet = \{t\}$, ${}^\bullet Et = \{t\}$, $Et^\bullet = t^\bullet$, $^\circ Bt = {}^\circ t \cup ({}^\circ t)^\bullet$, *and* $^\circ Et = \emptyset$. □

Since $\mathcal{N}$ is just another inhibitor net, we may use the standard definition of a firing sequence from Section 4.2, but with the following caveat: not every sequence from $\mathcal{T}^*$ can be interpreted as an interval order, for example $BaBcBb$ represents no interval order.

Let $\mathcal{D} \subseteq \mathcal{T}$ and let $s \in \mathcal{T}^*$. We define the projection of s onto $\mathcal{D}$ standardly as:

$$\pi_{\mathcal{D}}(\epsilon) \stackrel{df}{=} \epsilon, \quad \pi_{\mathcal{D}}(s\alpha) \stackrel{df}{=} \begin{cases} \pi_{\mathcal{D}}(s)\alpha & \text{if } \alpha \in \mathcal{D}, \\ \pi_{\mathcal{D}}(s) & \text{if } \alpha \notin \mathcal{D}. \end{cases}$$

For example $\pi_{\{Ba,Ea\}}(BbBaEbBaEaEc) = BaBaEa$ and

$\pi_{\{Ba,Ea,Bc,Ec\}}(BbBaEbBaEaEc) = BaBaEaEc$.

Definition 15. *We say that a string* $x \in \mathcal{T}^*$ *is an* **interval sequence** *iff*

$$\forall Bt, Et \in \mathcal{T}^*.\ \pi_{\{Bt,Et\}}(x) \in (BtEt)^*.$$

We use $\mathsf{InSeq}(\mathcal{T}^*)$ *to denote the set of all interval sequences of* $\mathcal{T}^*$. □

Definition 16. *Let $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$ be an interval representation of ENI. A sequence $x = \alpha_1 \dots \alpha_n \in \mathcal{T}^*$ is an **interval firing sequence** of $\mathcal{N}$ if there are configurations $C_1, \dots, C_n$ such that $C_1, C_n \in P$ (i.e. $C_1 \cap T = C_n \cap T = \emptyset$) and*

$$C_{init}[\![\alpha_1\rangle\!\rangle C_1 [\![\alpha_2\rangle\!\rangle C_2 \dots [\![\alpha_n\rangle\!\rangle C_n. \qquad \square$$

To improve readability, we use $[\![\alpha\rangle\!\rangle$ to denote firing transition α in some configuration of the net $\mathcal{N}$.

For example, for $\mathcal{N}$ from Figure 5.1, we have $\{s_1, s_2\}[\![BcBaEaBbEcEb\rangle\!\rangle\{s_4, s_5\}$.

The following result validates the above definition.

Proposition 1. *If x is an interval firing sequence of $\mathcal{N}$, then $x \in \mathsf{InSeq}(\mathcal{T}^*)$.*

Proof. We have to show that for each $a \in T$, $\pi_{\{Ba,Ea\}}(x) \in (BaEa)^*$.

Let $x = y\, Ba\, z$ and $C_{init}[\![y\, Ba\rangle\!\rangle C'$. Since $Ba^\bullet = \{a\}$, then $a \in C'$. We also have: for any $C_a \subseteq P \cup T$, if $a \in C_a$, then Ba is not enabled in C_a, and the only way to remove a from C_a is to fire Ea (as ${}^\bullet Ea = \{a\}$). Hence we must have $x = y\, Ba\, w\, Ea\, v$, where $\pi_{\{Ba,Ea\}}(w) = \varepsilon$. $\square$

Since all transitions of the interval representation of ENI are instantaneous, simultaneous executions of any kind are disallowed. Although defining interval firing step sequences is mathematically possible (for example the net $\mathcal{N}$ from Figure 5.1 can produce a step sequence $\{Ba, Bc\}\{Ea\}\{Bb, Ec\}\{Eb\}$), it does not make much sense as Bt and Et are interpreted as event beginning and end, i.e. they are *instantaneous*, so their simultaneous occurrence is not observable since we cannot measure time points with infinite precision. Therefore. the only operational semantics for interval representations, is the firing sequences semantics (c.f. Janicki (2008)).

The net $\mathcal{N}$ from Figure 5.1 has ten interval firing sequences that involve all elements of $\mathcal{T} = \{Ba, Ea, Bb, Eb, Bc, Ec\}$, namely $BaEaBbEbBcEc$ - which represents a total order $<_1^{\mathtt{N}}$ from Figure 1.1; $BcEcBaEaBbEb$ - which represents a total order $<_2^{\mathtt{N}}$ from Figure 1.1; $BaBcEcEaBbEb$, $BaBcEaEcBbEb$, $BcBaEcEaBbEb$, $BcBaEaEcBbEb$ - all four represent a stratified order $<_3^{\mathtt{N}}$ of Figure 1.1; and $BaBcEaBbEbEc$, $BaBcEaBbEcEb$, $BcBaEaBbEbEc$, $BcBaEaBbEcEb$ - all four represent an interval order $<_4^{\mathtt{N}}$ of Figure 1.1. It is important to stress that if observations are not allowed to be recorded as interval firing sequences, then $<_4^{\mathtt{N}}$ can be generated by neither firing sequence nor by firing step sequence. This order is an interval order, but it is not stratified, so step sequence semantics does not work.

The following proposition shows soundness and completeness of the interval representation from Definition 14 with respect to step sequence operational semantics. Note that a separate analysis of firing sequence semantics is not needed as each sequence $a_1 \dots a_n$ is uniquely represented by a step sequence $\{a_1\} \dots \{a_n\}$.
For every $A = \{t_1, ..., t_k\} \subseteq T$, let $A^{BE} \subseteq \mathcal{T}^*$ be defined as follows.
$A^{BE} = \{Bt_{i_1}...Bt_{i_k}Et_{j_1}...Et_{j_k} \mid i_1, ..., i_k \text{ and } j_1, ..., j_k \text{ are permutations of } 1, 2, ..., k\}$.
For example $\{a, b\}^{BE} = \{BaBbEaEb, BaBbEbEa, BbBaEaEb, BbBaEbEa\}$.

Proposition 2. *For every two configurations $C, C' \subseteq P \subseteq \mathcal{P}$ and every $A \subseteq T$,*

$$C\ [A\rangle\ C' \iff \forall x \in A^{BE}.\ C\ [\![x\rangle\!\rangle\ C'.$$

Proof. ($\Rightarrow$) Let $A = \{t_1, ..., t_k\}$. This means, if $i \neq j$ then $(t_i^\bullet \cup {}^\bullet t_i) \cap (t_j^\bullet \cup {}^\bullet t_j) = \emptyset$, ${}^\bullet A \subseteq C$, $(A^\bullet \cap {}^\circ A) \cap C = \emptyset$, and $C' = (C \setminus {}^\bullet A) \cup A^\bullet$. Let $y = Bt_{i_1}...Bt_{i_k}$ and $z =$

$Et_{j_1}...Et_{j_k}$, where $i_1, ..., i_k$ and $j_1, ..., j_k$ are permutations of $1, 2, ..., k$. Since ${}^\bullet t_i = {}^\bullet Bt_i$ and ${}^\circ t_i = {}^\circ Bt_i$, we have $C \ [\![y]\!\rangle \ C_B$, where $C_B = (C \backslash ({}^\bullet Bt_{i_1} \cup ... {}^\bullet Bt_{i_k})) \cup (Bt^\bullet_{i_1} \cup ... Bt^\bullet_{i_k}) = (C \backslash {}^\bullet A) \cup (Bt^\bullet_{i_1} \cup ... Bt^\bullet_{i_k})$. But $Bt^\bullet_i = \{t_i\}$, so $C_B = (C \backslash {}^\bullet A) \cup A$. However, ${}^\bullet Et_i = \{t_i\}$, so $C_B \ [\![z]\!\rangle \ C_E$, where $C_E = (C_B \backslash ({}^\bullet Et_{j_1} \cup ... \cup {}^\bullet Et_{j_k})) \cup (Et^\bullet_{j_1} \cup ... Et^\bullet_{j_k})$. Since ${}^\bullet Et_i = \{t_i\}$ and $Et^\bullet_i = t^\bullet_i$, $C_E = (C_B \setminus A) \cup A^\bullet = (((C \setminus {}^\bullet A) \cup A) \setminus A) \cup A^\bullet = (C \setminus {}^\bullet A) \cup A^\bullet = C'$.

Hence $C \ [A\rangle \ C' \Longrightarrow \forall x \in A^{BE}. \ C \ [\![x]\!\rangle \ C'$.

($\Leftarrow$) Let $A = \{t_1, ..., t_k\}$ and $C \ [\![yz]\!\rangle \ C'$. Hence there are configurations $C^0_B, C^1_B, ..., C^k_B$, $C^0_E, C^1_E, ..., C^k_E$ in N^{BE} such that $C = C^0_B$, $C^k_B = C^0_E$, $C^k_E = C'$, and $C^0_B \ [\![Bt_{i_1}]\!\rangle \ C^1_B \ [\![Bt_{i_2}]\!\rangle \ C^2_B ... C^{k-1}_B \ [\![Bt_{i_k}]\!\rangle \ C^k_B \ [\![Et_{j_1}]\!\rangle \ C^1_E \ [\![Et_{j_2}]\!\rangle \ C^2_E ... C^{k-1}_E$ $[\![Et_{j_k}]\!\rangle \ C^k_E$. We have $C^{l+1}_B = (C^l_B \setminus {}^\bullet Bt_{i_l}) \cup Bt^\bullet_{i_l}$, and $C^{l+1}_E = (C^l_E \setminus {}^\bullet Et_{j_l}) \cup Et^\bullet_{j_l}$, for $l = 0, ..., k-1$. Because ${}^\bullet Bt_i = {}^\bullet t_i$, and $Bt^\bullet_i = \{t_i\}$, then $C^0_E = C^k_B = (C^0_B \setminus {}^\bullet A) \cup A = (C \setminus {}^\bullet A) \cup A$. However, ${}^\bullet Et_i = \{t_i\}$ and $Et^\bullet_i = t^\bullet_i$, so $C^k_E = (C^0_E \setminus A) \cup A^\bullet$.

Thus, $C' = C^k_E = (C^0_E \setminus A) \cup A^\bullet = (((C \setminus {}^\bullet A) \cup A) \setminus A) \cup A^\bullet = (C \setminus {}^\bullet A) \cup A^\bullet$. But this means $C \ [A\rangle \ C'$. □

Proposition 2 shows that firing a step A in the net $\mathtt{N}$ is properly simulated by firing an appropriate sequence from $\mathcal{T}^*$ in the net $\mathcal{N}$. Moreover, while the net $\mathcal{N}$ defines behaviours that cannot be defined by $\mathtt{N}$ (as $<^{\mathtt{N}}_4$ for the net from Figure 1.1), it does *not* generate any new behaviour that can be described by step sequences of $\mathtt{N}$.

It is worth pointing out that there is a connection between Theorem 1 (Fishburn (1970)), Theorem 2 (Abraham et al. (1990)) and the results of this chapter. In all three cases, interval orders are represented by entities built from beginnings and ends. We then formally showed that such a representation is sound and complete in our case.

Chapter 6

Process Semantics

This chapter comprises the main results of Janicki and Koutny (1995); Kleijn and Koutny (2004); Juhás et al. (2007) as our approach presented in Chapter 8 is partially based on them.

While firing sequences represent the absolute sequential semantic of EN and ENI, which allows only one transition firing at any given marking, the firing step sequences offers the possibility of more than one transition occurring together at some configuration. However, step sequences still has a sequential flavour and it is closer to simultaneity than it is to concurrency. Despite the importance of operational semantics (firing sequences and firing step sequences) as a starting point, they are still very limited for providing a useful semantics for Petri nets.

One of the essential parts of concurrent processing is that many different system runs/executions are equivalent, but this aspect is virtually impossible to capture when only operational semantics is considered. Abstractions of these equivalent executions are often called *concurrent histories* or *non-sequential execution histories*, and dependent on the assumptions made about systems and system runs, are usually modeled

by partial orders Goltz and Reisig (1983); Nielsen et al. (1990), stratified order structures or interval order structures Janicki and Koutny (1991, 1995); Kleijn and Koutny (2004). However, such modeling is complex and not as straightforward as constructing *occurrence nets* which can be defined to be equivalent to the aforementioned structures. Put simply, the goal of using occurrences net is to record the changes of configurations due to transitions being executed along some legal behaviour (defined as firing sequence and/or firing step sequences) of EN and ENI systems. Recording changes implies keeping track of places whose tokens were consumed (input places) and places which are being populated with tokens (output places).

In this chapter, we will provide a brief overview of the semantical framework, defined in Kleijn and Koutny (2004) and then extended in Juhás et al. (2007), which aims at a systematic presentation of the process and causality semantics for various classes of Petri nets. Although, the framework is not followed exactly, its main parts are embedded in our research. In fact, the framework was a sort of a guideline for the development of the process semantics for interval ENI (defined in Chapter 5). After recalling the framework, a formal definition of occurrence nets will be given, and some theory about it will be provided before two algorithms for constructing occurrence nets generated by firing sequences, and firing step sequences of EN are discussed. The presence of inhibitor arcs requires a modified version of occurrence net with activator arcs; this will also be discussed in this chapter.

6.1 Semantical Framework for Process Semantics

For a given class of Petri nets, one has to define the following semantical domains: (see Kleijn and Koutny (2004))

- $\mathcal{EX}$ are executions, such as step sequences, employed by the operational (behavioural) semantics of nets in $\mathcal{PN}$;
- $\mathcal{LAN}$ are labelled acyclic nets, such as occurrence nets, providing the structural description of abstract processes of nets in $\mathcal{PN}$, with each labelled net in LAN representing a single non-sequential history;
- $\mathcal{LEX}$ are labelled executions, such as labelled step sequences, employed by the operational semantics of nets in $\mathcal{LAN}$;
- $\mathcal{LCS}$ are labelled causal structures, such as labelled partial orders, defining an abstract causality semantics of nets in $\mathcal{PN}$.

The relation between those domains is defined in terms of functions that, when instantiated, define and relate the three views on semantics for the Petri net. Those functions are illustrated in Figure 6.1 (taken from Juhás et al. (2007)). The functions are defined as follows. The function $\omega : \mathcal{PN} \rightarrow \mathbb{P}(\mathcal{EX})$ provides the operational semantics of the net through a non-empty set of executions. The function $\alpha : \mathcal{PN} \rightarrow (\mathbb{P})(\mathcal{LAN})$ associates with the net a non-empty set of processes. A process is given an operational semantics via $\lambda : \mathcal{LAN} \rightarrow \mathbb{P}(\mathcal{LEX})$ which associates with it a non-empty set of labelled executions. The labelled executions can be interpreted as an ordinary execution through the function $\phi : \mathcal{LEX} \rightarrow \mathcal{EX}$. To conclude the square-like part of Figure 6.1 (i.e. the part concerned with process semantics), the partial function $\pi_N : \mathcal{EX} \rightarrow \mathbb{P}(\mathcal{LAN})$ defines a none empty set of labelled acyclic nets for each execution of the net. As for the triangle-like part which is to define the abstract causality semantics of processes we have the following function. The function $\kappa : \mathcal{LAN} \rightarrow \mathcal{LCS}$ associating a labelled causal structure with each process in $\mathcal{LAN}$. The

function $\epsilon : \mathcal{LCS} \rightarrow \mathbb{P}(\mathcal{LEX})$ is used to relate the abstract causality semantics to the operational semantics of processes. Finally, the function $\iota : \mathbb{P}(\mathcal{LEX}) \rightarrow \mathcal{LCS}$ which allows to go back and forth between labelled causal structure and labelled executions.

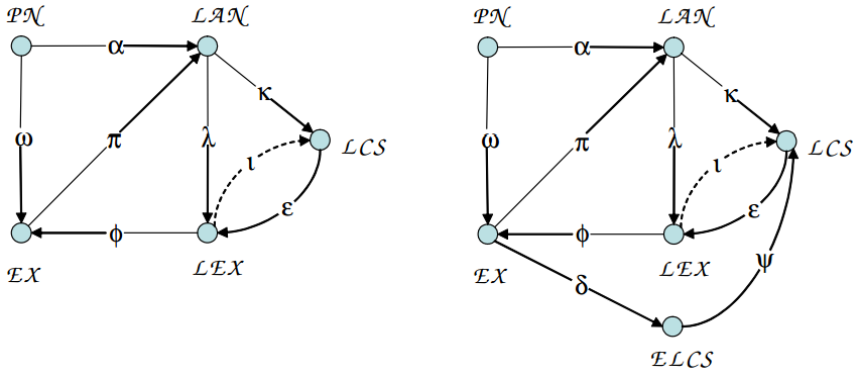


Figure 6.1: Semantical framework for process semantics from Kleijn and Koutny (2004) (right) and complete semantical framework for process semantics from Juhás et al. (2007)

The ultimate goal of the framework is to show that, for a given Petri net, different types of semantics agree with each other. This is done through introducing some natural conditions (called properties) which in turn allow one to focus solely on defining the different semantical domains and functions, then establishing the properties in question will lead the desired result (i.e. showing the agreement between different semantical domains). Those semantical characteristics are called aims. Here we will just list those properties and aims; readers are referred to Kleijn and Koutny (2004) and Juhás et al. (2007) for full formal arguments and formulation.

- Property 1. The functions $\omega, \alpha, \lambda, \phi$ and $\pi_N|_{\omega(N)}$ are total. Moreover, ω, α, λ and $\pi_N|_{\omega(N)}$ never return the empty set.

- Property 2 (Consistency). For all $\xi \in \mathcal{EX}$ and $LN \in \mathcal{LAN}$;

$$\xi \in \omega(N) \wedge LN \in \pi_N(\xi) \text{ iff } LN \in \alpha(N) \wedge \xi \in \phi(\lambda(LN)).$$

  Provided that this property has been established for a given net in $\mathcal{PN}$, the two aims formulated above follow.

  Aim 1. $\alpha = \pi_N \circ \omega$.

  Aim 2. $\omega = \phi \circ \lambda \circ \alpha$.

- Property 3. The functions κ, ϵ, and $\iota|\ \lambda(\mathcal{LAN})$ are total. Moreover, ϵ never return the empty set.

- Property 4 (Representation). $\iota \circ \epsilon = id_{\mathcal{LCS}}$.

- Property 5 (Fitting). $\lambda = \epsilon \circ \kappa$ and then we have

  Aim 3. $\kappa = \iota \circ \lambda$.

To use the above setup in practice, one needs to establish Properties 1 and 3, and check that the consistency, representation and fitting properties hold true (Properties 2, 4, and 5) then the semantical aims will follow.

The framework defined above was extended by the completeness-requirement in Juhás et al. (2007). The requirement was formulated in terms of enabled causality structures. In its essence, it states that causality semantics deduced from process nets should be complete w.r.t. step semantics in the sense that each causality structure which is consistent with the step semantics corresponds to some process net. This is done by introducing two additional functions (see Figure 6.1) namely δ and Ψ where

the former represents the definition of enabled labeled causal structures $\mathcal{ELCS}$ while the latter relates enabled labeled causal structures ($\mathcal{ELCS}$) and runs ($\kappa(\mathcal{LAN}) \subseteq \mathcal{LCS}$). Then through $\psi \circ \delta$ the aim of completeness can be added to the framework.

> Aim of completeness. The mapping δ assigns a set of step sequences $\mathcal{EX}$ onto the set of causal structures $\mathcal{ELCS}$ enabled w.r.t. $\mathcal{EX}$. The mapping ψ assigns a run $\mathcal{LCS}$ with less causality to each enabled causal structure in $\mathcal{ELCS}$ for which such a run exists.

Formally, a labeled causal structure is said to have less causality than a second one, if each labeled execution in $\mathcal{EX}$ generated by the second one is also generated by the first one (where the labeled executions generated by a labeled causal structure are given by ϵ) Juhás et al. (2007).

Again, in our research we did not explicitly define the properties and aims for any of the Petri net models we use but rather the connection between various semantical domains are more implicit. In the rest of this chapter, we recall the result of Janicki and Koutny (1995); Kleijn and Koutny (2008) which deals with defining process semantics for elementary nets and elementary nets with inhibitor arcs.

6.2 Processes of Elementary Nets

In the case of EN, an occurrence net is generated by either a firing sequence or a step sequence is just a plain net unfolding caused by the execution of it (c.f. Goltz and Reisig (1983); Rozenberg and Engelfriet (1998); Kleijn and Koutny (2008)). A process of an EN is an occurrence net the describes its structure and its legal behaviours through configurations and labeling.

Occurrence nets (sometimes called processes nets, or causal nets) is a special class of Petri nets that is acyclic and with "unbranching" places.

Definition 17. *A net $N = (B, E, R)$ is a condition/event net if $B \cap E = \emptyset$ and $R \subseteq (B \times E) \cup (E \times B)$. An* occurrence net (ON) *is a condition/event net N such that:*

1. *ON is acyclic, and*

2. *$| {}^\bullet b | \leq 1$ and $| b^\bullet | \leq 1$ for every $b \in B$.* □

Places of an occurrence nets are called *conditions* ('Bedingungen' in German thus the letter B) and transitions are called *events* ('Ereignisse' in German thus the letter E). Additionally, in an occurrence net, the *initial configuration* consists of all condition with no incoming arcs (i.e. $C^{ON}_{init} = \{b \in B \mid {}^\bullet b = \emptyset\}$), and the *finial configuration* consists of all condition with no outgoing arcs (i.e. $C^{ON}_{fin} = \{b \in B \mid b^\bullet = \emptyset\}$).

A process of an EN is constructed by a firing sequence or a step sequence of an EN using Algorithm 1, or Algorithm 2 respectively; (see Kleijn and Koutny (2008)).

We, first, define the processes generated by the firing sequence $x = t_1 \dots t_n$ as $P_x = ON_n$, where ON_n is the last *occurrence net* in the sequence $ON_0, \dots, ON_n$. Each net $ON_k = (B_k, E_k, R_k), 0 \leq k \leq n$, is a net that model an unfolding of the EN by the sequence $t_1 \dots t_k$. The components of ON_k correspond to places P, transitions T, and flow relation F of the underlying EN. The elements of $B_k \cup E_k$ are of the form r^i, where $r \in P \cup T$ and $i \geq 1$. We will denote $l(r^i) = r$. Moreover, for every $r \in P \cup T$ and $k \leq n$, Δr is the number of nodes of N_{k-1} labelled by r (i.e. the number of $\alpha \in B_k \cup E_k$ such that $l(\alpha) = r$.)

Algorithm 1 (Constructing P_x, for $x = t_1 \dots t_n$).

- *Step 0.* $ON_0 = (\{(p^1) \mid p \in C_{init}\}, \emptyset, \emptyset, \emptyset)$
- *Step k. Given* ON_{k-1}*, we define* ON_k *in the following way:*
  - $B_k = B_{k-1} \cup \{p^{1+\Delta p} \mid p \in t_k^\bullet\}$
  - $E_k = E_{k-1} \cup \{t_k^{1+\Delta t_k}\}$
  - $R_k = R_{k-1} \cup \{(p^{\Delta p}, t_k^{1+\Delta t_k}) \mid p \in {}^\bullet t_k\} \cup \{(t_k^{1+\Delta t_k}, p^{1+\Delta p}) \mid p \in t_k^\bullet\}$ □

An extension of Algorithm 1 to the case of firing step sequence $y = U_1 \dots U_n$ is rather straightforward.

Algorithm 2 (Constructing P_y for $y = U_1 \dots U_n$).

- *Step 0.* $ON_0 = (\{(p^1) \mid p \in C_{init}\}, \emptyset, \emptyset, \emptyset)$
- *Step k. Given* N_{k-1}*, we define* ON_k *in the following way:*
  - $B_k = B_{k-1} \cup \{p^{1+\Delta p} \mid p \in U_k^\bullet\}$
  - $E_k = E_{k-1} \cup \{t^{1+\Delta t} \mid t \in U_k\}$
  - $R_k = R_{k-1} \cup \{(p^{\Delta p}, t^{1+\Delta t}) \mid t \in U_k \wedge p \in {}^\bullet t\} \cup$
    $\{(t^{1+\Delta t}, p^{1+\Delta p}) \mid t \in U_k \wedge p \in t^\bullet\}$ □

Let's consider the EN in Figure 4.1 together with the firing sequence $x = \{t_1 t_4 t_5 t_6\}$. A process can be constructed following Algorithm 1 as details in Figure 6.2. It starts by recording the initial configuration of EN as in Figure 6.2(a). Then, t_1 fires for the first time (i.e. t_1^1) recording its output $\{s_3^1, s_4^1\}$ as in Figure 6.2(b). In (c), and (d) from the same figure, transition t_4 occurs then followed by the firing of t_5, both for the first time. Lastly, t_6 fires causing conditions $\{s_1^2, s_2^2\}$ to hold; see Figure 6.2(e). It

is also worth mentioning here that a process does not necessarily describe a complete run of the system as such run can be infinite. It rather describes an initial finite part of it. Moreover, processes have special characteristic: 1) if an event occurs, a possible conflict is resolved, and 2) different occurrences of the same event and/or condition are recorded differently. Furthermore, processes record conditions that hold and not the condition that do not hold. This implies that processes can only be defined for contact-free nets. Note that EN from Figure 4.1 is contact-free (see Definition 10) and the resulted occurrence net Figure 6.2(e) is conflict-free (see Definition 9)

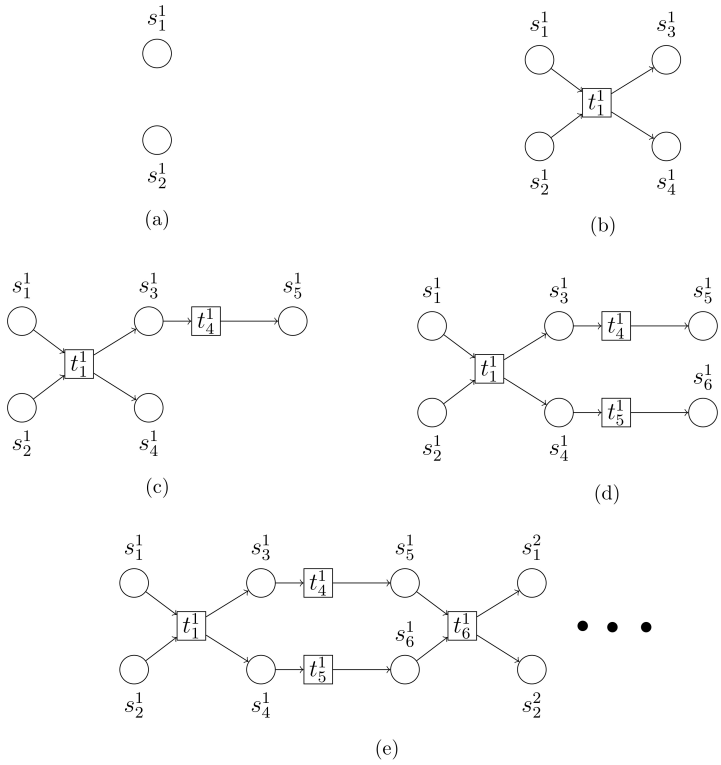


Figure 6.2: A step by step construction of a process for EN from Figure 4.1 generated by $x = \{t_1 t_4 t_5 t_6\}$.

As we have seen, constructing a process for an elementary net is rather straightforward as it is just unfolding of the EN based on either a firing sequence or a step

sequence. However, for elementary net with inhibitor arcs, activator arcs are needed to deal with inhibitor arcs and this will be discussed next.

6.3 Processes of Elementary Nets with Inhibitor Arcs

It was shown in Janicki and Koutny (1995); and Kleijn and Koutny (2008) that the plain unfolding of elementary nets to construct processes dose not work with ENI systems. This is because the absence of a token, unlike the presence of a token, cannot be tested. Hence we have to replace inhibitor arcs by appropriate *activator arcs*. Activator arcs allow a transition to check for a presence of a token (see Section 4.3). The idea is that an inhibitor arc which tests whether a place is empty, can be simulated by an activator arc which tests whether its *complement place* is not empty. To do such simulation, each inhibitor place must have its complements (See Definition 11), if it does not we can always add it, as it *does not change the net behaviour* (c.f. Goltz and Reisig (1983); Janicki and Koutny (1995); Kleijn and Koutny (2004); Nielsen et al. (1990)). We will call the nets with this property *complement closed*. The addition of complement places to the `ENI` in Figure 4.6 is presented in Figure 6.3. Now, we fix the ENI to be the complement closed presented in Figure 6.3 for the rest of this section.

Definition 18. *An **activator occurrence net (AO)** is a relational tuple* $AN = (B, E, R, Act)$ *such that the first three components are the underlying occurrence net and*

- $Act \subseteq B \times E$ *is the set of activator arcs.* □

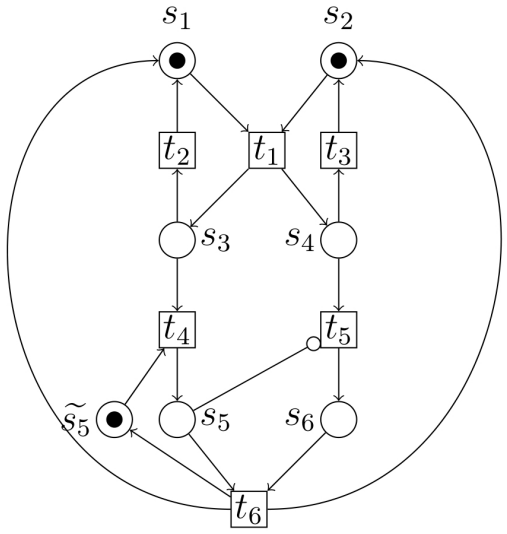


Figure 6.3: Complement closed ENI.

Similarly to the construction of processes for elementary nets, we will now demonstrate the construction of processes for the *complement closed* elementary nets with inhibitor arcs: first generated by a firing sequence $x = t_1 \dots t_n$, $t_i \in T$ (c.f. Janicki and Koutny (1995); Kleijn and Koutny (2008)) and then generated by a step sequence $y = U_1 \dots U_n$, $U_i \subseteq T$ (c.f. Janicki and Koutny (1995); Kleijn and Koutny (2008)).

We define the processes generated by $x = t_1 \dots t_n$ as $P_x = AN_n$, where AN_n is the last *activator occurrence net* in the sequence $AN_0, \dots, AN_n$. Each net $AN_k = (B_k, E_k, R_k, A_k), 0 \leq k \leq n$, is a net with *activator arcs* that model an unfolding of the net ENI by the sequence $t_1 \dots t_k$. The first three components of AN_k correspond to places P, transitions T, and flow relation F of the underlying ENI, while $A_k \subseteq B_k \times E_k$ is the set of activator arcs derived from inhibitors arcs I.

The elements of $B_k \cup E_k$ are of the form r^i, where $r \in P \cup T$ and $i \geq 1$. We will denote $l(r^i) = r$. Moreover, for every $r \in P \cup T$ and $k \leq n$, Δr is the number of nodes of N_{k-1} labelled by r (i.e. the number of $\alpha \in B_k \cup E_k$ such that $l(\alpha) = r$.)

Algorithm 3 (Constructing P_x, for $x = t_1 \dots t_n$).

- *Step 0.* $AN_0 = (\{(p^1) \mid p \in C_{init}\}, \emptyset, \emptyset, \emptyset)$
- *Step k. Given AN_{k-1}, we define AN_k in the following way:*
  - $B_k = B_{k-1} \cup \{p^{1+\Delta p} \mid p \in t_k^\bullet\}$
  - $E_k = E_{k-1} \cup \{t_k^{1+\Delta t_k}\}$
  - $R_k = R_{k-1} \cup \{(p^{\Delta p}, t_k^{1+\Delta t_k}) \mid p \in {}^\bullet t_k\} \cup \{(t_k^{1+\Delta t_k}, p^{1+\Delta p}) \mid p \in t_k^\bullet\}$
  - $A_k = A_{k-1} \cup \{(\widetilde{p}^{\Delta \widetilde{p}}, t_k^{1+\Delta t_k}) \mid p \in {}^\circ t_k\}$ □

An extension of Algorithm 3 to the case of firing step sequence $y = U_1 \dots U_n$ (first proposed in Janicki and Koutny (1995)) is rather straightforward.

Algorithm 4 (Constructing P_y for $y = U_1 \dots U_n$).

- *Step 0.* $AN_0 = (\{(p^1) \mid p \in C_{init}\}, \emptyset, \emptyset, \emptyset)$
- *Step k. Given AN_{k-1}, we define AN_k in the following way:*
  - $B_k = B_{k-1} \cup \{p^{1+\Delta p} \mid p \in U_k^\bullet\}$
  - $E_k = E_{k-1} \cup \{t^{1+\Delta t} \mid t \in U_k\}$
  - $R_k = R_{k-1} \cup \{(p^{\Delta p}, t^{1+\Delta t}) \mid t \in U_k \wedge p \in {}^\bullet t\} \cup$
    $\{(t^{1+\Delta t}, p^{1+\Delta p}) \mid t \in U_k \wedge p \in t^\bullet\}$
  - $A_k = A_{k-1} \cup \{(\widetilde{p}^{\Delta \widetilde{p}}, t^{1+\Delta t}) \mid t \in U_k \wedge p \in {}^\circ t\}$ □

Now let's consider the `ENI` in Figure 6.3 together with the firing step sequence $y = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4, t_5\}\}$. Using Algorithm 4, the construction is laid out step by step in Figure 6.4. It is very similar to the EN process in Figure 6.2 with two

noticeable differences. First, in Figure 6.4(c), one can see that the concurrent firing of transition t_1 and t_2 (i.e. event t_1^1 and t_2^1). Second, in Figure 6.4(e), there is the activator arc that connects $\widetilde{s_5^1}$ with t_5^1 simulating the inhibitor arc in the ENI between s_5 and t_5.

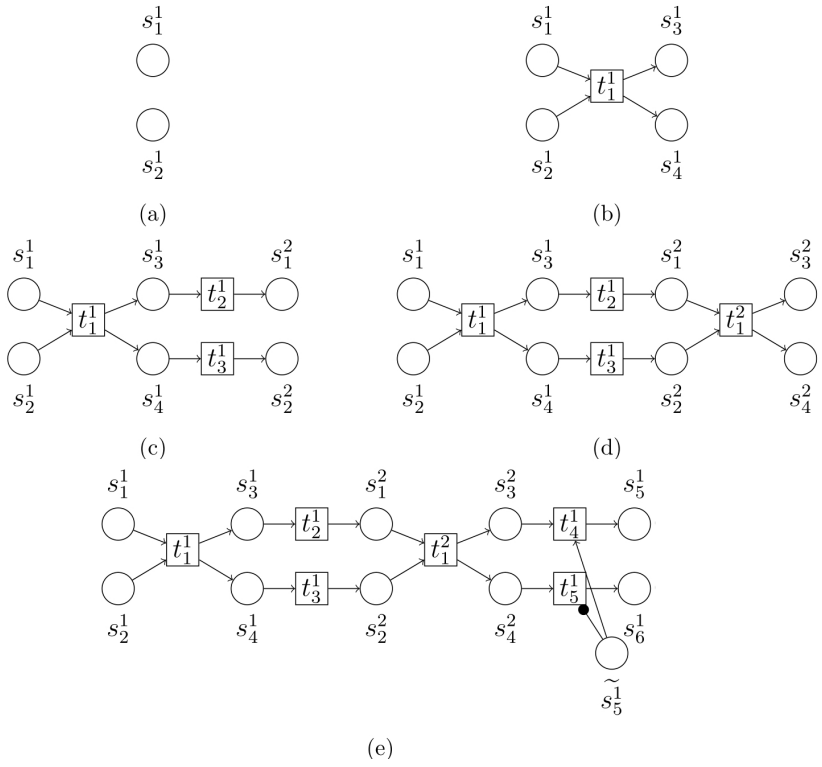


Figure 6.4: A step by step construction of a process for ENI from Figure 6.3 generated by $y = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4, t_5\}\}$.

The generated process can be viewed as *partial* unfoldings of the original net such that each event in the process represents a transition occurrence of the original net while each condition corresponds to the presence of a token in a place in the original net. If the original net does not have any inhibitor arcs, the generated process are the same as these for standard elementary nets (c.f. Janicki and Koutny (1995); Nielsen et al. (1990)).

6.4 Processes and Concurrent Histories

We will now show how processes, i.e. occurrence nets, can be interpreted as *concurrent histories* starting with the simple case of processes of elementary nets. Then we will do the same for processes of elementary nets with inhibitor arcs (i.e. activator occurrence nets). For an elementary net , given a process generated by a firing sequence x, one can simply abstract from the conditions of the process. Such an abstraction results in associating a *directed acyclic graph* ($\lessdot_x^{init}$) where the events of the process E are its nodes. Since the relation between the transitions in the underlying EN (and subsequently between the events in ON) can only be causality, the defined graph ($\lessdot_x^{init}$) represents the direct causality between event and its transitive closure (partial order $\lessdot_x^{proc}$) gives indirect causal relations (in addition to the direct ones) between events.

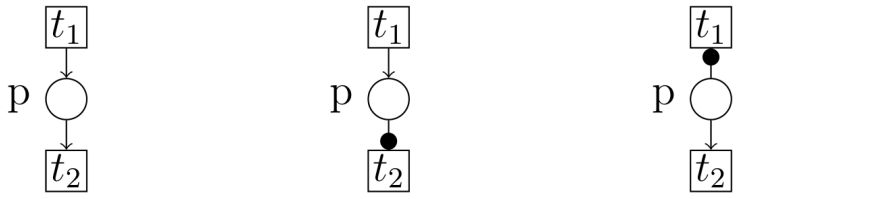
For the process in Figure 6.2(e) which is generated by $x = \{t_1 t_4 t_5 t_6\}$, a corresponding $\lessdot_x^{init} = (E, R \circ R \mid_{E \times E})$ and $(\lessdot_x^{init})^+$ (i.e. $\lessdot_x^{proc}$) are presented in Figure 6.5.

(a) $\lessdot_x^{init}$ (b) $\lessdot_x^{proc}$

Figure 6.5: Direct acyclic graph associated with the process in Figure 6.2(e) and its transitive closure.

In case of elementary nets with inhibitor arcs, the relation between transitions in the original net (and consequently between events in the process) is more complex.

This is true in the case of processes that are generated by firing step sequence. However, in case of processes generated by a firing sequence x, similarly to processes of EN, it can be interpreted as some partial order $\preccurlyeq_x^{proc}$. Such a partial order can be obtained by first transforming the process into a *directed acyclic graph* $\preccurlyeq_x^{init}$ using described in Figure 6.6; see Kleijn and Koutny (2008).



(a) $t_1 \preccurlyeq_x^{init} t_2$ (b) $t_1 \preccurlyeq_x^{init} t_2$ (c) $t_1 \preccurlyeq_x^{init} t_2$

Figure 6.6: Rules of deriving a partial order from an ENI process generated by a firing sequence x.

Formally, the construction of both the directed acyclic graph $\preccurlyeq_x^{init}$ and the partial order $\preccurlyeq_x^{proc}$ is defined as follows.

Definition 19. *Let $AN_n = (B_n, E_n, R_n, A_n)$ be the process generated by firing sequence. We define a* ***directed acyclic graph*** *$\preccurlyeq_x^{init}$ and a* ***partial order*** *$\preccurlyeq_x^{proc}$, both on E_n as follows:*

1. *For all $t_1, t_2 \in E_n$,*

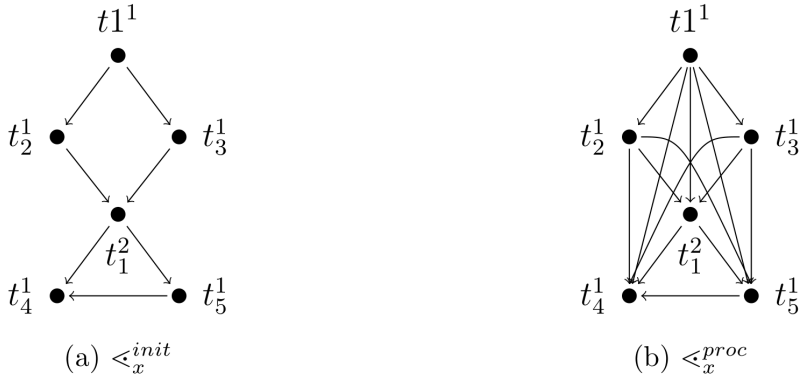
$$t_1 \preccurlyeq_x^{init} t_2 \iff t_1(R_n \circ R_n)t_2 \vee t_1(R_n \circ A_n)t_2 \vee t_1(A_n^{-1} \circ R_n)t_2$$

2. *For all $t_1, t_2 \in E_n$, $t_1 \preccurlyeq_x^{proc} t_2 \iff t_1(\preccurlyeq_x^{init})^+ t_2$* □

Note that although the processes of `ENI` presented in Figure 6.4(e) is generated by a firing step sequence $y = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4, t_5\}\}$, the same process would have

been constructed if a firing sequence $x = \{t_1t_2t_3t_1t_5t_4\}$ (which is a valid sequence for the ENI in Figure 6.3). The steps of such a construction would *not* be the same, however. Now, let's assume that Figure 6.4(e) was generated by firing sequence x, the the directed acyclic graph $\lessdot_x^{init}$ and the partial order $\lessdot_x^{proc}$ associated with it are presented in Figure 6.7.

The partial order $\lessdot_x^{proc}$ defines a concurrent behaviour comprising all total extensions of itself (this includes the total order defined by the sequence generating the process to which it is associated, in this case $x = \{t_1t_2t_3t_1t_5t_4\}$). Another sequence (defined as a total order) that is also an extension of the partial order $\lessdot_x^{proc}$ is $x = \{t_1t_3t_2t_1t_5t_4\}$.

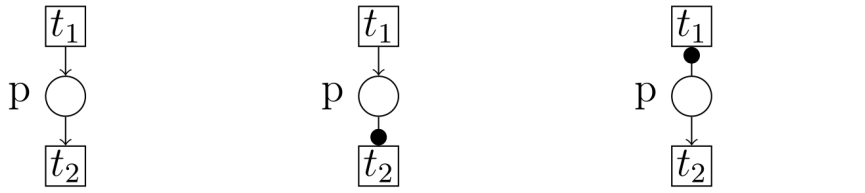


(a) $\lessdot_x^{init}$ (b) $\lessdot_x^{proc}$

Figure 6.7: Direct acyclic graph associated with the activator occurrence net in Figure 6.4(e) and its transitive closure assuming it was generated by $x = \{t_1t_2t_3t_1t_5t_4\}$.

On the other hand, when processes are generated by firing step sequences y, a partial order is no longer enough to capture ENI behaviours. Instead, we need to define a stratified order structure that allows us to capture not only causality between events but also the "no later than" relation. The rules for defining such a stratified order are informally presented in Figure 6.8. The relation $\prec_y^{init}$ captures causality while the relation $\sqsubset_y^{init}$ captures what is known as weak causality or no later than

relation. Intuitively, $\prec_y^{init}$ is when the first event has to produce a token which is consumed or tested (for existence) by the second event and $\sqsubset_y^{init}$ is when the first event must occur before the second one as the latter consumes a token for which the former test.



(a) $t_1 \prec_y^{init} t_2$ (b) $t_1 \prec_y^{init} t_2$ (c) $t_1 \sqsubset_y^{init} t_2$

Figure 6.8: Rules of deriving a stratified order from an activator occurrence net (process) generated by a step sequence y.

The rules in Figure 6.8 can be formalized as follows.

Definition 20. *$S_y = (E_n, \prec_y^{init}, \sqsubset_y^{init})$, where for all $t_1, t_2 \in E_n$,*

- $t_1 \prec_y^{init} 1_2 \iff t_1(R \circ R)t_2 \vee t_1(R \circ Act)t_2$
- $t_1 \sqsubset_y^{init} t_2 \iff t_1(Act^{-1} \circ R)t_2$ □

Given the process of `ENI` presented in Figure 6.4(e), a stratified order structure is constructed following Definition 20 and presented in Figure 6.9 in which $\prec_y^{init}$ is represented as solid lines and $\sqsubset_y^{init}$ is represented as dashed lines. Dashed lines are omitted if solid ones are present.

The stratified order structure $S_y = (E_n, \prec_y^{init}, \sqsubset_y^{init})$ defines a concurrent behaviour comprising all stratified order extensions of S_y. For the case of Figure 6.9, $S_{\{\{t_1\},\{t_2,t_3\},\{t_1\},\{t_4,t_5\}\}} = S_{\{\{t_1\},\{t_2,t_3\},\{t_1\},\{t_5\},\{t_4\}\}}$.

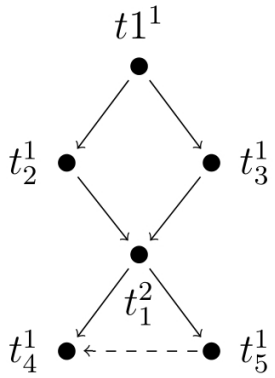


Figure 6.9: Stratified order structure associated with the activator occurrence net in Figure 6.4(e).

In this chapter we have given a practical overview of the well-developed process semantics for elementary nets and elementary nets with inhibitor arcs and their association with causal structures. However, none of those semantics and causal structure suffices when system observations are assumed to be interval orders. To do so, we will define process semantics for interval elementary nets with inhibitor arcs (defined in Chapter 5) in a similar fashion that had been provided in this section. However, before we do that, we will discuss another semantics for elementary nets and elementary nets with inhibitor arcs which aim to provide a sequential semantics that is enriched with important information about the complex relation between transitions (just like the important information provided by process semantics). The Trace semantics will be the focus of the next chapter.

Chapter 7

Trace Semantics

In this chapter, we overview briefly Mazurkiewicz traces which are suitable for modeling elementary nets, as well as comtrace which is used when Mazurkiewicz traces are not enough. This is the case with elementary nets with inhibitor arcs. We then discuss interval traces in more details as our model will be compared to them in a later chapter.

7.1 Mazurkiewicz Traces

Motivated by Petri nets and formal languages with automata, Antoni Mazurkiewicz came up with trace theory in Mazurkiewicz (1977). Its aim was to provide a mean to describe the behaviour of concurrent systems (Petri nets) in a more meaningful way than just interleaving which was the most common way to describe concurrency. To achieve such a goal, there was a need to reconcile the sequential nature of observation of the behaviour of the system on one hand and the non-sequential nature of causality between actions of the system on the other hand. Trace theory can be seen in two

different ways either rooted in formal language theory with the notion of partial commutativity at its center or rooted in the theory of labeled acyclic directed graphs Aalbersberg and Rozenberg (1988).

The general idea of traces can be explained as follows. The use of sequential observation is very natural to describe the behaviour of a concurrent system. This results in a record of behaviours of the system given as a sequence of actions as observed by a sequential observer. The set of all records gives the description of the behaviour of an entire system in a sequential fashion. The sequential nature of such records means that there is some information that is missing about the system: two action a and b may appear adjacent within a sequential record when they are actually performed concurrently within the system. Therefore, in order to extract faithful information about the system from sequential records, additional information about the system itself must be provided. The solution of trace theory is elegant as such information is given as a binary relation (called independence) over the set of all actions in the system. A pair (a, b) belongs to the independence relation if there is no direct causal relation between them within the system. Now, given that (a, b) belongs to independence and given a sequential record x of the form x_1abx_2, one may commute the occurrences of a and b to obtain another valid equivalent sequential run (i.e. x_1bax_2). Thus, when trace theory is used, one works with the equivalence classes of observation rather than with single observation only. The above concepts can be formalized as follows.

Definition 21 (Diekert et al. (1995); Mazurkiewicz (1977, 1995)).

1. *Let Σ be a finite set and let the relation $ind \subseteq \Sigma \times \Sigma$ be an irreflexive and symmetric relation (called* independency*). The pair (Σ, ind) is called a* trace

alphabet.

2. *Let* $\approx \in \Sigma^* \times \Sigma^*$ *be a relation defined as follows:*

   $x \approx y \iff$

   $\exists x_1, x_2 \in \Sigma^*. \exists (a,b) \in ind.\ x = x_1 abx_2 \wedge y = x_1 bax_2$

3. *Let* $\equiv_{ind}$ *the reflexive and symmetric closure of* $\approx$*, i.e.* $\equiv_{ind} = \approx^*$*. Clearly is an* equivalence *relation.*

4. *For every* $x \in \Sigma$*, the equivalence class* $[x]_{\equiv_{ins}}$ *is called a* **Mazurkiewicz trace***, or just a* **trace***.* □

We will often write $[x]$ or $[x]_{ind}$ instead of $[x]_{\equiv_{Ind}}$. One may show that $[x][y] = [x] \circ [y] = [xy]$, where $\circ$ is a concatenation of sets of sequences, a symbol that is usually omitted Diekert et al. (1995); Mazurkiewicz (1995).

Formally, an algebra of Mazurkiewicz traces is a quotient equational monoid over sequences (or words) Diekert et al. (1995); Mazurkiewicz (1977).

Example 1. *Let* $\Sigma = \{a, b, c\}$*,* $ind = \{(b,c),(c,b)\}$*. Given three sequences* $s =$ *abcbca,* $s_1 =$ *abc and* $s_2 =$ *bca, we can generate the traces* $[s] = \{$*abcbca, abccba, acbbca, acbcba, abbcca, accbba*$\}$*,* $[s_1] = \{abc, acb\}$ *and* $[s_2] = \{bca, cba\}$*. Note that* $[s] = [s_1][s_2]$ *since* $[abcbca] = [abc][bca] = [abc\ bca]$*.* □

Each sequence of events represents a total order of *enumerated events* in a natural way. For precise definitions see for example Janicki and Koutny (1995). Here we will be using the following notation.

Notation 2.

1. *For each set of events* Σ*, let* $\widehat{\Sigma} = \{a^{(i)} \mid a \in \Sigma, i \geq 1\}$*.*

2. *For each sequence* $s \in \Sigma^*$, *let* $\hat{s} \in \widehat{\Sigma}^*$ *denote its enumerated representation. For example if* $s = abbaa$ *then* $\hat{s} = a^{(1)}b^{(1)}b^{(2)}a^{(2)}a^{(3)}$.

3. *For each sequence* $s \in \Sigma^*$, $\widehat{\Sigma}_s$ *denotes the set of all* enumerated events *of* s. *For example*

   $\widehat{\Sigma}_{abbaa} = \{a^{(1)}, a^{(2)}, a^{(3)}, b^{(1)}, b^{(2)}\}$.

4. *For each trace* $[s]$, *we define* $\widehat{\Sigma}_{[s]} = \widehat{\Sigma}_s$.

5. *For ever* $s \in \Sigma^*$, $\lhd_s$ *is a* total order *defined by the enumerated sequence* $\widehat{s}$. *For example*

   $\lhd_{abbaa} = a^{(1)} \to b^{(1)} \to b^{(2)} \to a^{(2)} \to a^{(3)}$.

Each trace can be interpreted as a finite partial order.

Definition 22 (Mazurkiewicz (1995)). *For every trace* $[x]$, *the partial order*

$$\lessdot_{[x]}^{trace} = \bigcap_{s \in [x]} \lhd_s$$

is called the partial order generated by $[x]$. □

Note, for each trace $[x]$ its set of all enumerated events can be defined as $\widehat{\Sigma}_{[s]} = \widehat{\Sigma}_s$.

Example 2. *For the trace* $[s] = [abcbca]$ *from Example 1, we have* $\widehat{\Sigma}_{[s]} = \{a^{(1)}, b^{(1)}, c^{(1)}, b^{(2)}, c^{(2)}, a^{(2)}\}$. *The partial order* $\lessdot_{[s]}^{trace}$ *generated by* $[s]$ *is depicted as Hasse diagram in Figure 7.1.* □

The partial order $\lessdot_{[s]}^{trace}$ generated by $[s]$ defines a concurrent behaviour comprising all total extensions of $\lessdot_{[s]}^{trace}$ (i.e. all the words s's in that trace). Thus, all information on the dependencies between the occurrences in a trace is represented uniquely in the partial order generated by the trace.

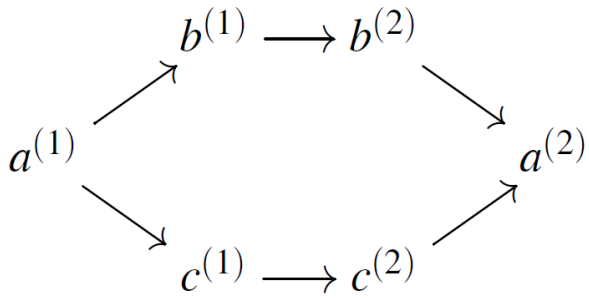


Figure 7.1: The partial order $\lessdot_{[s]}^{trace}$ generated by the trace $[s]$ where $s = abcbca$ and $ind = \{(b, c), (c, b)\}$.

In relation to elementary net (Section 4.1) the theory of traces can be applied to extract partial orders from firing sequences representing the necessary causal ordering of transition within these sequences.

Definition 23. *The **trace alphabet of EN** is the pair (T, Ind_{EN}) where Ind_{EN} is defined as follow.*

$$Ind_{EN} = \{(t_1, t_2) \mid t_1, t_2 \in T \wedge {}^\bullet t_1^\bullet \cap {}^\bullet t_2^\bullet = \emptyset\}. \qquad \square$$

In other words, Ind_{EN} is the structural independence relation of EN comprising all pairs of distinct transitions with disjoint neighbourhoods. For the `EN` in Figure 4.1, $Ind_{\texttt{EN}} = \{(t_2, t_3), (t_3, t_2), (t_2, t_5), (t_5, t_2), (t_3, t_4), (t_4, t_3), (t_4, t_5), (t_5, t_4)\}$. For firing sequences of EN, an important observation is that adjacent occurrences of independent transition could have occurred also in other order (due to the diamond property, see Fact 1). Therefore, all trace equivalent sequences of every firing sequence of a given EN are also firing sequences of that EN. The partial order defined by a trace gives the entire causal relations between transitions in the EN (Kleijn and Koutny (2008)).

Mazurkiewicz Traces and partial orders are useful when we are dealing only with causality but they fall short when the relations between transitions get more complex.

This was the case for occurrence nets (processes) and its associated partial order and it is the case with traces the partial order generated by them. In dealing with that for processes, activator occurrence nets and stratified orders were introduced and similar approach had been defined for the theory of traces, namely Comtraces (combined trace) which we will introduce next.

7.2 Comtraces

Traces can be thought of as a language representation of partial orders, therefore they can only model what partial orders can (i.e. "true concurrency). However, we have seen that partial orders cannot model some aspects of concurrency such as the "no later than" relation. When an event a is performed "no later than" event b, then this relation can be modeled by the following set of two step sequences $x = \{\{a\}\{b\}, \{ab\}\}$. The set x cannot be modeled by a trace (or a partial order).

Faced with this limitation of traces, Janicki and Koutny proposed the comtrace (combined trace) notion Janicki and Koutny (1995) in which a relation *sim* (called simultaneity), and a congruence relation *ser* (called serializability) were defined. Those two relations serve two distinct purposes; *sim* defines valid steps, and *ser* defines valid ways to split such steps. More precisely, if $(a, b) \in sim$ then a and b may occur together in one step $\{ab\}$ while $(a, b) \in ser$ means that a and b may occur in one step $\{ab\}$ and such step can be split into the sequence $\{a\}\{b\}$. Clearly the relation *ser* is a subset of *sim*. The notion of comtrace is formally defined as follows.

Definition 24 (Janicki and Koutny (1995)).

1. *let Σ be a finite set and let $ser \subseteq sim \subset \Sigma \times \Sigma$ be two relation called serializability*

and simultaneity respectively and the relation sim is irreflexive and symmetric. Then the triple (Σ, sim, ser) *is called a comtrace alphabet.*

2. *We define the set of all potential step* $\mathbb{S}$ *as the set of all cliques of the graph* (Σ, sim) *as* $\mathbb{S} = \{A \mid A \neq \emptyset \wedge \forall a, b \in A.(a = b \vee (a, b) \in sim)\}$.

3. *let* $\Theta = (\Sigma, sim, sir)$ *be a comtrace alphabet and let* $\equiv_{ser}$*, called* comtrace congruence*, be the EQ-congruence defined by the set of equations* $EQ = \{A = BC \mid A = B \cup C \in \mathbb{S} \wedge A \times C \subseteq ser\}$.

4. *The equational monoid* $(\mathbb{S}^* / \equiv_{sim,ser}, \circ, [\lambda])$ *is called a monoid of comtraces (i.e. the set of all comtrace over a comtrace alphabet with comtrace concatenation and the empty comtrace forms a monoid).* □

For simplicity, $[x]_{(sim,ser)}$ is written instead of $[x]_{\equiv_{(sim,ser)}}$. Similarly to traces and partial orders (where each trace can be uniquely interpreted as a finite partial order), it can be shown that each comtrace uniquely defines a stratified order structure. This is defined as follows; see Janicki and Koutny (1995); Kleijn and Koutny (2008); Janicki and Lê (2011); Lê (2010) for more details.

Definition 25 (Janicki and Koutny (1995)). *For every comtrace* $\boldsymbol{x} = [x]_{(sim,ser)}$ *over* (Σ, sim, ser)*, let the set* $Strat(\boldsymbol{x}) = \{\lhd_t \mid t \in \boldsymbol{x}\}$ *be the set of all stratified orders defined by the elements of* $\boldsymbol{x}$*, and let* $S^{\boldsymbol{x}} = (\widehat{\Sigma_{\boldsymbol{x}}}, \prec_{\boldsymbol{x}}, \sqsubset_{\boldsymbol{x}})$ *be the relational structure given by:*

$$\prec_{\boldsymbol{x}} \;=\; \bigcap_{<\in Strat(\boldsymbol{x})} <, \qquad \sqsubset_{\boldsymbol{x}} \;=\; \bigcap_{<\in Strat(\boldsymbol{x})} <^{\frown}.$$

Then, for every comtrace $\boldsymbol{x} = [x]_{(sim,ser)}$ *over* (Σ, sim, ser)*, the relational structure* $S^{\boldsymbol{x}}$ *is a stratified order structure called* stratified order structure generated by the comtrace $\mathbf{x}$. □

Example 3. *Let* $\Sigma = \{a, b, c, d\}$*; and sim and ser defined as in Figure 7.2. Then the set of step sequences is* $[x]_{(sim,ser)} = \{\{a\}\{b\}\{c\}\{d\}, \{a\}\{b\}\{d\}\{c\}, \{a\}\{b, c\}\{d\}, \{a\}\{b\}\{c, d\}\}$ *is the comtrace generated by the step sequence* x. □

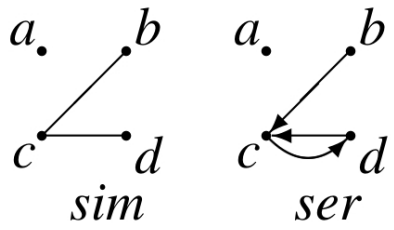


Figure 7.2: An example of relation *sim* (simultaneity), and congruence relation *ser* (serializability).

Comtraces can be applied to elementary nets with inhibitor arcs (defined in Section 4.2) to capture the intrinsic causality in their behaviours. To do so, we associate with ENI the comtrace alphabet as follows.

Definition 26. *The* ***comtrace alphabet of ENI*** *is the triple* (T, sim, ser) *where sim and ser are given respectively by:*

- $(t_1, t_2) \in sim$ *if* ${}^\bullet t_1^\bullet \cap {}^\bullet t_2^\bullet = {}^\circ t_2 \cap {}^\bullet t_1 = {}^\circ t_1 \cap {}^\bullet t_2 = \emptyset$.
- $(t_1, t_2) \in ser$ *if* $(t_1, t_2) \in sim$ *and* $t_1^\bullet \cap {}^\circ t_2 = \emptyset$. □

For the `ENI` in Figure 6.3, we have

$$sim = \{(t_2, t_3), (t_3, t_2), (t_2, t_5), (t_5, t_2), (t_3, t_4), (t_4, t_3), (t_4, t_5), (t_5, t_4)\}, \text{ and}$$

$$ser = sim \setminus \{(t_4, t_5)\}.$$

Let's consider two different step sequences, $y = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4, t_5\}\}$ (defined in Section 4.2) and $y' = \{\{t_1\}, \{t_2, t_3\}, \{t_1\}, \{t_4\}, \{t_5\}\}$. While y is a valid step sequence for `ENI` in Figure 6.3, y' is not since splitting $\{t_4, t_5\}$ into $\{t_4\}, \{t_5\}$ means that

t_4 will populate s_5 to which t_5 is connected by a inhibitor arc (i.e. t_5 cannot fire). This is dealt with in the comtrace alphabet by excluding (t_4, t_5) from ser. Similarly to traces and ENs, we may state that the causal behaviour of ENIs can be captured by the stratified order structure corresponding to the comtrace Kleijn and Koutny (2008).

Since traces theory (and comtrace) are based on sequential observations of a given system, the assumptions made about such observations are quite important. In fact, neither traces nor comtrace suffices to capture the behaviour of a system if observations are assumed to be interval orders (e.g. modelled as interval elementary net with inhibitor arcs). Therefore, interval traces were developed to address the shortcomings of traces and comtrace and it will be the topic of the next section.

7.3 Interval Traces

Interval traces, introduced in Janicki et al. (2012) and refined in Janicki and Yin (2015), stem from Mazurkiewicz traces and Fishburn's representation of interval orders. Since we will compare our model of interval process (to be defined in the next chapter) to interval traces, we will provide a more detailed overview of them.

7.3.1 Interval Traces Construction

Let Σ be a finite set (of events), and let

$$\mathcal{E}_\Sigma = \{Ba \mid a \in \Sigma\} \cup \{Ea \mid a \in \Sigma\},$$

be the set of all beginnings and ends of events in Σ. We will often just write $\mathcal{E}$ instead of $\mathcal{E}_\Sigma$. Every sequence from $x \in \mathcal{E}^*$ defines a total order $to(x)$, however not every

such total order can be interpreted as a representation of some interval order. For example, $BaBcBb$ represents no interval order (see Definition 15).

Definition 27 (Janicki et al. (2012)). *Let $x \in \mathsf{InSeq}(\mathcal{E}_\Sigma^*)$, and let $\blacktriangleleft_x$ be a relation on $\widehat{\Sigma}$, defined by*

$$a^{(i)} \blacktriangleleft_x b^{(j)} \iff Ea^{(i)} \lhd_x Bb^{(j)}.$$

By Theorem 1, the relation $\blacktriangleleft_x$ is an interval order, and it is called the interval order defined by the sequence x of beginnings and ends. □

For example if $x = BaEaBbBcEbBdEcEd$ then $\blacktriangleleft_x$ is the interval order $<_3$ from Figure 2.1.

Definition 28 (Janicki et al. (2012)). *Let $ind \subseteq \mathcal{E} \times \mathcal{E}$ be a symmetric and irreflexive relation such that for all $a, b \in \Sigma$*

1. *$(Ba, Ea) \notin ind$ and $(Ea, Ba) \notin ind$,*
2. *$(Ba, Bb) \in ind$ and $(Ea, Eb) \in ind$.*

*The relation ind is called **interval independence**, and the pair $(\mathcal{E}, ind)$ is called **interval trace alphabet**.* □

The condition (1) above follows from the fact that in any representation of any order, the beginning of an event always precede the end so that cannot commute. The condition (2) follows from the generalization of the observation that the interval sequences $BaBbEaEb$, $BbBaEaEb$, $BaBbEbEa$, and $BbBaEbEa$ represent the same fact, namely that a and b are simultaneous.

The interval traces are defined as a special distinctive class Mazurkiewicz traces.

Definition 29 (Janicki et al. (2012)). *A trace $[x]_{ind}$ over the interval trace alphabet $(\mathcal{E}, ind)$ is called an **interval trace** if $[x]_{ind} \subseteq \mathsf{InSeq}(\mathcal{E}^*)$.* □

The soundness of the above definition follows from the following non-trivial result.

Proposition 3 (Janicki and Yin (2015)). *Let $(\mathcal{E}, ind)$ be an interval trace alphabet, and let $x, y \in \mathsf{InSeq}(\mathcal{E}^*)$.*

1. *For each $x, y \in \mathcal{E}^*$, if $x \in \mathsf{InSeq}(\mathcal{E}^*)$ and $y \in \mathsf{InSeq}(\mathcal{E}^*)$ then $xy \in \mathsf{InSeq}(\mathcal{E}^*)$.*

2. *For each $s \in \mathcal{E}^*$, we have:*
   $s \in \mathsf{InSeq}(\mathcal{E}^*) \iff \forall x \in [s]_{ind}.\ x \in \mathsf{InSeq}(\mathcal{E}^*)$.

3. *For each $x, y \in \mathcal{E}^*$,*
   if $[x]_{ind} \subseteq \mathsf{InSeq}(\mathcal{E}^*)$ and $[y]_{ind} \subseteq \mathsf{InSeq}(\mathcal{E}^*)$, then $[x]_{ind}[y]_{ind} = [xy]_{ind} \subseteq \mathsf{InSeq}(\mathcal{E}^*)$.

4. *$\blacktriangleleft_x = \blacktriangleleft_y \Rightarrow x \equiv_{ind} y$.* □

As a partial orders generator, each interval trace can be interpreted twofold. First, it is also a Mazurkiewicz trace so it generates a partial order by Definition 22, second, each element of the interval trace is an interval sequence, so the trace can also be interpreted as representing a set of appropriate interval orders.

Definition 30. *Let $[x] \subseteq \mathsf{InSeq}(\mathcal{E}^*)$ be an interval trace.*

1. *The partial order $\lessdot_{[x]}^{trace}$ defined as:*
   $$\lessdot_{[x]}^{trace} = \bigcap_{s \in [x]} \lhd_s$$
   *is called **canonical order** defined by $[x]$.*

2. *The set* $\mathsf{interv}^{trace}([x]) = \{\blacktriangleleft_t |\ t \in [x]\}$

   is the set of all ***interval orders defined by*** $[x]$. □

Both the canonical order and the interval orders defined by an interval trace will be used to show the equivalence of interval order semantics and interval process semantics for elementary inhibitor nets.

7.3.2 Interval Traces Semantics for Elementary Net with Inhibitor Arcs

Let $\mathtt{N} = (P, T, F, I, C_{init})$ be an ENI system and let $\mathcal{CN} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$ be its *complement closed interval representation*.

We define the ***interval trace independency*** relation $ind_{\mathcal{CN}} \subseteq \mathcal{T} \times \mathcal{T}$ as follows.

Definition 31. *For all distinct* $a, b \in T$*:*

1. $(Ba, Bb) \in ind_{\mathcal{CN}} \wedge (Ea, Eb) \in ind_{\mathcal{CN}}$

2. $(Ba, Eb) \in ind_{\mathcal{CN}} \iff$

$$[(Ba^\bullet \cup {}^\bullet Ba) \cap (Eb^\bullet \cup {}^\bullet Eb) = \emptyset] \wedge$$
$$[(Ba^\circ \cap {}^\bullet Eb) \cup (Eb^\circ \cap {}^\bullet Ba) = \emptyset] \wedge$$
$$[(Ba^\bullet \cap Eb^\circ) \cup (Eb^\bullet \cap Ba^\circ) = \emptyset].$$

The ***interval trace alphabet*** *is* $(\mathcal{T}, ind_{\mathcal{CN}})$. □

The relation $ind_{\mathcal{CN}}$ is a refinement of the similar relations from Janicki and Koutny (1995); Kleijn and Koutny (2004).

Definition 32. *Let* $x = \alpha_1 \dots \alpha_n$ *be an interval firing sequence of* $\mathcal{CN}$*. The* interval trace $[x]_{ind_{\mathcal{CN}}}$ *is the* ***interval trace of*** $\mathcal{CN}$ ***generated by*** x**.** □

Proposition 1 and the result below prove the soundness of the above definition.

Proposition 4 (Janicki and Yin (2015)). *If x is an interval firing sequence of $\mathcal{CN}$, $\mathcal{C}_{init}[\![x\rangle\!\rangle\mathcal{C}_n$ for some n, and $y \in [x]_{ind_{\mathcal{CN}}}$, then $\mathcal{C}_{init}[\![y\rangle\!\rangle\mathcal{C}_n$.* □

For the net $\mathcal{CN}$ in Figure 8.1 we have:

$ind_{\mathcal{CN}}\{(Ba, Bb), (Ba, Eb), (BaEc), (Ba, Bc), (Ea, Eb), (Ea, Ec), (Bb, Ba), (Bb, Ec),$
$(Eb, Ba), (Eb, Ea), (Eb, Ec), (Bc, Ba), (Ec, Ba), (Ec, Ea), (Ec, Bb), (Ec, Eb)\}$. We also have:

$[BaEaBbEbBcEc]_{ind_{\mathcal{CN}}} = \{BaEaBbEbBcEc\}$ and

$[BcEcBaEaBbEb]_{ind_{\mathcal{CN}}} = \{BcEcBaEaBbEb, BaBcEcEaBbEb, BaBcEaEcBbEb,$
$BcBaEcEaBaEb, BaBcEaEcBbEb, BaBcEaBbEbEc, BaBcEaBbEcEb,$
$BcBaEaBbEbEc, BaBcEaBbEcEb\}$

Note that the fact that $(Ba, Bb) \in ind_{\mathcal{CN}}, (Ea, Eb) \in ind_{\mathcal{CN}}$, and $(Ba, Eb) \in ind_{\mathcal{CN}}$ are never used, as there is no extended firing sequence x starting from $\{s_1, s_2\}$ such that $x = uBaBbw, x = uEaEbw$ or $x = uBaEbw$, so $ind_{\mathcal{CN}}$ is bigger than needed. This is due to deriving the independency relation directly from the static structure of the net Janicki and Yin (2015).

Chapter 8

Interval Processes and Interval Order Structures

We will now introduce interval processes and show how they represent interval runs/ executions as well as how they relate to interval order structures. Then we will compare interval processes with interval traces and show that they describe the same concurrent histories (represented by interval order structures) which can be seen as a *validation of our approach*.

Let $\mathtt{N} = (P, T, F, I, C_{init})$ be an ENI, $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$ be its interval representation and let $x = \alpha_1 \dots \alpha_n$ be an interval firing sequence of $\mathcal{N}$. Since $\mathcal{N}$ is just another inhibitor nets, we can use Algorithm 3 and produce a process (an occurrence net) $\mathcal{P}_x^{\mathsf{iseq}}$ generated by $x = \alpha_1 \dots \alpha_n$. For example, let's consider the $\mathcal{N}$ in Figure 8.1. When, Algorithm 3 is applied to $\mathcal{N}$ (actually $\mathcal{CN}$ as the presence of inhibitor arcs requires introducing complement places, see Section 6.3) with $x = BaBcEaEcBbEb$, it generates $\mathcal{P}_x^{iseq}$ in Figure 8.1.

Assume that $\mathcal{P}_x^{\mathsf{iseq}} = \mathcal{N}_n = (\mathcal{B}_n, \mathcal{E}_n, \mathcal{R}_n, \mathcal{A}_n)$, where $\mathcal{N}_n$ is the last step of Algorithm

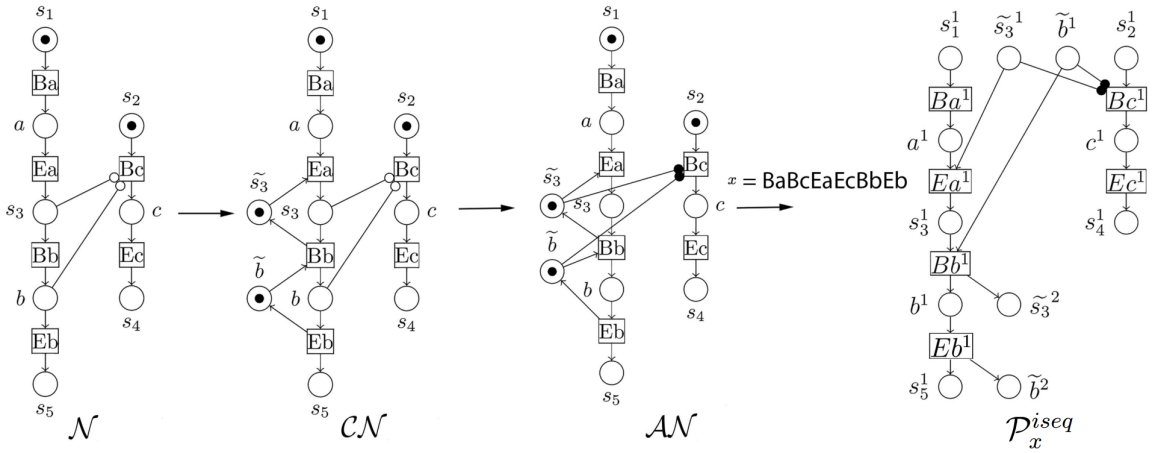


Figure 8.1: Process construction for $\mathcal{N}$ (interval representation of N from Figure 5.1).

3.

Similarly to processes generated by ENIs, where every process is associated with some partial order, we can formally define a partial order $\preccurlyeq_x$ derived from the process $\mathcal{P}_x^{\mathsf{iseq}}$ as in Definition 19. After all, $\mathcal{N}$ is just another ENI, however, such a partial order will be used to define an interval order structures. In Figure 8.2, $\mathcal{P}_x^{\mathsf{iseq}}$ produces $\preccurlyeq_x^{init}$, and $\preccurlyeq_x^{init}$ produces $\preccurlyeq_x^{proc}$.

In most cases many different x's can generate the same process $\mathcal{P}_x^{\mathsf{iseq}}$. In fact if x is any sequence of $\{BcEcBaEaBbEb, BaBcEcEaBbEb, BaBcEaEcBbEb,$ $BcBaEcEaBaEb, BaBcEaEcBbEb, BaBcEaBbEbEc, BaBcEaBbEcEb,$ $BcBaEaBbEbEc, BaBcEaBbEcEb\}$, then the same process $\mathcal{P}_x^{\mathsf{iseq}}$ would be produced. The idea is that if $\mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}$ then x and y are different observations of the same behaviour, so they are equivalent (w.r.t. concurrent behaviour) c.f. Kleijn and Koutny (2004, 2008). Take $\{BcEcBaEaBbEb,$ and $BaBcEcEaBbEb\}$, in both sequences we have the c is no later than a as in the first sequence, it starts and ends before a starts, and in the second one, it starts and end before the end of a. This is simply what

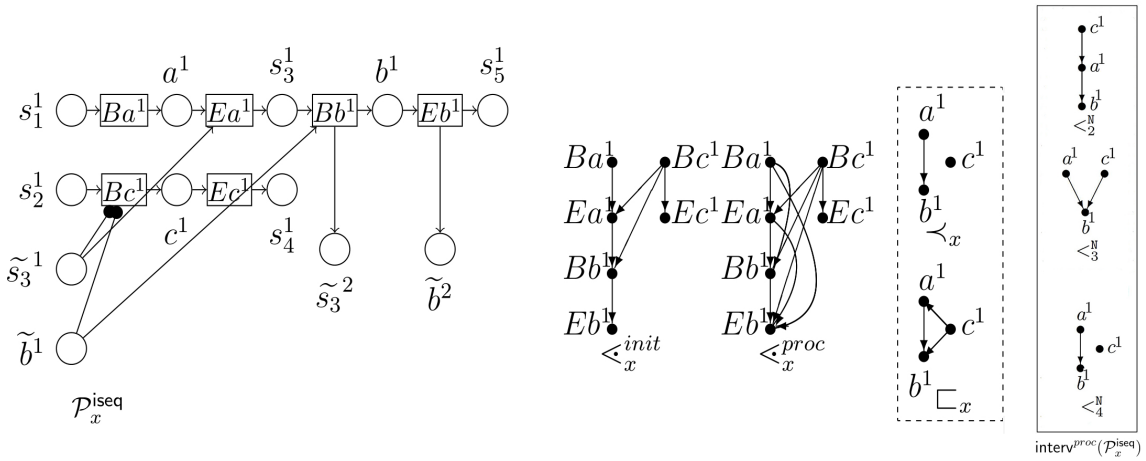


Figure 8.2: An example of a process $\mathcal{P}_x^{\mathsf{iseq}}$, the directed acyclic graph $\lessdot_x^{init}$, the partial order $\lessdot_x^{proc}$, the relations $\prec_x$, $\sqsubset_x$ and the interval order structure $S^x = (\{a^1, b^1, c^1\}, \prec_x, \sqsubset_x)$. The net here is $\mathcal{N}$ from Figure 5.1, and $x = BaBcEaEcBbEb$.

equivalent (w.r.t. concurrent behaviour) means.

Recall that for every sequence x, $\lhd_x$ denotes a total order defined by x (see Notation 1(3)) and $\lessdot_x^{proc}$ is defined in Definition 19.

The partial order $\lessdot_x^{proc}$ is derived from the process $\mathcal{P}_x^{\mathsf{iseq}}$ by just deleting places and preserving causality among transitions, so technically some information is lost. The lemma below shows that in reality no essential information is lost, so the partial order $\lessdot_x^{proc}$ can be interpreted as a faithful representation of the process $\mathcal{P}_x^{\mathsf{iseq}}$.

Lemma 2.

1. *Let x, y be interval firing sequences. Then:*

$$\lessdot_x^{proc} = \lessdot_y^{proc} \iff \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}.$$

2. *For each interval firing sequence x, $\mathsf{total}(\lessdot_x^{proc}) = \{\lhd_y \mid \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}\}$.* □

Proof. (1) Clearly $\mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}} \Rightarrow \lessdot_x^{proc} = \lessdot_y^{proc}$. We need only to show $\lessdot_x^{proc} = \lessdot_y^{proc} \Rightarrow \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}$.

To prove this we first recall how finite partial orders can induce Mazurkiewicz traces (c.f. Diekert et al. (1995); Janicki et al. (2010)). Let $<$ be a finite partial order and $\lhd = \alpha_1 \to \ldots \to \alpha_k$ be its total extension. The total order $\lhd$ can be represented by a sequence $\alpha_1 \ldots \alpha_k$ in a standard way. Therefore, we can identify $\lhd$ with $\alpha_1 \ldots \alpha_k$, and, by a small abuse of notation, write $\lhd = \alpha_1 \ldots \alpha_k$. Define the relation $\approx_<$ on total extensions of $<$ as follows:

$$\lhd_1 \approx_< \lhd_2 \iff \lhd_1 = x_1\alpha\beta x_2 \wedge \lhd_2 = x_1\beta\alpha x_2 \wedge \alpha \frown_< \beta.$$

Now define $\equiv_< = \approx_<^*$. The pair $(\{\alpha_1, \ldots, \alpha_k\}, \frown_<)$ is a legal trace alphabet and $\mathsf{total}(<) = [\lhd]_{\equiv_<}$ for any $\lhd \in \mathsf{total}(<)$.

Now it suffices to show that $\lhd_x, \lhd_y \in \mathsf{total}(\lessdot_x^{proc})$ and $\lhd_x \approx_{\lessdot_x^{proc}} \lhd_y \Rightarrow \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}$. First we show $\lhd_x \in \mathsf{total}(\lessdot_x^{proc})$. Let $x = \alpha_1 \ldots \alpha_n$. Let $(\mathsf{P}_x^{\mathsf{seq}})^i$ denote a process defined by the prefix $\alpha_1 \ldots \alpha_i$ of x, and let $(\lessdot_x^{init})^i$ denote a directed acyclic graph generated by $(\mathsf{P}_x^{\mathsf{seq}})^i$. By Algorithm 3, $(\lessdot_x^{init})^{i+1}$ is derived from $(\lessdot_x^{init})^i$ by adding α_{i+1}, but we always have $\neg(\alpha_{i+1}(\lessdot_x^{init})^{i+1}\alpha_k)$ for all $k \leq i$. Hence $\lhd_x$ is an extension of $\lessdot_x^{proc}$, so $\lhd_x \in \mathsf{total}(\lessdot_x^{proc})$, and consequently $\lhd_y \in \mathsf{total}(\lessdot_x^{proc})$. Let $\lhd_x = x_1\alpha\beta x_2$, $\lhd_y = x_1\beta\alpha x_2$ and $\alpha \frown_{\lessdot_x^{proc}} \beta$. Since $\alpha \frown_{\lessdot_x^{proc}} \beta$, then $\neg(\alpha \lessdot_x^{init} \beta)$ and $\neg(\beta \lessdot_x^{init} \alpha)$, so from Definition 19 we have: $(\alpha^\bullet \cup {}^\bullet\alpha) \cap (\beta^\bullet \cup {}^\bullet\beta) = \emptyset$, $\alpha^\circ \cap ({}^\bullet\beta \cup \beta^\bullet) = \emptyset$ and $\beta^\circ \cap ({}^\bullet\alpha \cup \alpha^\bullet) = \emptyset$, which from Algorithm 3 immediately implies $\mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}$.

(2) From the proof of (1) we have $\mathsf{total}(\lessdot_x^{proc}) = [\lhd_x]_{\equiv_{\lessdot_x^{proc}}} = \{\lhd_y \mid \lessdot_x^{proc} = \lessdot_y^{proc}\}$, which, by (1) of this lemma, gives $\mathsf{total}(\lessdot_x^{proc}) = \{\lhd_y \mid \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}\}$. □

Lemma 2(2) states that total orders defined by all sequences that can generate

a process (occurrence activator net) $\mathcal{P}_x^{\mathsf{iseq}}$ are just total extensions of a partial order $\lessdot_x^{proc}$ that is defined by the process $\mathcal{P}_x^{\mathsf{iseq}}$.

We will now formally define interval orders and interval order structures generated by interval firing sequences of $\mathcal{N}$ using interval processes semantics. Interval order structures represent concurrent histories, for every interval order structure S, the set of interval orders $interv(S)$ (see Theorem 4) represents equivalent observations belonging to the same concurrent history. We will show that the same set of equivalent observations can directly be derived from a given interval process.

Definition 33. *Let* $x \in \mathsf{InSeq}(\mathcal{T}^*)$, $\mathcal{P}_x^{\mathsf{iseq}} = \mathcal{N}_n = (\mathcal{B}_n, \mathcal{E}_n, \mathcal{R}_n, \mathcal{A}_n)$ *be the process generated by* x, *and let* $\widehat{E}_n = \{t^i \mid Bt^i \in \mathcal{E}_n \wedge Et^i \in \mathcal{E}_n\} \subseteq \widehat{T}$.
We define the relations $\blacktriangleleft_x, \prec_x, \sqsubset_x$ *on* $\widehat{E}_n$, *and the tuple* S^x *as follows:*

1. $a^i \blacktriangleleft_x b^j \iff^{df} Ea^i \lhd_x Bb^j$,
2. $a^i \prec_x b^j \iff^{df} Ea^i \lessdot_x^{proc} Bb^j$,
3. $a^i \sqsubset_x b^j \iff^{df} Ba^i \lessdot_x^{proc} Eb^j$, *and*
4. $S^x = (\widehat{E}_n, \prec_x, \sqsubset_x)$. □

The above definition allows us to connect interval processes with appropriate interval structures.

Corollary 1.

1. *The relation* $\blacktriangleleft_x$ *is an* ***interval order****.*
2. *The tuple* S^x *is an* ***interval order structure****.*

3. $\mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}} \iff S^x = S^y$. □

Proof. By Theorem 1 we have (1) and by Theorem 5 we have (2). From (2) and (3) of Definition 33, we have $\lessdot_x^{proc} = \lessdot_y^{proc} \iff S^x = S^y$, which by Lemma 2(1), implies (3). □

The interval order structure S^x will be called ***induced by process*** $\mathcal{P}_x^{\mathsf{iseq}}$.

Each $\mathcal{P}_x^{\mathsf{iseq}}$ is generated from $\mathcal{N}$ by an interval sequence x and each interval sequence x defines an interval order $\blacktriangleleft_x$. The set of all interval orders that can be derived from $\mathcal{P}_x^{\mathsf{iseq}}$ or $\lessdot_x^{proc}$ is defined as follows.

Definition 34. *For each interval firing sequence x, we define,*

1. $\mathsf{interv}^{ord}(\lessdot_x^{proc}) = \{\blacktriangleleft_y |\ \lhd_y \in \mathsf{total}(\lessdot_x^{proc})\}$.

2. $\mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}}) = \{\blacktriangleleft_y |\ \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}\}$. □

The main result of this chapter states that for every interval firing sequence x, an interval process $\mathcal{P}_x^{\mathsf{iseq}}$ and interval order structure S^x describe the same concurrent behaviour, so they can be seen as equivalent concepts.

Theorem 6. *For each interval firing sequence x,*

$$\mathsf{interv}^{str}(S^x) = \mathsf{interv}^{ord}(\lessdot_x^{proc}) = \mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}}).$$

□

Proof. First we prove $\mathsf{interv}^{ord}(\lessdot_x^{proc}) = \mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}})$. By Lemma 2(2), we have $\mathsf{total}(\lessdot_x^{proc}) = \{\lhd_y \mid \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}\}$. Hence $\blacktriangleleft_y \in \mathsf{interv}^{ord}(\lessdot_x^{proc}) \iff \lhd_y \in$

$\mathsf{total}(\lessdot_x^{proc}) \iff \mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}} \overset{\text{Def.34(2)}}{\iff} \blacktriangleleft_y \in \mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}})$. Thus $\mathsf{interv}^{ord}(\lessdot_x^{proc}) = \mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}})$.

We will now show $\mathsf{interv}^{str}(S^x) = \mathsf{interv}^{ord}(\lessdot_x^{proc})$. First we will prove $\mathsf{interv}^{ord}(\lessdot_x^{proc}) \subseteq \mathsf{interv}^{str}(S^x)$. Let $\blacktriangleleft_y \in \mathsf{interv}^{ord}(\lessdot_x^{proc})$, i.e. $\mathcal{P}_x^{\mathsf{iseq}} = \mathcal{P}_y^{\mathsf{iseq}}$. Consider the relation $\prec_x$. We have: $a^i \prec b^j \overset{\text{Def.33(2)}}{\iff} Ea^i \lessdot_x^{proc} Bb^j \overset{\mathcal{P}_x^{\mathsf{iseq}}=\mathcal{P}_y^{\mathsf{iseq}}}{\iff} Ea^i \lessdot_y^{proc} Bb^j \overset{\text{Lem.2(2)}}{\Rightarrow} Ea^i \lhd_y Bb^j \overset{\text{Def.33(1)}}{\iff} a^i \blacktriangleleft_y b^i$. Hence $\blacktriangleleft_y$ is an extension of $\prec_x$. For the relation $\sqsubset_x$ we have: $a^i \sqsubset b^j \overset{\text{Def.33(3)}}{\iff} Ba^i \lessdot_x^{proc} Eb^j \overset{\mathcal{P}_x^{\mathsf{iseq}}=\mathcal{P}_y^{\mathsf{iseq}}}{\iff} Ba^i \lessdot_y^{proc} Eb^j \overset{\text{Lem.2(2)}}{\Rightarrow} Ba^i \lhd_y Eb^j \iff \neg(Eb^j \lhd_y Ba^i) \overset{\text{Def.33(1)}}{\iff} \neg(b^j \blacktriangleleft_y a^i) \iff a^i \blacktriangleleft_y^{\frown} b^j$, so $\blacktriangleleft_y$ extends $\sqsubset_x$ too. Hence $\blacktriangleleft_y \in \mathsf{interv}^{str}(S^x)$, i.e. $\mathsf{interv}^{ord}(\lessdot_x^{proc}) \subseteq \mathsf{interv}^{str}(S^x)$.

Now we show $\mathsf{interv}^{str}(S^x) \subseteq \mathsf{interv}^{ord}(\lessdot_x^{proc})$. Let $\blacktriangleleft \in \mathsf{interv}^{str}(S^x)$ and let $\lhd_{\blacktriangleleft}$ be a total order representation of $\blacktriangleleft$ via Theorem 1, i.e. $a^i \blacktriangleleft b^j \iff Ea^i \lhd_{\blacktriangleleft} Bb^j$. Let $x_{\blacktriangleleft} \in \mathcal{E}_n$ be the sequence representation of the total order $\lhd_{\blacktriangleleft}$, i.e. $\lhd_{\blacktriangleleft} = \lhd_{x_{\blacktriangleleft}}$, where $\lhd_{x_{\blacktriangleleft}}$ is the total order generated by $x_{\blacktriangleleft}$. By Definition 33, $\blacktriangleleft = \blacktriangleleft_{x_{\blacktriangleleft}}$. To show that $\blacktriangleleft \in \mathsf{interv}^{ord}(\lessdot_x^{proc})$, it suffice to show that $\lhd_{\blacktriangleleft} \in \mathsf{total}(\lessdot_x^{proc})$. Since $\blacktriangleleft \in \mathsf{interv}^{str}(S^x)$, and $S^x = (\mathcal{E}_n, \prec_x, \sqsubset_x)$, then $\prec_x \subseteq \blacktriangleleft$ and $\sqsubset_x \subseteq \blacktriangleleft^{\frown}$. To prove $\lhd_{\blacktriangleleft} \in \mathsf{total}(\lessdot_x^{proc})$, we need to show that for all $\alpha, \beta \in \{Ba^i, Ea^i, Bb^j, Eb^j\}$ we have $\alpha \lessdot_x^{proc} \beta \Rightarrow \alpha \lhd_{\blacktriangleleft} \beta$. First note that by Theorem 5(1) and Theorem 1(1) we already have $Ba^i \lessdot_x^{proc} Ea^i$, $Bb^j \lessdot_x^{proc} Eb^j$, $Ba^i \lhd_{\blacktriangleleft} Ea^i$ and $Bb^j \lhd_{\blacktriangleleft} Eb^j$, so only four cases remain.

Case 1: $\alpha = Ea^i$ and $\beta = Bb^j$. We have $Ea^i \lessdot_x^{proc} Bb^j \overset{\text{Def.33(2)}}{\iff} a^i \prec_x b^j \Rightarrow a^i \blacktriangleleft b^j \overset{\text{Th.1}}{\iff} Ea^i \lhd_{\blacktriangleleft} Bb^j$. Hence $\lhd_{\blacktriangleleft} \in \mathsf{total}(\lessdot_x^{proc})$.

Case 2: $\alpha = Eb^j$ and $\beta = Ba^i$. Similarly as Case 1.

Case 3: $\alpha = Ba^i$ and $\beta = Eb^j$. We have $Ba^i \lessdot_x^{proc} Eb^j \overset{\text{Def.33(3)}}{\iff} a^i \sqsubset_x b^j \Rightarrow a^i \blacktriangleleft^{\frown} b^j \overset{\text{Th.1}}{\iff} Ba^i \lhd_{\blacktriangleleft} Eb^j$.

Case 4: $\alpha = Bb^j$ and $\beta = Ea^i$. Similarly as Case 3.

Hence $\lhd_{\blacktriangleleft} \in \mathsf{total}(\lessdot_x^{proc})$, so by Definition 34(1), $\blacktriangleleft \in \mathsf{interv}^{ord}(\lessdot_x^{proc})$, i.e. $\mathsf{interv}^{str}(S^x) \subseteq \mathsf{interv}^{ord}(\lessdot_x^{proc})$. □

This means the relationship between interval processes and interval order structures is the same as that between stratified processes and stratified order structures described and analyzed in Janicki et al. (2010); Kleijn and Koutny (2004, 2008). Figure 8.2 illustrates the relationships between $\mathcal{P}_x^{\mathsf{iseq}}$, $\lessdot_x^{init}$, $\lessdot_x^{proc}$ and $S^x = (\widehat{E}_n, \prec_x, \sqsubset_x)$, for a given $x = BaBcEaEcBbEb$ and the net $\mathcal{N}$ from Figure 5.1.

Note that the relationship from Figure 8.2 is valid for any $x \in \{BaEaBbEbBcEc$, $BcEcBaEaBbEb$, $BaBcEcEaBbEb$, $BaBcEaEcBbEb$, $BcBaEcEaBbEb$, $BcBaEaEcBbEb$, $BaBcEaBbEbEc$, $BaBcEaBbEcEb$, $BcBaEaBbEbEc$, $BcBaEaBbEcEb\}$. Also for each x from above, $\mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}}) = \mathsf{interv}^{str}(S^x) = \{<_2^{\mathtt{N}}, <_3^{\mathtt{N}}, <_4^{\mathtt{N}}\}$, where $<_2^{\mathtt{N}}, <_3^{\mathtt{N}}, <_4^{\mathtt{N}}$ are these of Figure 1.1 (when a^1, b^1, c^1 are replaced by a, b, c).

Considering the net $\mathtt{N}_{io}$ from Figure 1.1, N_{io} can generate only interval behaviours. It generates neither sequences nor step sequences that start from the marking $\{s_1, s_2\}$ and end at $\{s_4, s_5\}$. This also means that the nets $\mathtt{N}_{io}$, $\mathtt{CN}_{io}$, and $\mathtt{AN}_{io}$ generate no appropriate process, if the process derivation is based on a firing sequence or firing step sequence, and this includes all techniques presented in Busi and Pinna (1999); Janicki and Koutny (1995); Juhás et al. (2007); Kleijn and Koutny (2004); Montanari and Rossi (1995); Vogler et al. (1998); Winkowski (1998) and all their modifications. On the other hand, the interval representation of $\mathtt{N}_{io}$, the net $\mathcal{N}_{io}$ generates interval sequences, for example $z = BaBcEaBbEcEb$, that lead from $\{s_1, s_2\}$ to $\{s_4, s_5\}$. The interval sequence z generates the process $\mathcal{P}_z^{\mathsf{iseq}}$, which in turn describes the interval order $\lessdot_z$ which equals $<_4^{\mathtt{N}}$ (again with a^1, b^1, c^1 replaced by a, b, c). Since $<_4^{\mathtt{N}}$ is the

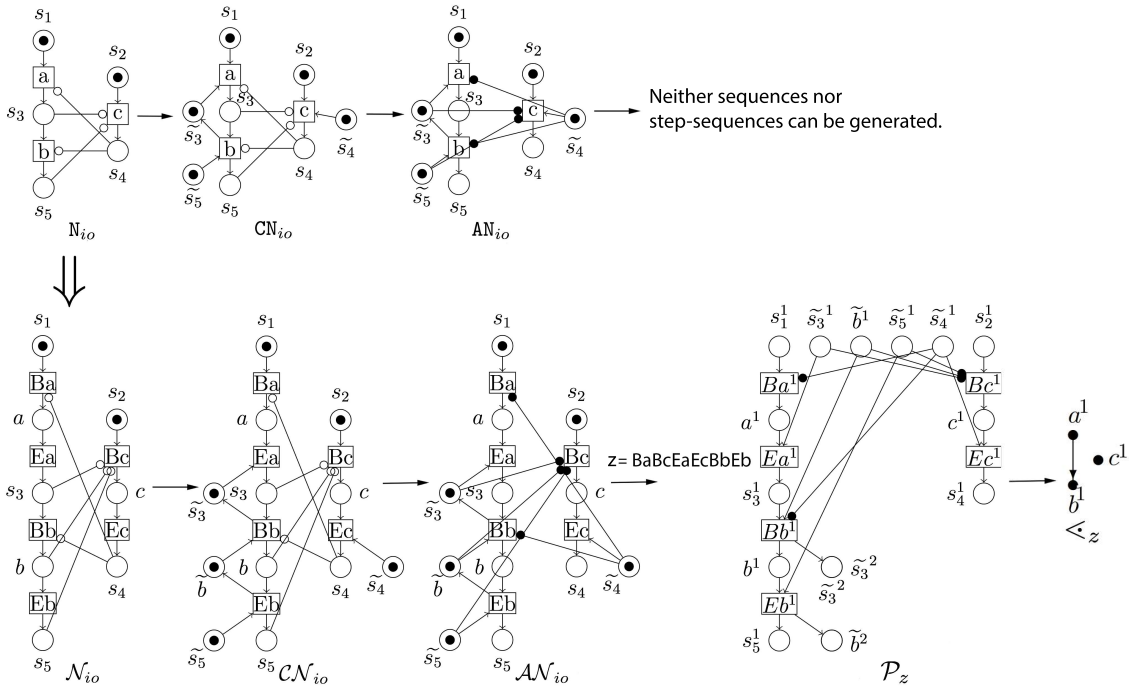


Figure 8.3: An example of an ENI that generates *only* interval orders. Our method results in the process $\mathcal{P}_z^{\mathsf{iseq}}$ and the interval order $\prec_z$, which is isomorphic to $<_4^{\mathtt{N}}$ of Figure 1.1, while all techniques based on either firing sequences or firing step sequences produce *empty* set.

only observation generated by $\mathtt{N}_{io}$, we have $S^z = (\{a^1, b^1, c^1\}, \prec_z, \sqsubset_z)$, where $\prec_z = <_4^{\mathtt{N}}$ and $\sqsubset_z = (<_4^{\mathtt{N}})^\frown$, and $\mathsf{interv}^{proc}(\mathcal{P}_z^{\mathsf{iseq}}) = \mathsf{interv}^{str}(S^z) = \{<_4^{\mathtt{N}}\}$. This all is illustrated in Figure 8.3.

We will now show that the model based on the concept of step sequences and stratified processes can be defined in terms of interval processes with identical results, which could be seen as a *validation of our approach*. However, first we need to recall some established results about the relation between stratified order structures and processes of elementary nets with inhibitor arcs.

Let $\mathtt{N} = (P, T, F, I, C_{init})$ be an ENI system, $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$ be its interval representation, $x = A_1 \dots A_n$ be a *firing step sequence* of $\mathtt{N}$, and let $ifs(x)$ be the *set*

of interval firing sequences of $\mathcal{N}$ corresponding to x, i.e.

$$ifs(x) = \{z \mid z = x_1 \dots x_n, x_i \in A_i^{BE}, \text{ for } i = 1 \dots n\}.$$

Let $\overline{x} \in ifs(x)$ and assume that $length(\overline{x}) = k$. Let $\mathsf{P}_x^{\mathsf{step}} = N_n = (B_k, E_k, R_k, A_k)$ be a process derived from $\mathtt{N}$ by using Algorithm 4 and step firing sequence x, and let $\mathcal{P}_{\overline{x}}^{\mathsf{iseq}} = \mathcal{N}_n = (\mathcal{B}_k, \mathcal{E}_k, \mathcal{R}_k, \mathcal{A}_k)$ be a process derived from $\mathcal{N}$ by using Algorithm 3 and interval firing sequence $\overline{x}$.

We also define:

$$\mathsf{strat}^{proc}(\mathsf{P}_x^{\mathsf{step}}) = \{<_y | \ \mathsf{P}_x^{\mathsf{step}} = \mathsf{P}_y^{\mathsf{step}}\}$$

(c.f. Kleijn and Koutny (2004)), and for each set X, let $\mathtt{strat}(X)$ denote the set of all stratified orders on X. The following result has been proved in Janicki and Koutny (1995).

Theorem 7 (Janicki and Koutny (1995); Kleijn and Koutny (2004)). *For every firing step sequence x,*

$$\mathsf{strat}^{proc}(\mathsf{P}_x^{\mathsf{step}}) = \mathsf{strat}^{str}(S^x),$$

where S^x is a stratified order structure derived from $\mathsf{P}_x^{\mathsf{step}}$. □

The following two results show that for firing step sequences, i.e. when runs are represented by stratified orders, standard stratified order processes of Janicki and Koutny (1995); Kleijn and Koutny (2004) and our interval processes produce the same results.

Lemma 3. *For each firing step sequences x, y and each $\overline{x} \in ifs(x)$, $\overline{y} \in ifs(y)$, we have*

$$\mathsf{P}_x^{\mathsf{step}} = \mathsf{P}_y^{\mathsf{step}} \iff \mathcal{P}_{\overline{x}}^{\mathsf{iseq}} = \mathcal{P}_{\overline{y}}^{\mathsf{iseq}}.$$

□

Proof. The proof uses the fact that processes are also a kind of activator nets.

Let N be an elementary net with inhibitor arcs, let $\mathsf{CN} = (P, T, F, I, C_{init})$ be its complement closed (if $\mathsf{N} \neq \mathsf{CN}$) and, let $\mathcal{CN} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$ be the interval representation of CN. Consider the step sequence x and the process $\mathsf{P}_x^{\mathsf{step}} = (B_n, E_n, R_n, A_n)$ derived by Algorithm 4. Let $C_{init}^1 = B_0 = \{p^1 \mid p \in C_{init}\}$. The quintuple $\mathsf{N}(\mathsf{P}_x^{\mathsf{step}}) = (B_n, E_n, R_n, A_n, C_{init}^1)$ is a well defined conflict free (see Definition 9) elementary net with activator arcs. Define $C_{\mathit{final}} = \{p \mid p \in B_n \wedge p^\bullet = \emptyset\}$. Let $\overline{x} \in ifs(x)$. The process $\mathcal{P}_{\overline{x}}^{\mathsf{iseq}} = (\mathcal{B}_m, \mathcal{E}_m, \mathcal{R}_m, \mathcal{A}_m)$ also can be regarded as a conflict free elementary net with activator arcs $\mathcal{N}(\mathcal{P}_{\overline{x}}^{\mathsf{iseq}}) = (\mathcal{B}_m, \mathcal{E}_m, \mathcal{R}_m, \mathcal{A}_m, C_{init}^1)$. Note that we have $C_{\mathit{final}} = \{p \mid p \in \mathcal{B}_m \wedge p^\bullet = \emptyset\}$. There is a special relationship between CN and $\mathsf{N}(\mathsf{P}_x^{\mathsf{step}})$ and a similar relationship between $\mathcal{CN}$ and $\mathcal{CP}_{\overline{x}}^{\mathsf{iseq}}$.

Namely, for every step sequence y, we have $\mathsf{P}_x^{\mathsf{step}} = \mathsf{P}_y^{\mathsf{step}}$ iff there is a marking $C_x \subseteq P$ such that $l(C_{\mathit{final}}) = C_x$ and $C_{init}[y\rangle C_x \iff C_{init}^1[y\rangle C_{\mathit{final}}$, where the left firing (i.e. $[y\rangle$) is in CN while the right $[y\rangle$ is in $\mathsf{N}(\mathsf{P}_x^{\mathsf{step}})$.

Moreover, we have $\mathcal{P}_{\overline{x}}^{\mathsf{iseq}} = \mathcal{P}_{\overline{y}}^{\mathsf{iseq}}$ iff $C_{init}[\![\overline{y}\rangle\!\rangle C_x \iff C_{init}^1[\![\overline{y}\rangle\!\rangle C_{\mathit{final}}$, for the same C_x as above, where the left firing $[\![\overline{y}\rangle\!\rangle$ is in $\mathcal{CN}$ while the right $[\![\overline{y}\rangle\!\rangle$ is in $\mathcal{N}(\mathcal{P}_x^{\mathsf{iseq}})$. These two properties are direct consequences of Algorithm 4 applied to CN and Algorithm 3 applied to $\mathcal{CN}$. From Proposition 2 it follows $C_{init}[y\rangle C_x \iff C_{init}[\![\overline{y}\rangle\!\rangle C_x$, which immediately implies $\mathsf{P}_x^{\mathsf{step}} = \mathsf{P}_y^{\mathsf{step}} \iff \mathcal{P}_{\overline{x}}^{\mathsf{iseq}} = \mathcal{P}_{\overline{y}}^{\mathsf{iseq}}$. □

The above result shows that if two processes generated by two step sequences (x, y) of

an elementary net with inhibitor arcs are equivalent, then the two interval processes generated by the interval representation of the two step sequences $(\overline{x}, \overline{y})$ are equivalent as well.

Theorem 8. *For every firing step sequence x, we have*

$$\mathsf{strat}^{proc}(\mathsf{P}_x^{\mathsf{step}}) = \mathsf{interv}^{proc}(\mathcal{P}_{\overline{x}}^{\mathsf{iseq}}) \cap \mathtt{strat}(\mathcal{E}_n).$$

□

Proof. Let $< \in \mathsf{strat}^{proc}(\mathsf{P}_x^{\mathsf{seq}})$. This means there is a step sequence $y = B_1 \dots B_r$ such that $< = \lhd_y$, where $\lhd_y$ is a stratified order defined by y, and $\mathsf{P}_x^{\mathsf{step}} = \mathsf{P}_y^{\mathsf{step}}$. By Lemma 3, $\mathcal{P}_{\overline{x}}^{\mathsf{iseq}} = \mathcal{P}_{\overline{y}}^{\mathsf{iseq}}$, so $<_{\overline{x}} \in \mathsf{interv}^{proc}(\mathcal{P}_{\overline{x}}^{\mathsf{iseq}})$, and by Proposition 2, $\blacktriangleleft_{\overline{x}} = \lhd_y$ (where $\blacktriangleleft_{\overline{x}}$ is an interval order defined by the sequence $\overline{x}$), and $\blacktriangleleft_{\overline{x}} \in \mathtt{strat}(\mathcal{E}_n)$, so $\mathsf{strat}^{proc}(\mathsf{P}_x^{\mathsf{step}}) \subseteq \mathsf{interv}^{proc}(\mathcal{P}_{\overline{x}}^{\mathsf{iseq}}) \cap \mathtt{strat}(\mathcal{E}_n)$.

Let $< \in \mathit{interv}(\mathcal{P}_{\overline{x}}) \cap \mathtt{strat}(\mathcal{E}_n)$. This means there is an interval sequence z such that $\lhd_z$ represents $<$ via Theorem 1 and $\mathcal{P}_z = \mathcal{P}_{\overline{x}}$. Since $<$ is a stratified order, by Proposition 2 there is a set of steps $B_1, \dots, B_r$ such that $z = z_1 \dots z_r$ and $z_i \in B_i^{BE}$, $i = 1, \dots, r$. Define $z' = B_1 \dots B_r$. By Proposition 2 again, $< = <_{z'}$, where $\lhd_{z'}$ is the stratified order defined by z'. Clearly $\overline{z'} = z$. By Lemma 3(2), $\mathsf{P}_x^{\mathsf{seq}} = P_{z'}$, so $\lhd_{z'} \in \mathit{strat}(\mathsf{P}_x^{\mathsf{seq}})$. Hence $\mathit{interv}(\mathcal{P}_{\overline{x}}) \cap \mathtt{strat}(\mathcal{E}_n) \subseteq \mathit{strat}(\mathsf{P}_x^{\mathsf{seq}})$. □

The last two results are partially illustrated by the far right part of Figure 8.4. For $x = \{a, c\}\{b\}$, we have $\mathit{ifs}(x) = \{BaBcEaEcBaBb,\ BaBcEcEaBaBb,\ BcBaEaEcBaBb,\ BcBaEcEaBaBb\}$, so $y = BaBcEaEcBaBb \in \mathit{ifs}(x)$.
We can show by inspection that $\mathit{interv}(\mathcal{P}_y^{\mathsf{iseq}}) = \{<_2^{\mathsf{N}}, <_3^{\mathsf{N}}, <_4^{\mathsf{N}}\}$ and $\mathit{interv}(\mathcal{P}_y^{\mathsf{iseq}}) \cap \mathtt{strat}(\{a^1, b^1, c^1\}) = \{<_2^{\mathsf{N}}, <_3^{\mathsf{N}}\}$. Moreover, using the results of Janicki and Koutny

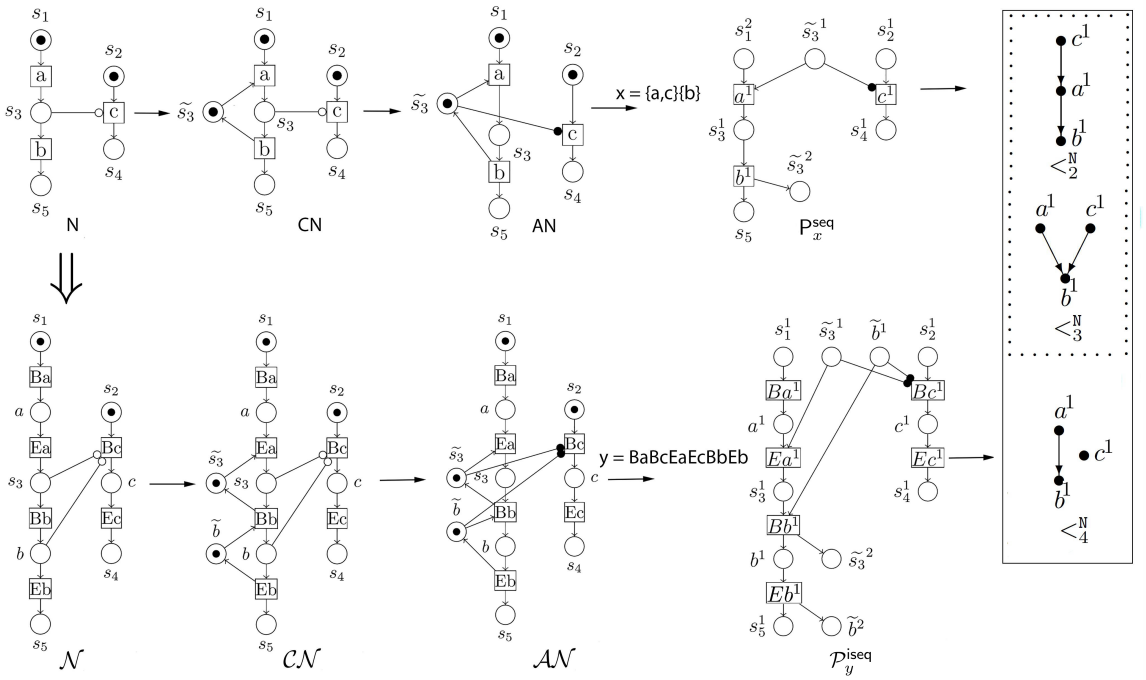


Figure 8.4: An example of an ENI, its interval representation, processes and concurrent histories they generate. The process $\mathsf{P}_x^{\mathsf{seq}}$ generates a concurrent history $\{<_2^{\mathsf{N}}, <_3^{\mathsf{N}}\}$ while the process $\mathcal{P}_y^{\mathsf{iseq}}$ generates $\{<_2^{\mathsf{N}}, <_3^{\mathsf{N}}, <_4^{\mathsf{N}}\}$.

(1995); Kleijn and Koutny (2004), we may show that $strat(\mathsf{P}_x^{\mathsf{seq}}) = \{<_2^{\mathsf{N}}, <_3^{\mathsf{N}}\}$, as required by Theorem 8.

Chapter 9

Interval Processes and Interval Traces

The interval processes of elementary nets with inhibitor arcs presented in our research are an extension and generalization of step sequence process semantics of elementary inhibitor Petri nets proposed in Janicki and Koutny (1995) and improved in Kleijn and Koutny (2004); while the interval traces are a generalization of classical Mazurkiewicz traces Diekert et al. (1995); Mazurkiewicz (1995). In this section, we aim to show that interval trace semantics is equivalent to the interval process semantics. The process semantics studied in this thesis, i.e. in the style of Kleijn and Koutny (2004); Nielsen et al. (1990), does not usually require complex validation as intuitively it is just a set of system unfoldings, so it is as natural as any operational semantics. Hence, the results of this section can also be interpreted as a validation of the interval traces semantics; another contribution of our research.

For the net $\mathcal{CN}$ of Figure 5.1 and $x = BaBcEaEcBbEb$, the content of $[x]_{ind_{\mathcal{CN}}}$ consists of ten sequences analyzed in Section 7.3.2 and $\prec^{trace}_{[x]}$ is the same as $\prec^{proc}_{x}$ from

Figures 8.2. Moreover $\mathsf{interv}^{trace}([x]_{ind_{\mathcal{CN}}}) = \mathsf{interv}^{ord}(\lessdot_x^{proc}) = \mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}}) = \{<_2^N, <_3^N, <_4^N\}$ (see Figure 8.2).

Following, we will formally show that this kind of relationship holds in all cases.

Theorem 9 (Equivalence of Process and Interval Traces Semantics). *Let* $\mathtt{N} = (P, T, F, I, C_{init})$ *be an ENI system,* $\mathcal{CN} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{I}, C_{init})$ *be its complement closed interval representation and let* $x = \alpha_1 \dots \alpha_n$ *be an interval firing sequence of* $\mathcal{CN}$*. The the following equations hold.*

1. $\lessdot^{trace}_{[x]_{ind_{\mathcal{CN}}}} \;=\; \lessdot^{proc}_x$
2. $\mathsf{interv}^{trace}([x]_{ind_{\mathcal{CN}}}) = \mathsf{interv}^{ord}(\lessdot_x^{proc}) = \mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}})$ □

Proof. (1) For every $x \in \mathcal{T}^*$, let $\widehat{\mathcal{T}}_x$ is the set of all enumerated transitions from which $\widehat{x}$ is built. For example for $x = BaBbEbEaBaEa$ we have
$\widehat{x} = Ba^1Bb^1Eb^1Ea^1Ba^2Ea^2$, and $\widehat{\mathcal{T}}_x = \{Ba^1, Ea^1, Ba^2, Ea^2, Bb^1, Eb^1\}$.
Define the relation $ind^{\widehat{x}}_{\mathcal{CN}} \subseteq \widehat{\mathcal{T}}_x \times \widehat{\mathcal{T}}_x$, as follows:

$$(\alpha^i, \beta^j) \in ind^{\widehat{x}}_{\mathcal{CN}} \iff (\alpha, \beta) \in ind_{\mathcal{CN}}.$$

Note that $(\widehat{\mathcal{T}}_x, ind^{\widehat{x}}_{\mathcal{CN}})$ and $(\widehat{\mathcal{T}}_x, \frown_{\lessdot_x^{proc}})$ are proper traces alphabets. Moreover from Definition 31 and Algorithm 3 applied to $\mathcal{CN}$, we have

$$(\alpha^i, \beta^j) \in ind^{\widehat{x}}_{\mathcal{CN}} \iff \alpha^i \frown_{\lessdot_x^{proc}} \beta^j.$$

Hence for any y which is an interval firing sequence of $\mathcal{CN}$,

$$x \equiv_{ind_{\mathcal{CN}}} y \iff \widehat{x} \equiv_{\frown_{\lessdot_x^{proc}}} \widehat{y},$$

i.e. $y \in [x]_{ind_{\mathcal{CN}}} \iff \widehat{y} \in [\widehat{x}]_{\frown_{\lessdot_x^{proc}}}$. Now we have:

$$\mathsf{total}(\lessdot_x^{proc}) = \{\lhd_y \mid \widehat{y} \in [\widehat{x}]_{\frown_{\lessdot_x^{proc}}}\} = \{\lhd_y \mid y \in [x]_{ind_{\mathcal{CN}}}\} = \mathsf{total}(\lessdot^{trace}_{[x]_{ind_{\mathcal{CN}}}}),$$

which means $\lessdot^{trace}_{[x]_{ind_{\mathcal{CN}}}} = \lessdot_x^{proc}$.

(2) Recall that $\mathsf{interv}^{ord}(\lessdot_x^{proc}) = \{\blacktriangleleft_y | \lhd_y \in \mathsf{total}(\lessdot_x^{proc})\}$ and

$\mathsf{interv}^{trace}([x]_{ind_{\mathcal{CN}}}) = \{\blacktriangleleft_t | \ t \in [x]_{ind_{\mathcal{CN}}}\}$

By Theorem 6 we already have

$\mathsf{interv}^{ord}(\lessdot_x^{proc}) = \mathsf{interv}^{proc}(\mathcal{P}_x^{\mathsf{iseq}})$. Let $\blacktriangleleft \in \mathsf{interv}^{trace}([x]_{ind_{\mathcal{CN}}})$, i.e. $\blacktriangleleft = \blacktriangleleft_t$ for some $t \in [x]_{ind_{\mathcal{CN}}}$, so $\lhd_t \in \mathsf{total}(\lessdot_x^{proc})$, i.e. $\blacktriangleleft_t \in \mathsf{interv}^{ord}(\lessdot_x^{proc})$. Now let $\blacktriangleleft \in \mathsf{interv}^{ord}(\lessdot_x^{proc})$. Hence $\blacktriangleleft = \blacktriangleleft_t$ for some t such that $\lhd_t \in \mathsf{total}(\lessdot_x^{proc}) = \mathsf{total}(\lessdot^{trace}_{[x]_{ind_{\mathcal{CN}}}})$. But this means that $t \in [x]_{ind_{\mathcal{CN}}}$, i.e. $\blacktriangleleft_t \in \mathsf{interv}^{trace}([x]_{ind_{\mathcal{CN}}})$. □

The above theorem is an equivalent of similar seminal results for step sequences (i.e. stratified orders) operational semantics, comtraces and stratified orders process semantics of Janicki and Koutny (1995); Kleijn and Koutny (2004). In principle, it states that the interval process semantics and interval traces semantics are equivalent for elementary inhibitor nets.

Chapter 10

Interval Semantic for Other Petri Nets

The notion of interval runs has been applied successfully in Pelz et al. (2015); Pelz and Kabouche (2016) for Timed Petri nets and Chatain et al. (2015) for Contextual nets.

First, the idea of treating transitions as intervals rather than atomic actions have been applied to Timed Petri Nets (TPN) resulting in the introduction of Interval Timed Petri Nets (ITPN). ITPN was introduced and interval calculus for it was provided in Pelz et al. (2015). Then, a process semantics for ITPN was defined in Pelz and Kabouche (2016). Similarly to our interval representation of ENI, a transition of ITPN is divided into a beginning and end (called sartfire and endfire in ITPN). However, the concept of time that a transition takes (from starting to end) is different between the two approaches as in ITPN, clearly, considering time is essential while in our approach, time is disregarded. The time a transition takes in ITPN is given by an interval (thus duration is not fixed) rather than a fixed number showing

a transition's duration in classical TPN.

In Chatain et al. (2015), similar ideas to the ones presented here are applied to Contextual nets (i.e. net with contextual or activator arcs which is orthogonal to the inhibitor arcs used in our model).

In this chapter, a summary of those papers is presented for the purpose of comparing them to our approach.

10.1 Interval-Timed Petri Nets

The initial goals of Petri nets was to model concurrent systems focusing only on the causal relation between activities in such system. The rational of such a goal was that other factor like *time* is not important for a wide spectrum of systems. However, in real systems, time is of a great importance and cannot be neglected. As a result, a number of new classes of Petri nets taken time into consideration have been developed (Heiner and Popova-Zeugmann (1997); Ramchandani (1974); Popova-Zeugmann (2014); Zuberek (1980) are a few examples). Enriching classical Petri nets with time increases the modeling power of the nets such that almost all time-dependent Petri nets models are Turing equivalent.

Interval-Timed Petri nets (ITPN) extends Timed Petri nets (TPN) Ramchandani (1974); Sifakis (1980) by allowing the firing duration of a transition (which is given by a fixed natural number in TPN) to vary with an interval associated with the transition.

Definition 35 (Pelz et al. (2015)). *Let N be an elementary net (Definition 5) and $D : T \to \mathbb{N} \times \mathbb{N}$ be a function. Then the pair $Z = (N, D)$ is* ***called an Interval-Timed Petri net (ITPN)*** *where N is its the skeleton and D is its duration function*

including zero duration. □

The net in Figure 10.1 is an example of a simple ITPN (taken from Pelz and Kabouche (2016)).

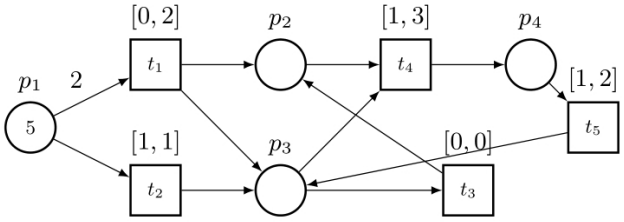


Figure 10.1: An interval-timed Petri net (ITPN).

The function D defines an interval for each transition within which its firing duration can vary. The bounds $sfd(t)$ and $lfd(t)$ with $D(t) = \Big(sfd(t), lfd(t)\Big)$ are called the shortest firing duration for t and the longest firing duration for t, respectively and are allowed to be zero (i.e. the firing can be considered to take no time). Additionally, $\sigma_i \in \Big(D(t_i) \cap \mathbb{N}\Big)$ can be the actual duration of transition t_i.

The behaviour of ITPN is defined as maximal steps with (enforced) auto-concurrency. This means that when transitions are enabled they must start firing even when another instance of the same transition is being executed. Thus a maximal step will be a multiset of events which occur at the same moment. Formally, a multiset U of events E is a total function $U : E \to \mathbb{N}$, where $U(e_i)$ denotes how many times event e_i occurs in U.

A token arrives to the output place of a transition t_i only after the time corresponding to the actual duration of t_i has passed. Such a duration is not known at the beginning of the firing of t_i which can stop after an arbitrary number $\sigma_i \in D(t_i)$ of ticks has elapsed.

Unlike the configurations of other Petri nets which are given by places, configurations of ITPN uses the notion of state which includes both the marking (subset of places) and the corresponding temporal information. That is, a state in ITPNs are pairs $S = (M, h)$ where M being the subset of places and h codes the clocks of the transitions in the form of matrix of dimension ($\mid T \mid \times d$). Thus the clock-matrix h has as many rows as transition of the skeleton N and $(\max_{t_i \in T}(lfd(t_i)) + 1)$ columns. The value $h_{i,j+1}$ represents the number of active transitions t_i with age j (i.e. fired since j times ticks) where $j \in D(t_i)$.

The initial state $S^{(0)} = (M^{(0)}, h^{(0)})$ of Z is given by the initial marking of the skeleton and the zero-clock-matrix where $h^{(0)} = 0$ for all i, j. Additionally, the firing rules of ITPN distinguishes between three different types of event:

1. *Startfire* event: denoted as $[t_i$ and must occur immediately (even n times) if t_i becomes enable in the skeleton. For each occurrence of $[t_i$, tokens from t_i's input places are removed, the clock associated with t_i counts this occurrence by increasing the number $h_{i,1}$ and t_i is called active.

2. *Endfire* event: denoted as $t_i\rangle$ and must occur (even n times) if the clock associated with t_i is expiring (i.e. $h_{i,j+1} = n \neq 0$ and $j = lfd(t_i)$). For each occurring endfire event, the corresponding $h_{i,j+1}$ decreases and the output places of t_i is populated with tokens.

3. *Tick* events: denoted as ✓and is enabled if and only if there is no firing event that must either start firing or end firing. A tick event occurrence increments the clocks for all active transitions (i.e. it is global).

An ITPN changes from an (after-tick) state to another one by the occurrence of

Globalstep which consists of *Endstep* (a multiset of endfire events), and *Iteratedstep* (an iterative union of two multisets *Maxstep* and *EndstepZero*). A Maxstep is a maximal step of startfire events and an EndstepZero is a multiset of endfire events of transitions with zero firing duration. The iteration stops when no further Maxstep exists. The formalization of those concepts are rather complex and readers are referred to Pelz et al. (2015) for more. Ultimately, a firing step sequence of an ITPN is an alternating sequence of Globalstep and ticks. Formally, a firing sequence σ is given by

$$\sigma = S_0 \overset{globalstep_1}{\longrightarrow} \tilde{S}_0 \overset{\checkmark}{\longrightarrow} S_1 \overset{globalstep_2}{\longrightarrow} \tilde{S}_1 \overset{\checkmark}{\longrightarrow} S_2 \ldots S_{n-1} \overset{globalstep_n}{\longrightarrow} \tilde{S}_{n-1} \overset{\checkmark}{\longrightarrow} S_n.$$

A possible firing step for the ITPN on Figure 10.1 is

$$(\{\}, \{[t_1^2, [t_2, t_1\rangle, [t_3, t_3\rangle\}), \checkmark \equiv (\{[t_1^2, [t_2, t_1\rangle, [t_3, t_3\rangle\}, \checkmark).$$

The set of all after-tick states and intermediate states are used to build the reachability graph which tends to grow very quickly. Such a quick growth makes the consideration of the state equation to decide unreachability a better alternative.

In Pelz and Kabouche (2016) a process semantics has been defined for ITPN. Formally a process of ITPN is a pair (N, ϕ) where N' is an occurrence net (Definition 17) and ϕ is a homomorphism which labels the CN with information from the ITPN. The set of clock labels CL is introduced to capture information about time passed since a transition has start firing.

$$CL = \{(t_i, j) \mid t_i \in T \text{ and } j \leq lfd(t_i)\}.$$

A clock label (t_i, j) shows that t_i has started firing and has age j. Also, the set of firing event $\mathcal{FE}$ is defined as

$$\mathcal{FE} = \{[t_i \mid t_i \in T\} \cup \{t_i\rangle \mid t_i \in T\} \cup \{\checkmark\}.$$

The net $\phi = (N', \phi)$ is a process of $\mathcal{N}$ if $\phi : B \cup E \to (P \cup CL) \cup \mathcal{FE}$ is a homomorphism satisfying:

1. $\forall b \in B, \phi(b)\ in P \cup CL$ and $\forall e \in E. \phi(e) \in \mathcal{FE}$.

2. ${}^\bullet\pi \subseteq B$ and $\forall p \in P, \mid \phi(e)^{-1}(p) \cap {}^\bullet\Pi \mid = M_0(p)$.

3. For each event e of the occurrence net N' one of the following cases holds

   - Case1: if $\phi(e) = [t_i$ then

     a) $\forall b \in {}^\bullet e, \phi(b) \in P$,

     b) $\forall p \in P, \mid \phi^{-1}(p) \cap {}^\bullet e \mid = v(p, t_i)$ and $\phi(e^\bullet) = \{(t_i, 0)\}$ and

     c) ${}^\bullet e$ is an antichain.

   - Case2: If $\phi(e) = t_i\rangle$ then

     a) $\phi({}^\bullet e) = \{(t_i, j)\}$ for some $j \in [sfd(t_i), lfd(t_i)]$ and

     b) $\forall b \in e^\bullet, \phi(b) \in P$‘and $\forall p \in P, \mid \phi^{-1}(p) \cap e^\bullet \mid = v(t_i, p)$

   - Case3: If $\phi(e) = \checkmark$ then

     a) $\forall b \in {}^\bullet e \cup e^\bullet, \phi(b) \in CL$,

     b) $\forall b\ in^\bullet e, (\phi(b) = (t_i, j)) \wedge (j < lfd(t_i))$,

     c) $\forall t_i \in T, \forall j \in [0, d], \mid \phi^{-1}(t_i, j) \cap {}^\bullet e \mid = \mid \phi^{-1}(t_i, j+1) \cap e^\bullet \mid$ and

     d) $\forall b \notin {}^\bullet e, (\phi(b) \in CL \Rightarrow \exists b'\ in^\bullet e, b \preceq b' \vee b' \preceq b)$.

The process in Figure 10.2 (from Pelz and Kabouche (2016)) illustrates this construction.

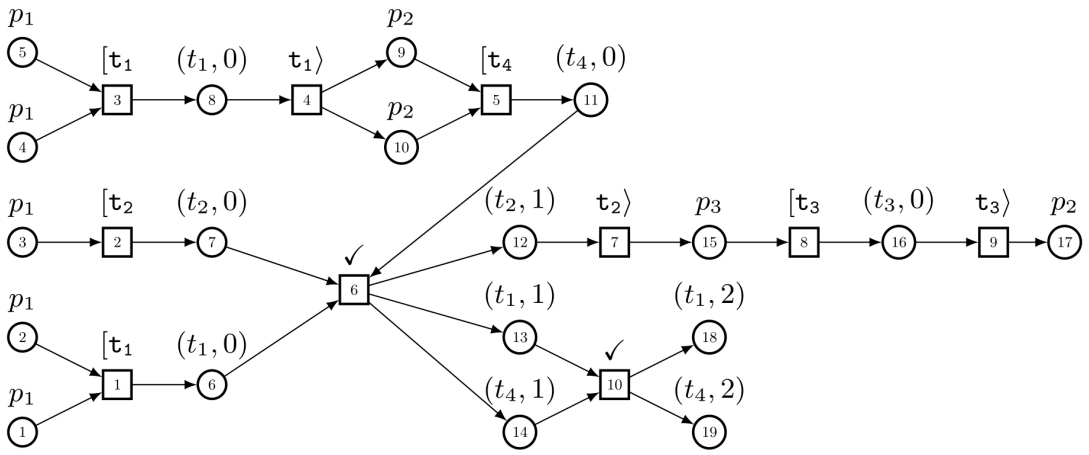


Figure 10.2: An initial part of an arbitrary process of the ITPN of Figure 10.1

Again, there are similarities rather at a higher level between interval elementary nets with inhibitor arcs and the interval timed Petri nets. In both cases, transitions are divided and are assumed to be allowed to take time which is more important in ITPN.

10.2 Interval Semantic for Activator Nets

Interval semantics have been defined for activator nets in Chatain et al. (2015). The general idea of such semantics is that a transition is split into two phases: t^- (checking phase of a transition t) and t^+ (firing phase for t). Then the step semantics of activator nets (defined in Section 4.3 can be redefined such that every step consists of any permutation of the actions of type t^- followed by any permutation of the actions t^+. The explicit splitting of transitions on the net makes the idea of two phases firing clearer as illustrated in Figure 10.3.

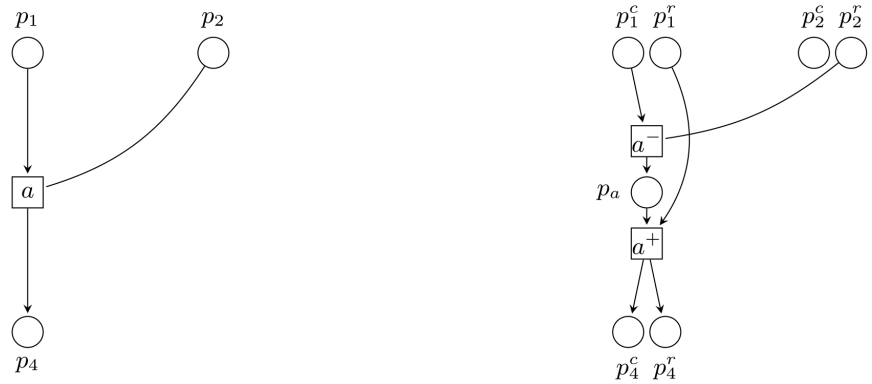


Figure 10.3: The splitting of transition a (left) into a^- and a^- (right).

The spiting of transitions is formalized as follows (see Chatain et al. (2015)).

Definition 36 (Chatain et al. (2015)). *For every activator net $N = (P, T, F, A, C_{initi})$, an activator net split $split(N)$ is defined as follows:*

1. *T' contains two copies, denoted t^- and t^+ of every transition $t \in T$.*
2. *P' contains two copies, denoted p^c and p^r of every place $p \in P$, plus one place p_t per transition $t \in T$.*
3. $^\bullet t^- = \{p^c \mid p \in {}^\bullet t\}$
4. $^\blacklozenge t^- = \{p^r \mid p \in {}^\blacklozenge t\}$
5. $t^{-\bullet} = \{p_t\}$
6. $^\bullet t^+ = \{p^r \mid p \in {}^\bullet t\} \cup \{p_t\}$
7. $^\blacklozenge t^+ = \emptyset$
8. $t^{+\bullet} = \{p^c \mid p \in t^\bullet\} \cup \{p^r \mid p \in t^\bullet\}$
9. $C'_{init} = \{p^c \mid p \in C_{init}\} \cup \{p^r \mid C_{init}\}$. □

The construction of $split(N)$ allows for a more general semantics that captures cases where both standard firing sequences and step sequences cannot. This is defined as interval semantics.

Definition 37 (Chatain et al. (2015)). *Every firing sequence of $split(N)$ is called i-run of N or run of N under the interval semantics. An i-run is complete if every t^- is matched by a t^+.* □

Given an i-run, a process can be constructed for the $split(N)$ in the standard way where processes are just unfolding of the original net.

Chapter 11

Conclusion

In this thesis, we have provided both an interval order operational semantics and an interval process semantics for elementary Petri nets with inhibitor arcs. Then, from the interval process semantics we derived an interval 'true concurrency' semantics in terms of interval order structures. Interval order or rather observing transitions of a concurrent system under the assumption that they are interval is often regarded as the most general and precise way when it comes to recording observations of systems. This fact has been argued from a philosophical perspective in Wiener (1914) and formally proved in Janicki and Koutny (1993).

Our approach started with transforming a given net with inhibitor arcs $\mathtt{N}$ into another net $\mathcal{N}$ that is called the interval representation of $\mathtt{N}$. This is done by splitting every transition of the original net $\mathtt{N}$ into a beginning and an end of the transition with a place in between (i.e. in the form $\boxed{Bt} \rightarrow \textcircled{t} \rightarrow \boxed{Et}$). Such a transformation is trivial in the case of elementary nets. However, introducing inhibitor arcs presents a challenge. In $\mathtt{N}$ a transition t that is connected to a place p by an inhibitor arc is not enabled unless the place is not marked (i.e. has no token), therefore, the token

has to be consumed by another transition t'. However, "during" the firing of t', the inhibitor arcs continues inhibiting t. This assumption is subtle in $\mathtt{N}$, however, in $\mathcal{N}$ it is of great importance. The idea is that, an inhibitor arc in $\mathtt{N}$ is represented by two inhibitor arcs in $\mathcal{N}$ to address the aforementioned subtlety. This ensures that an inhibited transition will not begin until the transition consuming the token from inhibitor place ends, provided that the inhibited transition has not already started. In $\mathcal{N}$, this allows for occurrences of transition to overlap while disallowing undefined behaviours of $\mathtt{N}$.

We formally defined the construction of $\mathcal{N}$ (given $\mathtt{N}$) and its operational semantics in terms of interval firing sequences. Then we assumed that all behavioral properties of $\mathtt{N}$ are defined by appropriate behavioral properties of $\mathcal{N}$ before we formally demonstrated that firing a step sequence in $\mathtt{N}$ is properly simulated by firing an appropriate interval sequence in $\mathcal{N}$. Moreover, while $\mathcal{N}$ defines behaviours that cannot be defined by $\mathtt{N}$, it does *not* generate any new behaviour that can be described by step sequences of $\mathtt{N}$.

Although it is possible to derive interval processes directly from $\mathtt{N}$, without using $\mathcal{N}$ or explicit complementary places, some intuition is then lost so we do not explore this issue here.

In addition, we define process semantics of $\mathcal{N}$ which can be seen as an extension and generalization of the already well developed step sequence process semantics of elementary nets with inhibitor arcs Janicki and Koutny (1995); Kleijn and Koutny (2004). Generally speaking, a process is an unfloding of the original net along some predefined legal behaviours. In case of $\mathcal{N}$, activator occurrence nets generated by interval firing sequences are used to defined processes. Then, we demonstrated how

an interval order structure that characterizes $\mathtt{N}$ is derived from a given interval process of $\mathcal{N}$. Since the process type semantics is based on the concept of nets unfolding, it is very natural and usually does not require complex justification. Therefore, the results presented in this thesis can also be interpreted as some validation of the interval order semantics for inhibitor nets.

This idea of validating other domains of interval semantics is applied to interval trace semantics Janicki et al. (2012); Janicki and Yin (2015). We applied the interval traces semantics to the interval representation of elementary nets with inhibitor arcs, and showed that the interval process semantics proposed in this thesis and the interval traces semantics are equivalent.

Moreover, we demonstrated that when operational semantics is restricted to step sequences, or stratified orders, our model produces the same results as that of Janicki and Koutny (1995); Kleijn and Koutny (2004).

Last but not least, it is important to point out that there are concurrent systems that, when modeled as Petri nets with inhibitor arcs, produce *only* pure interval behaviours, i.e. they generate neither firing sequences nor firing step sequences, only interval firing sequences. The net $\mathtt{N}_{io}$ from Figure 1.1 is one of such nets. If observations are only represented as step sequences, then $\mathtt{N}_{io}$ generates no behaviour at all. However, if runs are represented as interval orders, then it generates an observation (system run) that is exactly the interval order $<_4^{\mathtt{N}}$. This cannot be modeled by standard semantics and behaviours of such nets can only be analyzed using our model. Concurrent systems are known for generating extremely complex behaviours, so there is a need for tools that can adequately model all of them, even if some do not occur that often.

Our research can be extended in a few direction. For instance, an extension of interval order semantics to general Place/Transition Petri nets with inhibitor arcs (a higher class of Petri nets) is a serious future research project. Another direction is to apply the interval semantics to other extensions of Petri nets. For example, our approach can be applied to activator nets (Section 4.3) or to interval-timed Petri nets (Section 10.1).

Bibliography

Aalbersberg, I. J. and G. Rozenberg (1988). Theory of traces. *Theoretical Computer Science 60*(1), 1 – 82.

Abraham, U., S. Ben-david, and M. Magidor (1990). On global-time and inter-process communication. In *Semantics for Concurrency*, pp. 311–323. Springer-Verlag.

Agerwala, T. and M. Flynn (1973). Comments on capabilities, limitations and "correctness" of petri nets. *SIGARCH Comput. Archit. News 2*(4), 81–86.

Alqarni, M. and R. Janicki (2015). On interval process semantics of petri nets with inhibitor arcs. In *Application and Theory of Petri Nets and Concurrency*, pp. 77–97. Springer.

Alqarni, M. and R. Janicki (2016a). Interval process semantics of petri nets with inhibitor arcs. *Theoretical Computer Science*. submitted.

Alqarni, M. and R. Janicki (2016b). On modeling inhibitor nets with interval processes and interval traces. In *The 2015 International Conference on Foundations of Computer Science*, pp. 3–9.

Baldan, P., N. Busi, A. Corradini, and G. M. Pinna (2004). Domain and event structure semantics for petri nets with read and inhibitor arcs. *Theoretical Computer Science 323*, 129–189.

Busi, N. and G. M. Pinna (1999). Process semantics for place/transition nets with inhibitor and read arcs. *Fundamenta Informaticae 40*((2, 3)), 156–197.

Chatain, T., S. Haar, M. Koutny, and S. Schwoon (2015). Non-atomic transition firing in contextual nets. In R. Devillers and A. Valmari (Eds.), *Application and Theory of Petri Nets and Concurrency*, Volume 9115 of *Lecture Notes in Computer Science*, pp. 117–136. Springer International Publishing.

Chiola, G., S. Donatelli, and G. Francheschinis (1991). Priorities, inhibitor arcs and concurrency in p/t-nets. In *Proc. of ATPN'91. (Applications and Theory of Petri Nets)*, pp. 182–205.

Chuang, L., Q. Yang, R. Fengyuan, and D. C. Marinescu (2002). Performance equivalent analysis of workflow systems based on stochastic petri net models. In *Engineering and Deployment of Cooperative Information Systems*, pp. 64–79. Springer.

Desel, J. and W. Reisig (1998). Place/transition petri nets. In *Lectures on Petri Nets I: Basic Models*, pp. 122–173. Springer.

Diekert, V., G. Rozenberg, and G. Rozenburg (1995). *The book of traces*, Volume 15. World Scientific.

Fishburn, P. C. (1970). Intransitive indifference with unequal indifference intervals. *Journal of Mathematical Psychology 7*, 144–149.

Fishburn, P. C. (1985). Interval graphs and interval orders. *Discrete Mathematics 55*(2), 135 – 149.

Gaifman, H. and V. Pratt (1987). Partial order models of concurrency and the computation of function. In *Proc. of LICS'87. (Logic in Computer Science)*, pp. 72–85.

Goltz, U. and W. Reisig (1983). The non-sequential behaviour of petri nets. *Information and Control 57*(2), 125–147.

Hack, M. (1979). *Decidability questions for Petri nets.* Garland.

Heiner, M. and L. Popova-Zeugmann (1997). *Worst Case Analysis of Concurrent Systems with Duration Interval Petri Nets.* Citeseer.

Hoodgeboom, H. J. and G. Rozenberg (1991). Diamond properties of elementary net systems. *Fundamenta Informaticae 14*(3), 287–300.

Janicki, R. (2008). Relational structures model of concurrency. *Acta Informatica 45*, 279–320.

Janicki, R., J. Kleijn, and M. Koutny (2010). Quotient monoids and concurrent behaviours. *Scientific Applications of Language Methods 1*, 313–385.

Janicki, R. and M. Koutny (1991). Invariants and paradigms of concurrency theory. In *Proc. of PARL'91. (Lecture Notes in Computer Science)*, Volume 506, pp. 59–74.

Janicki, R. and M. Koutny (1993). Structure of concurrency. *Theoretical Computer Science 112*, 5–52.

Janicki, R. and M. Koutny (1995). Semantics of inhibitor nets. *Information and Computation 112*, 5–52.

Janicki, R. and M. Koutny (1997). Fundamentals of modelling concurrency using discrete relational structures. *Acta Informatica 34*, 367–388.

Janicki, R. and D. T. M. Lê (2011). Modelling concurrency with comtraces and generalized comtraces. *Information and Computation 209*(11), 1355 – 1389.

Janicki, R. and X. Yin (2015). Modeling concurrency with interval orders. *Information and Computation*.

Janicki, R., X. Yin, and N. Zubkova (2012). Modeling interval order structures with partially commutative monoids. In M. Koutny and I. Ulidowski (Eds.), *CONCUR 2012 Concurrency Theory*, Volume 7454 of *Lecture Notes in Computer Science*, pp. 425–439. Springer Berlin Heidelberg.

Juhás, G., R. Lorenz, and S. Mauser (2007). Complete process semantics for inhibitor nets. In *Proc. of ICATPN'07. (Lecture Notes in Computer Science)*, Volume 4546, pp. 184–203.

Kleijn, H. C. M. and M. Koutny (2004). Process semantics of general inhibitor nets. *Information and Computation 190*, 18–69.

Kleijn, J. and M. Koutny (2008). Formal languages and concurrent behaviour. *Studies in Computational Intelligence 113*, 125–182.

Lamport, L. (1986). The mutual exclusion problem. *Journal of ACM 33*(2), 313–326.

Lê, D. T. M. (2010). A characterization of combined traces using labeled stratified order structures. In *Applications and Theory of Petri Nets: 31st International Conference, PETRI NETS 2010, Braga, Portugal, June 21-25, 2010. Proceedings*, pp. 104–124. Springer.

Marsan, M. A., G. Balbo, and G. Conte (1986). Performance models of multiprocessor systems.

Mazurkiewicz, A. (1977). Concurrent program schemes and their interpretations. *DAIMI Report Series 6*(78).

Mazurkiewicz, A. (1995). *The Book of Traces*, Chapter Introduction to Trace Theory, pp. 3–40. World Scientific.

Montanari, U. and F. Rossi (1995). Contextual nets. *Acta Informatica 32*(6), 545–596.

Murata, T. (1989). Petri nets: Properties, analysis and applications. In *Proc. of IEEE*, Volume 77, pp. 541–579.

Nielsen, M., G. Rozenberg, and P. S. Thiagarajan (1990). Behavioural notions for elementary net systems. *Distributed Computing 4*, 45–57.

Pelz, E. and A. Kabouche (2016). On processes and branching processes of itpns. *Draft*.

Pelz, E., A. Kabouche, and L. Popova-Zeugmann (2015). Interval-timed petri nets with auto-concurrent semantics and their state equation. In *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'15)*, pp. 245–265.

Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Petri, C. A. (1962). *Kommunikation mit Automaten*. Ph. D. thesis, Universitt Hamburg.

Petri, C. A. (1966). *Communication with automata*. Ph. D. thesis, Universitt Hamburg.

Petri, C. A. (1996). Nets, time and space. *Theoretical computer science 153*(1), 3–48.

Pomello, L., G. Rozenberg, and C. Simone (1992). A survey of equivalence notions for net based systems. In *Advances in Petri Nets 1992*, pp. 410–472. Springer.

Popova-Zeugmann, L. (2014). Time petri nets: theory, tools and applications. part 1.

Ramchandani, C. (1974). Analysis of asynchronous concurrent systems by timed petri nets.

Reisig, W. (2013). *Understanding Petri Nets*, Volume 4. Springer.

Rozenberg, G. and J. Engelfriet (1998). Elementary net systems. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets. (Lecture Notes in Computer Science)*, pp. 12–121.

Rozenberg, G. and P. Thiagarajan (1986). Petri nets: Basic notions, structure, behaviour. In J. de Bakker, W.-P. de Roever, and G. Rozenberg (Eds.), *Current Trends in Concurrency*, Volume 224 of *Lecture Notes in Computer Science*, pp. 585–668. Springer Berlin Heidelberg.

Sifakis, J. (1980). Use of petri nets for performance evaluation. *Acta Cybernetica 4*(1978), 185–202.

Szpilrajn, E. (1930). Sur l'extension de l'ordre partiel. *Fundam. Mathematicae 16*, 386–389.

Tsai, J., S. Jennhwa Yang, and C. Yao-Hsiung (1995). Timing constraint petri nets and their application to schedulability analysis of real-time system specifications. *Software Engineering, IEEE Transactions on 21*(1), 32–49.

Van Der Aalst, W. and K. M. Van Hee (2004). *Workflow management: models, methods, and systems*. MIT press.

Venkatesh, K., M. Zhou, and R. J. Caudill (1994). Comparing ladder logic diagrams and petri nets for sequence controller design through a discrete manufacturing system. *Industrial Electronics, IEEE Transactions on 41*(6), 611–619.

Vogler, W., A. Semenov, and A. Yakovlev (1998). Unfolding and finite prefix for nets with read arcs. In *Proc. of CONCUR'98. (Lecture Notes in Computer Science)*, Volume 1466, pp. 501–516.

Wiener, N. (1914). A contribution to the theory of relative position. In *Proc. of the Cambridge Philosophical Society*, Volume 17, pp. 441–449.

Winkowski, J. (1998). Process of contextual nets and their characteristics. *Fundamenta Informaticae 33*, 1–31.

Zuberek, W. M. (1980). Timed petri nets and preliminary performance evaluation. In *Proc. of the 7th Annual Symp. on Computer Architecture*, pp. 89–96.