A METRIC INTERVAL-BASED TEMPORAL DESCRIPTION LOGIC

A METRIC INTERVAL-BASED TEMPORAL DESCRIPTION LOGIC

By

MORTEZA YOUSEF SANATI, B.Sc., M.Sc.

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

McMaster University

DOCTOR OF PHILOSOPHY (2015) MCMASTER UNIVERSITY
(Computing & Software) Hamilton, Ontario

TITLE: **A Metric Interval-based Temporal Description Logic**

AUTHOR: Morteza Yousef Sanati
B.Sc., M.Sc.

SUPERVISOR: Prof. Thomas S.E. Maibaum

CO-SUPERVISOR: Prof. Wendy MacCaull

NUMBER OF PAGES: xvi, 234

To my dear wife, Zahra, and my beloved mother for her support

Abstract

Because of the importance of undecidability and the concern with the high complexity of automated reasoning, a few interval-based temporal description logics (ITDLs) have been designed. Moreover, most existing ITDLs are not able to specify the lengths of intervals. In other words, they are not metric. On the other hand, some domains (e.g., medicine) are inherently interval-based, and require a metric logic in order to formalize defined processes and to check process consistency. Hence, a metric interval-based temporal description logic is required. In this thesis, we introduce such a logic (MITDL) along with two algorithms for the satisfiability checking of its formulas.

We first introduce an interval-based temporal logic, called IMPNL, inspired by Metric Propositional Neighbourhood Logic. We also present a sound, complete and terminating tableau-based algorithm for checking the satisfiability of IMPNL formulas. Afterwards, we combine a restricted version of IMPNL (IMPNL without a negation operator) with the $\mathcal{ALC}$ description logic to form a MITDL. We propose two tableau-based algorithms for checking the satisfiability of MITDL formulas. We show and prove they are sound, complete and terminate. These algorithms have PSpace and 2NExp-Time complexities. As a proof of concept, we use IMPNL and MITDL to model some clinical practice guidelines (CPG) and check their consistency. We compare MITDL with several languages commonly used for modeling CPGs.

List of Acronyms

ABox	Assertional Box
ACTL	A temporal logic
AIDS	Acquired Immunodeficiency Syndrome
AL	Attributive description logic Language
$\mathcal{ALC}$	Attributive Concept Language
$\mathcal{ALC}(\mathbf{D})$	Attributive Concept Language with Concrete domain
$\mathcal{ALCF}$	A description logic
$\mathcal{ALCN}$	A description logic
$\mathcal{ALCO}$	A description logic
Asbru	A clinical modeling language
BCDT$^+$	Branching CDT
BNF	Backus-Naur Form
CD4	Cluster of Differentiation 4
CDT	Venema's interval-based temporal logic
Condor	A description logic reasoner
CPG	Clinical Practice Guideline
CPT-4	A medical ontology
CTL	Computation Tree Logic
DL	Description Logic

ECTL	A temporal logic
$\mathcal{EL}$	A basic description logic
$\mathcal{ELHIF}_{\mathcal{R}^+}$	A description logic
EON	An architecture for protocol-based decision support
ExpSpace	Exponential Space
ExpTime	Exponential Time
FACT++	A description logic reasoner
FF	Finder Function
$\mathcal{FL}_\mathbf{0}$	Frame-based description Logic
FOL	First Order Predicate Logic
GLIF	GuideLine specification method and Interchange Format
HeCaSe2	The name of a platform
Hermit	A description logic reasoner
HIV	Human Immunodeficiency Virus
HS	Halpern-Shoham
ICD-9-CM	A medical ontology
IMPNL	A metric interval-based temporal
ITDL	Interval-based Temporal Description Logic
ITL	Interval Temporal Logic
KIV	Karlsruhe Interactive Verifier
LOINC-3	A medical ontology
LN	The name of a function
LTL	Linear time Temporal Logic
MITDL	Metric Interval-based Temporal Description Logic
MITDL_SAT	The name of an algorithm
MPNL	Metric Propositional Neighbourhood Logic

MPNL$_l$	Metric Propositional Neighbourhood Logic
NExp-Time	Non-Deterministic Exponential Time
NNF	Negation Normal Form
Pellet	A description logic reasoner
$\mathcal{PITL}$	Propositional Interval Temporal Logic
PNL	Propositional Neighbourhood Logic
PRODIGY	A guideline modeling language for chronic disease management
PROforma	A formal knowledge representation language
PSpace	Polynomial Space
QPTL	Quantified Propositional Temporal Logic
SAT	The name of an algorithm
Sub	The name of a function
SubD	The name of a function
$\mathcal{SHIQ}$	A description logic
$\mathcal{SHOIN}(\mathcal{D})$	A description logic
SMV	Symbolic Model Verifier
SNOMED CT	Systematized Nomenclature of Medicine–Clinical Terms
SOS	Structural Operational Semantics
SPIN	A model checker
$\mathcal{T}$-$\mathcal{ALC}$	A temporal description logic
$\mathcal{TALC}$	A temporal description logic
TB	Tuberculosis
TBox	Terminological Box
TDL	Temporal Description Logic
TL	Temporal Logic

$\mathcal{TL}$-$\mathcal{ALCF}$	A temporal description logic
$\mathcal{TL}$-$\mathcal{SHION}(\mathcal{D})$	A temporal description logic
$\mathcal{TL}$-$\mathcal{F}$	A temporal description logic
$\mathcal{TLU}$-$\mathcal{FU}$	A temporal description logic
$\mathcal{TLF}_5^-$	A temporal description logic
TSB	Temporal Subset Blocking
UML	Unified Modeling Language
UMLS	The name of a medical ontology
Uruz	The name of an environment

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The aim of this thesis is to introduce a *metric interval-based temporal description logic*.

In computer science, an ontology is a formal representation of the knowledge of a domain in terms of the domain concepts and the relationships between them; e.g., SNOMED CT is a comprehensive clinical terminology [1] which includes: clinical findings, symptoms, diagnoses, procedures, body structures, organisms and other etiologies, substances, pharmaceuticals, devices and specimen [2]. Moreover, an ontology can be used to reason about the entities of the domain [3].

One important logical formalism for representing an ontology is Description Logics (DL). Description logic is a family of fragments of predicate logic which can be used to define the static aspects [1] of a domain. An important feature of the logic is decidability of reasoning; i.e., there is a sound, complete and terminating method for the reasoning in the logic which always returns

[1] In this thesis, we restrict the static aspects of a domain to the aspects which can be modeled using unary and binary predicates.

the answer in a finite time [4]. There are many different DLs which are constructed on top of the three basic logics, $\mathcal{AL}$, $\mathcal{FL}_0$, $\mathcal{EL}$ [5]. The basic logics are extended by various constructors, e.g., Nominals, Concrete domains [6], and there are some tools for the logics which provide many useful reasoning services, e.g., satisfiability checking, subsumption checking, equality checking, instance checking and consistency checking [7, 8].

To model the dynamic aspect of a domain, another kind of logic is needed. Since time is a crucial part of the dynamic aspect, a temporal logic (TL) is necessary in order to represent time constraints in the domain. Generally, two kinds of temporal logics exist, point-based [81] and interval-based [83]. In point-based TL, the main ontological element is a time point. Hence, this logic is mostly suitable to model durationless events. Sometimes a user needs to model events which have duration or to determine whether two events overlap. An interval-based TL is able to deal with these situations. In some interval-based TLs, i.e., metric interval-based TLs (e.g., MPNL_l), a user can bind the duration of an event to a specific amount of time. Generally, these logics are more expressive than non-metric versions of interval-based TLs.

Neither a DL alone nor a TL alone is sufficient to describe both dynamic and static aspects of a domain. A combined logic, a temporal description logic (TDL), is needed. Having chosen a domain, one should select the most suitable TDL in order to model the domain. For instance, in the domain of medicine, most activities occur over an interval and it is not possible to model them as occurring at a time point, e.g., infusion of blood serum; thus, an interval-based TDL is required. On the other hand, the execution duration of some activities is generally restricted to a specific amount of time, e.g., take Ibuprofen for 2 days; therefore, the TDL must allow a user to specify the duration of an activity. In other words, a metric version of a TDL is needed.

To the best of our knowledge, there is only one metric TDL, Schmiedel's formalism; this formalism has no negation [9], and no algorithm has been designed for reasoning (satisfiability and subsumption reasonings) in this formalism [10]. The aim of this research is to design a metric TDL along with a sound, complete and terminating algorithm for checking the satisfiability of the logic formulas. Satisfiability checking is an important reasoning service of a logic because if we find that a formula is not satisfiable, it indicates that the formula contain an inconsistency. In some domains (e.g., medicine), it is very important to find such information. We will explain the importance of satisfiability checking in the domain of medicine later.

In order to design MITDL, we have designed a temporal logic inspired by MPNL$_l$ [11, 12] as the temporal logic part of our metric TDL. MPNL$_l$, proposed by D. Bresolin et al. in 2010, is a metric interval-based temporal language which has two modal operators, *meet* and *met-by*. In terms of expressivity, MPNL$_l$ is expressive enough to model a metric form of all Allen's relations between intervals, with the exception of *during* [11]. MPNL$_l$ does not meet all the requirements for formalizing static aspects of the domain of medicine because it is a propositional logic and does not allow us to define a relation between the elements of the domain; e.g., we are not able to model that Kaletra has a contraindication with Rifampin. Having designed a logic inspired by MPNL$_l$, we have selected $\mathcal{ALC}$ to create a metric TDL with the goal of maintaining the decidability of reasoning (i.e., existence of an effective method for reasoning). We believe the resulting logic will be very useful in many domains, especially medicine. The domain of medicine is inherently interval-based and many of its events have specific durations. Therefore, our logic is intended to help medical experts to formally design clinical practice

guidelines (CPGs), and check their satisfiability before they are used by clinicians. If a CPG is not satisfiable, it indicates that the CPG contains an inconsistency (e.g., two prescribed medicines are contraindicated), and it should not be used for a patient because it may be harmful for the patient. Sometimes, two CPGs cannot be used together because there are some contraindications between medicines used in the CPGs. Experts can test whether it is possible to use more than one guideline simultaneously for a patient who has different diseases.

The main objective of the dissertation is to propose a metric interval-based temporal description logic (MITDL) which supports the decidable satisfiability checking of its formulas. MITDL is a combination of IMPNL and $\mathcal{ALC}$ and benefits a user in different ways by providing an ability to describe the concepts of a domain with respect to time intervals along with the relationship between them using roles. We provide decidable tableau-based algorithms for checking the satisfiability of IMPNL and MITDL formulas. We prove that the satisfiability reasoning in these logics is decidable by proving the soundness, completeness and termination of the algorithms.

The remaining chapters of this thesis are organized as follows.

- In chapter 2, an overview of description logic is provided. In this chapter, some basic definitions, different kinds of reasoning in description logic and the syntax of a specific description logic, called $\mathcal{ALC}$, are presented. Afterwards, a tableau algorithm for checking the satisfiability of an $\mathcal{ALC}$ formula is explained. Finally, the relationship between description logic and first order predicate logic is discussed.

- In chapter 3, preliminary material about temporal logic is provided. This

material includes a short discussion of both point-based and interval-based temporal logics, as well as the structure of time intervals and possible relations between intervals. Also, some temporal logics, $\mathcal{ITL}$, $\mathcal{HS}$, CDT, BCDT$^+$, and PNL, are briefly explained. Then, a tableau algorithm for PNL$^+$ is presented. The last section of this chapter is devoted to the explanation of an important interval-based temporal logic, called Metric Propositional Neighbourhood logic.

- In chapter 4, a temporal logic called IMPNL is introduced. After investigating the difference between IMPNL and MPNL, a tableau-based algorithm for checking the satisfiability of an IMPNL formula is proposed. Further, the soundness and completeness of the algorithm is proved. Moreover, some case studies are provided in order to demonstrate how this language can be used to model some medical guidelines.

- In chapter 5, an interval-based temporal description logic (called MITDL), a combination of $\mathcal{ALC}$ and a restricted version of IMPNL, is proposed. Then, a tableau-based algorithm for checking satisfiability of some MITDL formulas, called *Simple* formulas, is developed. Also, the termination and the PSpace complexity of the algorithm is proved. Next, the tableau-based algorithm is extended in order to be able to check the satisfiability of any MITDL formula. It is shown that the extended algorithm terminates and has 2ExpTime complexity. At the end, we review existing interval-based temporal description logics.

- In chapter 6, three clinical practice guidelines are modeled with IMPNL and MITDL, and the satisfiability of the guidelines is checked with the proposed tableau-based algorithm. Finally, MITDL is compared with

five existing languages used for the modeling of clinical practice guidelines.

- In chapter 7, the summary of the thesis is presented, and possible future work is discussed.

Chapter 2

Description Logic

One important family of logical formalism for representing an ontology is Description Logics (DLs). "Description Logics (DL) are a well-investigated family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way" [7]. A key feature of DLs is decidability of reasoning, as this property makes it possible to develop automated tools to support their use. For example, satisfiability (resp. subsumption) checking reasoning in a DL is decidable when there exists an effective method for determining satisfiability of a concept (resp. determining whether a concept is subsumed by another concept) in the logic. In terms of usefulness, description logics are employed in many areas such as software engineering, medicine, information systems and importantly, development of ontologies.

2.1 Basic Definitions

In description logic languages, users may define the important notions (classes, relations, objects) of the domain using *concepts, roles*, and *individuals*

[7]. A concept is a set of individuals, and a role is a binary relationship between individuals [8]. A non-atomic concept, i.e., a *concept description*, equivalently, a defined concept, is a concept built from *atomic concepts* (unary predicates) and *roles* (binary predicates) using concept constructions. In some description logic languages, it is possible to define $features$ which are partial functions. We will present an example of a *feature* later.

A DL-*signature* Σ of a description logic, is a tuple Σ=(**C, R, I,** $\top$**,** $\bot$) [13] which contains a set of concept names (**C**), a set of role names (**R**), a set of names of individuals (**I**) and two constant names, $\top$ (Top concept) and $\bot$ (Bottom concept); some description logics have a set of feature names (**F**) and a set of extra predicates (**P**) associated with concrete domains (see Page 15); thus their signature is of the form (**C, R**$\cup$**P, I, F,** $\top$**,** $\bot$). We can define DL-*Sentences* and DL-*Schemas* over a DL-*signature* Σ. A DL-*Schema* is a concept construction used to define a concept description, and a DL-*Sentence* is a description logic axiom that can be one of the following items.

1. $C \sqsubseteq D$ $C, D \in \mathbf{C}$;

   Intuition: C is subsumed by D. In other words, every individual of C is an individual of D; e.g., $Cat \sqsubseteq Animal$: every cat is an animal.

2. $C \equiv D$ $C, D \in \mathbf{C}$;

   Intuition: Every individual of C is an individual of D and vice versa; e.g., $Male \sqcup Female \equiv Child \sqcup Adult$: every male or female is a child or is an adult, and every child or adult is a male or is a female. We will explain the $\sqcup$ construction later.

3. $R \sqsubseteq S$ $R, S \in \mathbf{R}$;

   Intuition: Every individuals, which are related by R, are semantically related by S. In other words, R role (relation) is a fragment of S role

(relation); e.g., $isProperSubsetof \sqsubseteq isSubsetof$: If a set (say $Male$) is proper subset of another set (say $Human$), $Male$ is subset of $Human$.

4. C(i) $C \in \mathbf{C}$ and $i \in \mathbf{I}$;
   Intuition: i is an individual of C; e.g., $Adult(John)$: John is an adult.

5. $R(i_0,i_1)$ $R \in \mathbf{R}$ and $i_0, i_1 \in \mathbf{I}$.
   Intuition: i_0 and i_1 are related by R; e.g., $Married(John, Sue)$: John is married to Sue.

Description logic, like first order logic, has a model-theoretic semantics [8]. Thus, as usual in a Σ-*Model* of a description logic language, we need a non-empty set and an interpretation. Formally, a Σ-*Model* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set, called the domain of interpretation, and $.^{\mathcal{I}}$ is an interpretation function which assigns to every concept name $\mathcal{C}$, a set of domain elements, i.e., $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$; to every role name R, a set of pairs of domain elements, i.e., $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, to every individual name $i \in \mathbf{I}$, an element of the domain, i.e., $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$; to the *Top concept*, $\top$, all individuals in the domain, i.e., $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$; and to the *Bottom concept*, $\bot$, the empty set, i.e., $\bot^{\mathcal{I}} = \emptyset$. We can extend $.^{\mathcal{I}}$ to DL-axioms (see Table 2.1) as well as DL constructions. Before we explain the DL constructions, we define a notion required for defining the semantics of some constructions. Henceforth we abuse notation by saying a concept C (or role R) where we mean concept name C (or role name R). Also, note that a user can use the DL constructions only in a language equipped with the required constructors.

Definition 1 ([15])**.** Given an interpretation $\mathcal{I}$, a role R and an individual x, an $R^{\mathcal{I}}$-*Successor-of* x denotes any individual y such that $\langle x, y\rangle \in R^{\mathcal{I}}$.

- Concept Construction

Table 2.1: Description Logic Syntax and Semantics [8, 14, 15, 16]

Name	Syntax	Semantics
ABox Axiom		
Concept Assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role Assertion	$R(a, b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$
TBox Axiom		
Concept Inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept Definition	$C \equiv D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
Role Inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

The constructed concepts which can be used in the definition of a concept description are as follows.

- Concept: $\neg C$
  - Constructor name: Complement;
  - Description: This concept denotes the individuals in the domain which are not individuals of concept C;
  - Semantics: $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$;
  - Example: $\neg Female$;
  - Description of Example: This expression denotes the individuals who are not Female.
- Concept: $C \sqcap D$
  - Constructor name: Conjunction;
  - Description: This concept denotes the individuals which are simultaneously individuals of C and individuals of D;
  - Semantics: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
  - Example: $Female \sqcap Employee$;

- Description of Example: Employees who are female are determined by this expression.

⋇ Concept: $C \sqcup D$

- Constructor name: Disjunction;
- Description: This concept denotes the individuals which are either individuals of C or individuals of D;
- Semantics: $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- Example: $Female \sqcup Married$;
- Description of Example: All individuals who are either female or married are denoted by this expression.

⋇ Concept: $\forall R.C$

- Constructor name: Value Restriction (Universal Restriction [15]);
- Description: This concept denotes the individuals whose property is that all $R^{\mathcal{I}}$-*Successors-of* them are in $C^{\mathcal{I}}$;
- Semantics: $(\forall R.C)^{\mathcal{I}} = \{x|$ all $R^{\mathcal{I}}$-*Successors-of* x are in $C^{\mathcal{I}}\}$;
- Example: $\forall Managedby.Male$;
- Description of Example: Assume $Managedby$ is a role which denotes that an individual is managed by another individual. The expression determines the set of individuals whose managers are male.

⋇ Concept: $\exists R.\top$

- Constructor name: Limited Existential Restriction;
- Description: This concept denotes the individuals which have participated in a certain role as a first argument;

- Semantics: $(\exists R.\top)^{\mathcal{I}} = \{x | \langle x, y \rangle \in R^{\mathcal{I}}\}$;
- Example: $\exists Managedby.\top$;
- Description of Example: This expression determines the individuals who have a manager.

⋇ Concept: $\exists R.C$

- Constructor name: Full Existential Restriction;
- Description: This concept denotes the individuals who are related to an individual of concept C by a certain role R s.t. the individual of concept C is the second argument of the role;
- Semantics: $(\exists R.C)^{\mathcal{I}} = \{x |$ some $R^{\mathcal{I}}$-*Successor-of* x is in $C^{\mathcal{I}}\}$;
- Example: $\exists Managedby.Male$;
- Description of Example: This concept denotes the individuals which have at least one male manager.

Two constructions, *Value Restriction* and *Existential Restriction* can be used to restrict both the domain and also the range of a role. The expression $\exists Managedby.\top \sqsubseteq Male$ restricts the domain of the *Managedby* role to males, and the expression $\top \sqsubseteq \forall Managedby.Female$ confines the range of the *Managedby* role to females.

⋇ Concept: $\{a\}$

- Constructor name: Nominal;
- Description: This concept has exactly one individual in its interpretation [17], so is modelled as a singleton set [14]. For more information see [8];
- Semantics: $(\{a\})^{\mathcal{I}} = \{a^{\mathcal{I}}\}$;

- Example: $\{Iran\}$;
- Description of Example: The expression denotes a singleton set consisting only of $Iran$.

Note that in a language which supports the disjunction operator, a user can enumerate a set with its constituent nominals, e.g., $Countries \equiv \{France\} \sqcup \{Italy\} \sqcup \{Iran\}$.

⋇ Concept: $\leq n\ R\ C$

- Constructor name: At-most Qualifying Number Restriction;
- Description: This concept denotes the individuals which are related to at most n individuals of concept C via the role R;
- Semantics: $(\leq n\ R\ C)^{\mathcal{I}} = \{x|$ at most n $R^{\mathcal{I}}$-*Successors-of* x are in $C^{\mathcal{I}}\}$;
- Example: $\leq 5\ Manages\ Female$;
- Description of Example: This expression determines the individuals who manage at most 5 females.

⋇ Concept: $\geq n\ R\ C$

- Constructor name: At-least Qualifying Number Restriction;
- Description: This concept denotes the individuals which are related to at least n individuals of concept C via the role R;
- Semantics: $(\geq n\ R\ C)^{\mathcal{I}} = \{x|$ at least n $R^{\mathcal{I}}$-*Successors-of* x are in $C^{\mathcal{I}}\}$;
- Example: $\geq 5\ Manages\ Female$;
- Description of Example: This expression determines the individuals who manage at least 5 females.

⋇ Concept: $\leq n\ R$

- Constructor name: At-most Number Restriction;
- Description: This concept denotes the individuals which are related to at most n individuals via the role R;
- Semantics: $(\leq n\ R)^{\mathcal{I}} = \{x | \#\{(x,y) \in R^{\mathcal{I}}\ \} \leq n\}$;
- Example: $\leq 10\ Manages$;
- Description of Example: The individuals who manage at most 10 individuals are determined by this expression.

⁕ Concept: $\geq n\ R$

- Constructor name: At-least Number Restriction;
- Description: This concept denotes the individuals which are related to at least n individuals via the role R;
- Semantics: $(\geq n\ R)^{\mathcal{I}} = \{x | \#\{(x,y) \in R^{\mathcal{I}}\ \} \geq n\}$ [14];
- Example: $\geq 2\ Manages$;
- Description of Example: The individuals who manage at least 2 individuals are denoted by this expression.

• Role Construction

The following roles are used in the concept constructions (e.g., in the concept $\exists R.C$).

⁕ Role: R^{-}

- Constructor name: Inverse Role;
- Description: This role is the inverse of the role R.
- Semantics: $(R^{-})^{\mathcal{I}} = \{\langle y,x\rangle | \langle x,y\rangle \in R^{\mathcal{I}}\}$;
- Example: $Manages \equiv Managedby^{-}$;
- Description of Example: The $Manages$ role is the inverse of the $Managedby$ role.

- ⋇ Role: $R_{|C}$
  - ∗ Constructor name: Role Restriction;
  - ∗ Description: This role is obtained by confining the range of a certain role R to the individuals of a certain concept C;
  - ∗ Semantics: $(R_{|C})^{\mathcal{I}} = \{\langle x, y\rangle |\ R^{\mathcal{I}}\text{-}Successor\text{-}of\ x \text{ is in } C^{\mathcal{I}}\}$;
  - ∗ Example: $Managedbyfemales \equiv Managedby|_{Female}$;
  - ∗ Description of Example: The $Managedbyfemales$ role is a role that contains the pairs from the $Managedby$ role which have a female as their second argument.
- ⋇ Role: $TR(R)$
  - ∗ Constructor name: Transitive Role;
  - ∗ Description: This constructor specifies that the role R is transitive;
  - ∗ Semantics: $(TR(R))^{\mathcal{I}}$: if $\langle x, y\rangle \in R^{\mathcal{I}}$ and $\langle y, z\rangle \in R^{\mathcal{I}}$, then $\langle x, z\rangle \in R^{\mathcal{I}}$ [18];
  - ∗ Example: $TR(Partof)$;
  - ∗ Description of Example: The $Partof$ role is a transitive role.

- Language Construction
  - ⋇ Language: $\mathcal{DL}(\mathcal{D}_1, ..., \mathcal{D}_k)$ where $\mathcal{D}_1, ..., \mathcal{D}_k$ are domains.
    - ∗ Constructor name: Concrete Domain [19];
    - ∗ Description: This construction allows the user to employ strings, numbers or other domains in a description logic language [20]. A DL language can be equipped with several domains, $\mathcal{DL}(\mathcal{D}_1$, ..., $\mathcal{D}_k)$, together with partial functions (f_i) and additional predicates.

* Semantics: A *feature* $f_i^{\mathcal{I}}$ is a partial function from $\Delta^{\mathcal{I}}$ to $\bigcup_{1\leq j\leq n} \Delta^{\mathcal{D}_j}$, and a *predicate* p with arity $n \geq 0$ is a set $p^{\mathcal{D}_i}$, s.t. $p^{\mathcal{D}_i} \subseteq (\Delta^{\mathcal{D}_i})^n$ [19]. Every *concrete domain* $\Delta^{\mathcal{D}_i}$ has an associated set of predicates ($pred(\mathcal{D}_i)$). In fact, $p(f_1, ..., f_k)$ for $p \in pred(\mathcal{D}_i)$ is interpreted as $\{\ d \in \Delta^{\mathcal{I}} \mid \exists y_1, ..., y_k \in \Delta^{\mathcal{D}_i} : f_j^{\mathcal{I}} = y_j$ for $1 \leq j \leq k \wedge (y_1, ..., y_k) \in p^{\mathcal{D}_i}\}$;
* Example: $\mathcal{EL}(integer)$ equipped with a feature $Wage$ which returns the monthly wage of an individual and a predicate $\geq_{5000}$ which is a unary predicate with its obvious meaning over the integers;
* Description of Example: With the above language, specifying all individuals that manage at least three females each of whom earn 5000 dollars or more every month, is modelled by $(\geq 3\ Manages\ Females \sqcap \exists Wage.\geq_{5000})$.

A DL-*knowledge-base* (i.e., Σ-kn= $\langle TBox, ABox\rangle$) [21] is a finite set of DL-*Sentences* and the interpretation $.^{\mathcal{I}}$ is a model for the knowledge base Σ-kn if and only if $\mathcal{I}$ satisfies all the sentences in Σ-kn. Also, a Σ-kn is divided into two parts, TBox Sentences and ABox Sentences, where TBox and ABox are, respectively, the abbreviations of Terminological Box and Assertional Box. TBox statements describe "the relevant notions of an application domain by stating properties of concepts and roles, and relationships between them – it corresponds to the schema in a database setting" [22], and ABox statements are the axioms about a specific application situation stated by introducing named individuals (e.g., $Lecturer$(Franz), $Course$(C1)) and relating them by using roles (e.g., $teaches$(Franz, C1)) [23]. They constitute the "facts" of the domain. A TBox in a knowledge-base can be empty, cyclic or acyclic (see Definition 2). Below, we present the $\mathcal{ALC}$ language and an example of an

acyclic TBox and an ABox in the $\mathcal{ALC}$ language.

Definition 2 ([24])**.** A TBox $\mathcal{T}$ is called *cyclic* iff it contains a *cycle*, i.e., there exists an atomic concept defined (in)directly in terms of itself; otherwise $\mathcal{T}$ is called *acyclic*.

Definition 3 ([7])**.** $\mathcal{ALC}$ Concept Descriptions

Let N_C be the set of all concept names and N_R the set of all role names; a concept description is inductively defined as follows:

- An atomic concept name is a concept description.
- if $C \in N_C$ and $D \in N_C$ are concept descriptions and $r \in N_R$ is a role name, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists r.C$, $\forall r.C$ are concept descriptions.

The following example includes an ABox and an acyclic TBox, generated by the ABox and the TBox axioms mentioned in the Table 2.1. The TBox and ABox describe a small part of an education system [25]. Here *Person* and *Course* are atomic concepts and *teaches* and *attends* are roles while *Lecturer*, *Student*, *BusyLecturer* are concept descriptions.

Example 1. A simple description of an education system:

TBox Axioms:

$Lecturer \equiv Person \sqcap \exists teaches.Course$

$Student \equiv Person \sqcap \exists attends.Course$

$BusyLecturer \equiv Lecturer \sqcap (\geq 3\ teaches.Course)$

$Lecturer \sqcap Student \sqsubseteq \bot$

ABox Axioms:

$Lecturer$(Franz), $Course$(C1), $teaches$(Franz, C1),

$Course$(C2), $Course$(C3), $Course$(C4)

$Person$(Franz), $Person$(John), $Student$(John)

$attends$(John,C1), $teaches$(Franz, C3), $teaches$(Franz, C4)

Having identified TBox and ABox axioms, one can use a reasoner to make inferences, i.e., deduce implicit knowledge (e.g., Franz is a $BusyLecturer$) from the explicitly represented knowledge [26] in the system. A reasoner is a program which uses the TBox and the ABox, to draw inferences. To this point in time, many reasoners have been developed, e.g., Pellet [27], FACT++ [28], Condor, Hermit [29]; different reasoners are optimized to provide different reasoning services, e.g., classification [30, 31, 32, 33, 34], "the computation of subsumption hierarchies for classes and properties" [35]. Generally, there are six primitive reasoning types in two categories.

- TBox reasoning [36]

    1. **Satisfiability Checking**

       A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ s.t. $C^{\mathcal{I}}$ is non-empty ($C^{\mathcal{I}} \neq \emptyset$). In the Example 1, all the concepts are satisfiable because we can instantiate at least one individual for each concept. Suppose we add one axiom, $Stu\_Lec = Lecturer \sqcap Student$, to the TBox of Example 1. The new concept (Stu_Lec) describes the people who are a student and a lecturer simultaneously. On the other hand, the axiom $Lecturer \sqcap Student \sqsubseteq \bot$ states that nobody can be a student and a lecturer at the same time. The definition of Stu_Lec contradicts the axiom $Lecturer \sqcap Student \sqsubseteq \bot$. It is not possible to find an individual for Stu_Lec; thus Stu_Lec is not satisfiable.

    2. **Subsumption Checking**

       With respect to $\mathcal{T}$, we say C is subsumed by D, equivalently, D subsumes C, if every instance of concept C is an instance of concept D. This means $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. For example,

$BusyLecturer \sqsubseteq Person$, because based on the TBox axioms, every $BusyLecturer$ is a $Lecturer$ and every $Lecturer$ is a $Person$; therefore, every $BusyLecturer$ is a $Person$.

3. **Equivalence Checking**

   Two concepts C, D are equivalent if they always have the same interpretation in a model w.r.t. a particular TBox.

4. **Disjointness Checking**

   With respect to $\mathcal{T}$, two concepts C, D are disjoint if they have no common individual. Hence, we have $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$. As can be seen in the TBox of Example 1, *Student* and *Lecturer* are disjoint in the sense that it is not possible to have an individual who is a lecturer and a student at the same time.

- ABox reasoning [36]

1. **Consistency Checking**

   An ABox is consistent w.r.t. a TBox if there is a model $\mathcal{I}$ for both the ABox and the TBox. The ABox of Example 1 is consistent, but if we add $Lecturer(John)$ to the ABox, it is no longer consistent, because a TBox axiom states that *Student* and *Lecturer* are disjoint but *John* is an individual who is both.

2. **Instance Checking**

   Checking whether an individual is an instance of a concept (i.e., $a^{\mathcal{I}} \in C^{\mathcal{I}}$).

Indeed, in order to ensure the reasonable and predictable behaviour of a DL system, the aforementioned reasoning types should be decidable with a low computational complexity [7]. Table 2.2, shows the complexity of

subsumption reasoning in some basic description logics and Table 2.3 describes the complexity of satisfiability in some of the nonbasic description logics. There is a trade off between expressivity and complexity; generally, a more expressive logic has a higher computational complexity of (TBox, ABox) reasoning. For example, $\mathcal{ELHIF}_{\mathcal{R}^+}$ (See Table 2.4) is an extension of $\mathcal{EL}$ which contains role hierarchies, inverse, functional roles but has no tractable reasoning algorithm [37].

Practically, in most cases, one type of reasoning can be reduced to another. Thus, a reasoner needs to implement some of the reasoning types as primitive services, and performs the other types based on the implemented ones. The reductions are as follows. Suppose C and D are concepts and a is an individual.

- Reduction to Subsumption [8]
  - Unsatisfiability to subsumption: C *is unsatisfiable* $\Leftrightarrow C \sqsubseteq \bot$
  - Equality to subsumption: $C \equiv D \Leftrightarrow C \sqsubseteq D$ *and* $D \sqsubseteq C$
  - Disjointness to subsumption: C, D *are disjoint* $\Leftrightarrow C \sqcap D \sqsubseteq \bot$
- Reduction to Unsatisfiability [8]
  - Subsumption to unsatisfiability: $C \sqsubseteq D \Leftrightarrow C \sqcap \neg D$ *is unsatisfiable.*
  - Equality to unsatisfiability: $C \equiv D \Leftrightarrow C \sqcap \neg D$ *and* $\neg C \sqcap D$ *are unsatisfiable.*
  - Disjointness to unsatisfiability: C, D *are disjoint* $\Leftrightarrow C \sqcap D$ *is unsatisfiable.*
- Reduction of Instance Checking to Inconsistency [8]
  - *ABox* $\mathcal{A} \models C(a) \Leftrightarrow \mathcal{A} \sqcup \{\neg C(a)\}$ *is inconsistent.*

- Reduction of Satisfiability to Consistency [8]
  - *C is satisfiable $\Leftrightarrow$ $\{C(a)\}$ is consistent.*

As we already mentioned in the introduction, there are many different kinds of description logics. Three of them, $\mathcal{AL}$, $\mathcal{EL}$ and $\mathcal{FL}_0$ mentioned in the first part of the Table 2.4 are basic logics, and the others are constructed by adding some constructors to these basic logics (see the second part of the Table 2.4). For example, $\mathcal{ALC}$ is the $\mathcal{AL}$ language equipped with *complete negation* ($\mathcal{C}$), while $\mathcal{SHIQ}$ has many constructors [38]: *Top concept, Bottom concept, Conjunction, Disjunction, Complete Negation, Existential Restriction, Universal Restriction and Qualifying Number Restriction.* $\mathcal{SHIQ}$ is based on the $\mathcal{AL}$ language.

Table 2.2: Computational complexity of Subsumption [7, 39]

Assumption	Language	Complexity
Empty TBox	$\mathcal{FL}_o$	Polynomial
General TBox	$\mathcal{FL}_o$	ExpTime-Complete
General TBox	$\mathcal{EL}$	Polynomial
General TBox	$\mathcal{ALC}$	ExpTime

2.2 A tableau algorithm for $\mathcal{ALC}$

A tableau algorithm can be designed to check the satisfiability of a concept description in a description logic. For example, in order to check the satisfiability of a given $\mathcal{ALC}$ concept description C_0, a tableau algorithm tries to construct a finite interpretation $\mathcal{I}$ that satisfies C_0 [41]. Before we explain the details of the algorithm, we define the following notion.

Table 2.3: Computational complexity of Satisfiability [40]

Assumption	Language	Complexity
-	$\mathcal{ALC}$	PSPACE-Complete
-	$\mathcal{ALCO}$	PSPACE-Complete
Empty TBox	$\mathcal{ALCF}$	PSPACE-Complete
-	$\mathcal{ALCF}$	EXPTIME-Complete
Empty TBox	$\mathcal{ALCN}$	PSPACE-Complete
-	$\mathcal{ALCN}$	EXPTIME-Complete
Acyclic TBox	$\mathcal{ALCF}$	NEXPTIME-Hard
-	$\mathcal{SHIQ}$	EXPTIME-Complete

Table 2.4: DL Naming Convention [5]

Symbol	Description
$\mathcal{AL}$	Atomic negation, Concept intersection, Value restrictions, Limited existential quantification.
$\mathcal{FL}$	Concept intersection, Value restrictions, Limited existential quantification, Role restriction.
$\mathcal{EL}$	Concept intersection, Full existential quantification.
$\mathcal{S}$	An abbreviation for $\mathcal{ALC}$ with transitive roles.
$\mathcal{FL}^-$	A sub-language of $\mathcal{FL}$, without role restriction. Equivalent to $\mathcal{AL}$ without atomic negation.
$\mathcal{FL}_o$	A sub-logic of $\mathcal{FL}^-$, without limited existential quantification.
$\mathcal{EL}^{++}$	Alias for $\mathcal{ELRO}$.
$\mathcal{H}$	Role hierarchy.
$\mathcal{O}$	Nominal.
$\mathcal{Q}$	Qualified cardinality restrictions.
$\mathcal{I}$	Inverse role.
$(\mathcal{D})$	Use of datatype properties, data values or data types.
$\mathcal{N}$	Cardinality restrictions.
$\mathcal{F}$	Functional properties.
$\mathcal{E}$	Full existential qualification.
$\mathcal{U}$	Concept union.
$\mathcal{C}$	Complex concept negation.
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.

Definition 4. A concept description is in *Negation Normal Form (NNF)* when the negation operator ($\neg$) is allowed to appear only before the atomic concepts.

It is more efficient to build a tableau to check the satisfiability of a concept description when all concepts are in NNF [41, 42]. Using negation normal form of concept descriptions in the construction of a tableau decreases the number of completion rules (see Figure 2.1) [43] and reduces the size of a tableau. In order to find the NNF of a formula, we need to push the negations inside the concept description using de Morgan's laws [44]. In fact, a concept description can be transformed to NNF in linear time [41] by applying the following rules.

1. $\neg(\neg C) \Leftrightarrow C$
2. $\neg(C \sqcup D) \Leftrightarrow \neg C \sqcap \neg D$
3. $\neg(C \sqcap D) \Leftrightarrow \neg C \sqcup \neg D$
4. $\neg(\forall R.C) \Leftrightarrow \exists R.\neg C$

Example 2. Let $C = \neg(\forall R.(D \sqcup E))$. The NNF of C is $\exists R.(\neg D \sqcap \neg E)$.

- The $\rightarrow_{\sqcap}$-rule
  $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$ if $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but not both $C_1(x)$ and $C_2(x)$
- The $\rightarrow_{\sqcup}$-rule
  $\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' := \{C_2(x)\}$ if $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$
- The $\rightarrow_{\exists}$-rule
  $\mathcal{A}' := \mathcal{A} \cup \{C(y), R(x, y)\}$ (y is an individual name not occurring in $\mathcal{A}$) if $\mathcal{A}$ contains $(\exists R.C)(x)$, and there is no individual name z s.t. $C(z)$ and $R(x, z)$ are in $\mathcal{A}$.
- The $\rightarrow_{\forall}$-rule
  $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$ if $\mathcal{A}$ contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$.

Figure 2.1: Completion Rule for $\mathcal{ALC}$ [36, 45]

Let $\mathcal{T}$ be a TBox, and let C be a concept description in NNF. In order to check the satisfiability of C w.r.t. $\mathcal{T}$, the algorithm starts with an ABox, $\mathcal{A}_0 = \mathcal{T} \cup \{C(a_0)\}$ and applies the completion rules (see Figure 2.1) in an arbitrary order where each rule produces one or two new ABoxes. These rules preserve the consistency of a concept. Applying the $\rightarrow_{\sqcup}$-rule produces two new ABoxes such that $\mathcal{A}_0$ is consistent iff at least one of the new ABoxes is consistent.

Definition 5 ([41])**.** Let C be a concept description, and let a_0 be an individual. An ABox contains a *clash* iff it contains $C(a_0)$ and $\neg C(a_0)$ simultaneously. An ABox is called *closed* if it contains a clash, and is called *open* otherwise.

A concept description C is *satisfiable* if one of the ABoxes, produced while applying the completion rules, remains open and no rule is applicable to any axiom in the ABox. If all the ABoxes are closed, C is *unsatisfiable*. When the TBox is empty or acyclic, the algorithm always terminates in a finite number of steps (see the proof in [46]), but if we have a cyclic TBox, the algorithm may not terminate. For example, let C be a concept description; let R a role name, and let $\mathcal{T} = \{C \equiv \exists R.C\}$. The process of checking the satisfiability of C is started with $\mathcal{A}_0 = \{C \equiv \exists R.C, C(a_0)\}$ and never terminates. Figure 2.2 exhibits a fragment of the tableau for C. This process does not terminate because there is a cycle (recall Definition 2) in $\mathcal{T}$; so a technique is required to detect the cycles and to prevent the non-termination. So far, different techniques have been proposed in the literature, e.g., subset blocking [46], dynamic blocking [46] and dynamic double blocking [47].

Definition 6 ([25])**.** (Subset Blocking.) Let a, b be two individuals in $\mathcal{A}$. b is *blocked* by a iff $\{C|C(b) \in \mathcal{A}\} \subseteq \{C|C(a) \in \mathcal{A}\}$.

$$\mathcal{A}_0 = \{C \equiv \exists R.C, C(a_0)\}$$
$$\downarrow \textit{Applying} \rightarrow_\exists\text{-rule}$$
$$\mathcal{A}_1 = \{C \equiv \exists R.C, C(a_0), R(a_0, a_1), C(a_1)\}$$
$$\downarrow \textit{Applying} \rightarrow_\exists\text{-rule}$$
$$\mathcal{A}_2 = \{C \equiv \exists R.C, C(a_0), R(a_0, a_1), C(a_1), R(a_1, a_2), C(a_2)\}$$
$$\downarrow \textit{Applying} \rightarrow_\exists\text{-rule}$$
$$\mathcal{A}_3 = \{C \equiv \exists R.C, C(a_0), R(a_0, a_1), C(a_1), R(a_1, a_2), C(a_2), R(a_2, a_3), C(a_3)\}$$
Never terminates!

Figure 2.2: Tableau for satisfiability checking of concept description C

Here we adopt the subset blocking technique in order to resolve the termination issue. Definition 6 presents the notion of a blocked individual. Based on this notion, we have modified the completion rules (see Figure 2.3), so they are not allowed to be applied on blocked individuals. When b is blocked by a, b can use the role successors of a; so it is no longer needed to generate a new individual. For example, we can construct an alternative version of Figure 2.2 in which a_1 would be blocked by a_0, and we would use a_1 rather than generate a_2. Hence, we would have $\mathcal{A}_2 = \{C \equiv \exists R.C, C(a_0), R(a_0, a_1), C(a_1), R(a_1, a_1)\}$, and no further rule is applicable, which means the algorithm terminates. Note, in some cases, two individuals a and b can block each other. This situation is called cyclic blocking and should be prevented by the algorithm. To do so, we consider an enumeration of all the individuals. We allow an individual a to block the individual b if a appears before b in the enumeration. In $\mathcal{ALC}$, using subset blocking along with prevention of cyclic blocking guarantees that the algorithm always terminates.

Theorem 1. The tableau algorithm for checking satisfiability of a concept description C in $\mathcal{ALC}$ always terminates.

- The $\rightarrow_{\sqcap}$-rule
  $\mathcal{A}^{'} := \mathcal{A} \cup \{C_1(a), C_2(a)\}$ if a is not blocked and $\mathcal{A}$ contains $(C_1 \sqcap C_2)(a)$, but not both $C_1(a)$ and $C_2(a)$

- The $\rightarrow_{\sqcup}$-rule
  $\mathcal{A}^{'} := \mathcal{A} \cup \{C_1(a)\}$, $\mathcal{A}^{''} := \{C_2(a)\}$ if a is not blocked and $\mathcal{A}$ contains $(C_1 \sqcup C_2)(a)$, but neither $C_1(a)$ nor $C_2(a)$

- The $\rightarrow_{\exists}$-rule
  $\mathcal{A}^{'} := \mathcal{A} \cup \{C(b), R(a, b)\}$ (b is an individual name not occurring in $\mathcal{A}$) if a is not blocked, and $\mathcal{A}$ contains $(\exists R.C)(a)$, but there is no individual name c s.t. $C(c)$ and $R(a, c)$ are in $\mathcal{A}$.

- The $\rightarrow_{\forall}$-rule
  $\mathcal{A}^{'} := \mathcal{A} \cup \{C(b)\}$ if a is not blocked and $\mathcal{A}$ contains $(\forall R.C)(a)$ and $R(a, b)$, but it does not contain $C(b)$.

Figure 2.3: Completion Rule for $\mathcal{ALC}$ [36, 45]

Proof. Let $\mathcal{T}$ be a TBox; let $\mathcal{A}$ be an ABox; let C, D be two concept descriptions, and let $sub(\mathcal{O})$ be the set of all subconcepts of the concepts appearing in $\mathcal{A}$ together with all subconcepts of $\text{NNF}(\neg C) \sqcup D$ for each $C \sqsubseteq D \in \mathcal{T}$. The termination is a consequence of following facts [48]:

- A rule replaces an ABox with at most two ABoxes.

- The ABoxes are built in a monotonic way in relation to the size of the ABox.

- Let $|C|$ be the string length of concept description C. The number of concept assertions which can be added during the process is restricted to the following number:

  $$sub(\mathcal{O}) \leq \Sigma_{C \subseteq D \in \mathcal{O}}(2 + |C| + |D|) + \Sigma_{a:C \in \mathcal{O}}|C|$$

  It can be proved by induction that $sub(C) \leq |C|$.

- When we use subset blocking, if a is blocked, no rule can be applied on

a; so there can be at most $2^{sub(\mathcal{O})}$ individuals in an ABox.

Hence, the ABoxes which are built during the satisfiability checking of a concept will be complete in the sense that each of them either may contain a clash, or is consistent, and no further rule is applicable to any axiom in it. Therefore, the algorithm terminates. □

Since there are many different members in the family of description logics, different tableau rules and different blocking techniques should be used for reasoning in them. For the details see [41, 42, 44, 46, 47, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61]

2.3 Relationship between Description Logic and First Order Predicate Logic

Most of description logics are decidable fragments of first-order logic [62]. Therefore, sometimes we can use a translation into predicate logic to define the semantics of description logic formulas. Generally, this transformation yields a first order formula but there are some situations which we cannot transform to first order formulas, e.g., those involving a transitive role, or nominals. Since we do not have any variables in the syntax of a DL, translation of a concept name yields a formula with one free variable while a role name is translated to a formula with two free variables [21, 36, 62]. A translation is given by a mapping π_x from DL concepts into predicate logic formulas with one free variable and a mapping $\pi_{x,y}$ from DL roles into predicate logic formulas with two free variables. The translation is inductively defined as follows [36].

$$\pi_x(\top) = true \qquad \pi_x(\bot) = false$$
$$\pi_x(A) = A(x) \qquad \pi_x(\neg A) = \neg\pi_x(A)$$
$$\pi_{x,y}(R) = R(x,y) \qquad \pi_{x,y}(R^-) = \pi_{y,x}(R)$$
$$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D) \qquad \pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$$
$$\pi_x(\exists R.C) = \exists y.\pi_{x,y}(R) \wedge \pi_y(C) \qquad \pi_x(\exists R.\top) = \exists y.\pi_{x,y}(R)$$
$$\pi_x(\forall R.C) = \forall y.\pi_{x,y}(R) \rightarrow \pi_y(C) \qquad \pi_x(a) = F(x) \text{ where } \forall x F(x) = a$$
$$\pi_{x,y}(R_{|C}) = \pi_{x,y}(R) \wedge \pi_y(C)$$
$$\pi_x(\geq n\ R) = \exists y_1, \ldots, y_n.(\textstyle\bigwedge_{i\neq j} y_i \neq y_j \wedge \bigwedge_i \pi_{x,y_i}(R))$$
$$\pi_x(\leq n\ R) = \forall y_1, \ldots, y_{n+1}.(\textstyle\bigwedge_{i\neq j} y_i \neq y_j \rightarrow \bigvee_i \neg\pi_{x,y_i}(R))$$
$$\pi_x(\geq n\ R\ C\ ) = \exists y_1, \ldots, y_n.(\textstyle\bigwedge_{i\neq j} y_i \neq y_j \wedge \bigwedge_i (\pi_{x,y_i}(R)) \wedge \pi_y(C)))$$
$$\pi_x(\leq n\ R\ C) = \forall y_1, \ldots, y_{n+1}.(\textstyle\bigwedge_{i\neq j} y_i \neq y_j \rightarrow \bigvee_i \neg(\pi_{x,y_i}(R)) \rightarrow \pi_y(C)))$$

When we transform a DL concept to a formula in predicate logic, we preserve the satisfiability of the concept in the sense that if the predicate formula is satisfiable, the translated DL concept is also satisfiable. Given a TBox $\mathcal{T} = \{C_i \sqsubseteq D_i | 1 \leq i \leq n_1\} \cup \{C_i \equiv D_i | 1 \leq i \leq n_2\} \cup \{R_i \sqsubseteq S_i | 1 \leq i \leq n_3\}$ and a translation π from description logic concepts into first order formulas, we define

$$\pi(\mathcal{T}) = \forall x. \textstyle\bigwedge_{i=1}^{n_1}(\pi_x(C_i) \rightarrow \pi_x(D_i)) \wedge \forall x. \bigwedge_{i=1}^{n_2}(\pi_x(C_i) \leftrightarrow \pi_x(D_i))$$
$$\wedge\ \forall x, y. \textstyle\bigwedge_{i=1}^{n_3}(\pi_{x,y}(R_i) \rightarrow \pi_{x,y}(S_i))$$

Now, we have:

- C is satisfiable w.r.t. $\mathcal{T}$ iff the formula $\pi_x(C) \wedge \pi(\mathcal{T})$ is satisfiable;
- $C \sqsubseteq D$ is satisfiable w.r.t. $\mathcal{T}$ iff $\pi_x(C) \wedge \neg\pi_x(D) \wedge \pi(\mathcal{T})$ is unsatisfiable;
- $C \equiv D$ is satisfiable w.r.t. $\mathcal{T}$ iff $\pi_x(C) \wedge \neg\pi_x(D) \wedge \pi(\mathcal{T})$ and $\neg\pi_x(C) \wedge \pi_x(D) \wedge \pi(\mathcal{T})$ are unsatisfiable;
- $R \sqsubseteq S$ is satisfiable w.r.t. $\mathcal{T}$ iff $\pi_{x,y}(R) \wedge \neg\pi_{x,y}(S) \wedge \pi(\mathcal{T})$ is unsatisfiable;

- An ABox $\{R_k(a_i, a_j)|$ all applicable $i, j, k\} \cup \{C_j(a_i)|$ all applicable i,j$\}$ is consistent w.r.t. $\mathcal{T}$ iff the following formula is satisfiable, where a_is are constants corresponding to the individuals in the ABox.

$$\bigwedge_{i,j,k} R_k(a_i, a_j) \wedge \bigwedge_{i,j} \pi_x(C_j)(a_i) \wedge \pi(\mathcal{T})$$

Chapter 3

Temporal Logic

A *temporal logic* is used to model any system of rules and symbolism in order to represent and to reason about propositions qualified in terms of time. These logics are useful in many different areas, e.g., it has enough power to model the behaviours of some systems in terms of logical formulas, including temporal constraints, events, and the relationships between them [64]. Moreover, temporal logic can be used for the formal verification of a software or a hardware system. For instance, a statement like "whenever a request is made, access to a resource is *eventually* granted, but it is *never* granted to two requestors simultaneously" [63] can be easily modeled by a temporal logic, and then a model checker [65] (e.g., SPIN [66]) or a theorem prover [67] (e.g., KIV [68]) is used to determine if this statement is always true in the system.

In the domain of medicine, most events and procedures are time-dependent, e.g., when symptoms for an illness occur [69]; therefore, temporal logic plays an important role in this domain because it can be used to describe such events or procedures. Moreover, a temporal logic can model the properties that should hold whenever an event occurs (e.g., after any Insulin shot, the blood sugar eventually drops) or during a procedure (e.g., during an operation, the heart

rate of a patient must always be monitored).

Until today, several temporal logics have been designed in order to be used in various areas. Generally, these logics are divided into two main categories: point-based temporal logics and interval-based temporal logics. We explain these categories below.

3.1 Point-based Temporal Logics

A point-based temporal logic is essentially an extension of a propositional logic using some modalities, e.g., the necessity modality [70]. These modalities allow one to add dynamicity to classical logic. It means that a formula of a point-based temporal logic is evaluated on a set of time points. This set of time points is called *linear*, if each moment in time has a unique possible future; otherwise it is called *branching*, i.e., several possible futures may exist for each moment in time [71]. Based on these notions (linear or branching), point-based temporal logics are classified into two categories: linear time point-based temporal logics (e.g., linear time temporal logic (LTL) [72], Quantified Propositional Temporal Logic (QPTL) [73]) and branching time point-based temporal logics (e.g., Computation Tree Logic (CTL) [71, 74], CTL* [75]). These languages have different expressivity. For instance, CTL* is more expressive than CTL and LTL. Note that sometimes the expressivity of two temporal logics are not comparable. For example, the expressivity of LTL and the expressivity of CTL are not comparable in the sense that LTL is able to model some situations that CTL cannot model, and vice versa.

The main ontological element of point-based logics is a time instant (point), which is a *durationless* element [76]. Therefore, these logics are mostly suitable for modelling instantaneous events (or actions) in a system. Moreover, most

of these logics are decidable and have good computational properties [77]. For more information, we refer the reader to [78, 79, 80, 81, 82].

3.2 Interval-based Temporal Logic

There are many scientific areas, e.g., philosophy, linguistics, artificial intelligence and computer science [83] which involve processes which have durations. It is difficult to model processes with durations using a point-based temporal logic. Furthermore, there are other situations that a point-based temporal logic is not able to model, e.g., expressing the decomposition of one time interval into several [84]. To resolve these issues, a temporal logic, called *interval-based temporal logic*, may be used. In this kind of logic, a formula is evaluated on a set of time intervals. In the next section we will talk about the structure of a time interval in more detail.

In some interval-based temporal logics, time intervals are defined in terms of one of their endpoints. In these logics, a *locality assumption*, which says an atomic proposition is true if and only if it is true at the first point of the interval, reduces intervals to time points [83]. Thus, these logics turn out to be point-based temporal logics. In this thesis, time intervals will be considered as primitive entities [76] in the logics; so the logics are inherently interval-based, and it is not possible to reduce them to point-based temporal logics.

3.3 Structure of Time Intervals

Definition 7. Let $\mathbb{D} = \langle \mathrm{D}, < \rangle$ be a partially ordered set (see [85]).

- An *interval* in $\mathbb{D}$ is a pair [a,b], where a,b $\in$ D and a $<$ b or $a = b$.

- If $a < b$, the interval is called *strict* or *proper*, otherwise, it is called a *point interval*.
- The set of strict intervals are denoted by $\mathbb{I}(\mathbb{D})^-$ while the set of all non-strict intervals is denoted by $\mathbb{I}(\mathbb{D})^+$.
- By $\mathbb{I}(\mathbb{D})$ we will denote either $\mathbb{I}(\mathbb{D})^+$ or $\mathbb{I}(\mathbb{D})^-$.
- An *interval structure* is a pair $\langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle$ [76].

Definition 8. An interval structure is *linear* if every two points are comparable.

Also, a partial ordering has the *linear interval property* if every interval in the structure is linear. Formally,

$$\forall x \forall y (x < y \rightarrow \forall k_1 \forall k_2 (x < k_1 < y \wedge x < k_2 < y \rightarrow k_1 < k_2 \vee k_1 = k_2 \vee k_2 < k_1))$$

Figure 3.1 exhibits an interval structure with(out) the linear interval property. Note that every linear interval structure has the linear interval property.

Figure 3.1: With linear interval property (left), Without linear interval property (right)

A linear interval structure can be:

- *finite*, if it has finitely many points [76].
- *unbounded above (below)*, if every point has a successor (predecessor) [76].

- *dense*: Between every two distinct points, a third point exists, $\forall x, y(x < y \rightarrow \exists z\ x < z < y)$;
- *discrete*: Every point in the flow with a successor/predecessor has an immediate successor/predecessor.
  - $\forall x, y(x < y \rightarrow \exists z(x < z \wedge \neg\exists x < u < z)$
  - $\forall x, y(x < y \rightarrow \exists z(z < y \wedge \neg\exists z < u < y)$
- *dedekind complete*: Every non-empty set of points which is bounded above has a least upper bound [76].

Note that in this thesis, we use $\mathbb{Z}$ as our linear interval structure.

3.4 Possible relations between two intervals

Generally, thirteen relations can be defined between two intervals. Allen introduced them in [86] and, hence, the relations are called Allen's relations [87] (see Figure 3.2). Moreover, the three following natural relations between two intervals can be defined [88] based on Allen's relations. Let $\mathbb{D}$ be a partially order set, and let $[c_i, c_j]$ and $[c_{i'}, c_{j'}]$ be two intervals.

- $[c_{i'}, c_{j'}]$ is a *sub-interval* of $[c_i, c_j]$ iff $c_i \leq c_{i'}$ and $c_{j'} \leq c_j$.
  This relation corresponds to the $Equal \cup Starts \cup During \cup Ends$ relation.
  Formally, $[c_{i'}, c_{j'}] \equiv \langle = \rangle[c_i, c_j] \vee \langle B \rangle[c_i, c_j] \vee \langle D \rangle[c_i, c_j] \vee \langle E \rangle[c_i, c_j]$.
- $[c_{i'}, c_{j'}]$ is a *proper sub-interval* of $[c_i, c_j]$ iff $[c_{i'}, c_{j'}]$ is a *sub-interval* of $[c_i, c_j]$ and $[c_{i'}, c_{j'}] \neq [c_i, c_j]$.
  This relation corresponds to the $Starts \cup During \cup Ends$ relation. Formally, $[c_{i'}, c_{j'}] \equiv \langle B \rangle[c_i, c_j] \vee \langle D \rangle[c_i, c_j] \vee \langle E \rangle[c_i, c_j]$.

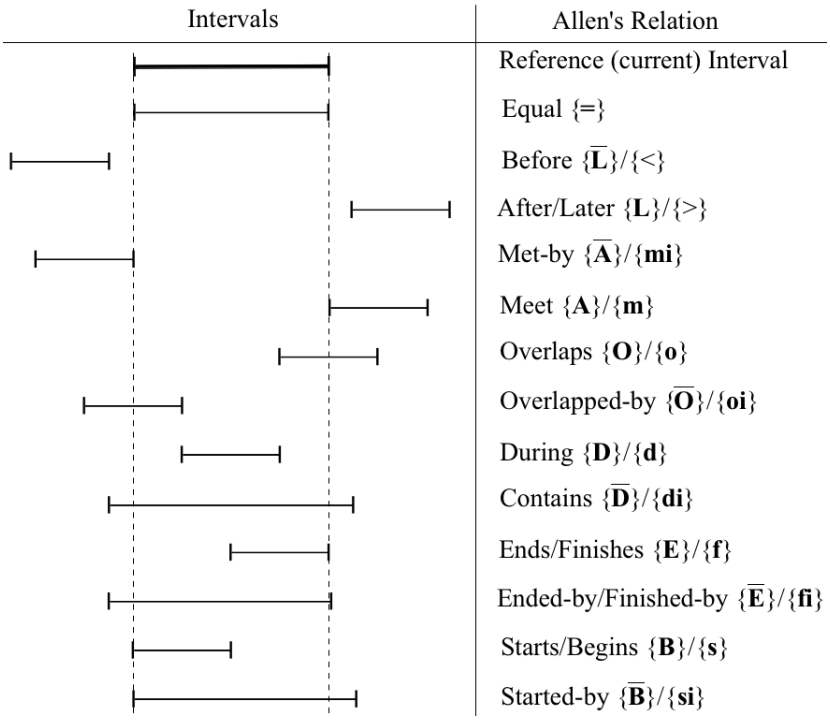

Figure 3.2: Allen's relations with alternative notations [11, 83]

- $[c_{i'}, c_{j'}]$ is a *strict sub-interval* of $[c_i, c_j]$ iff $c_i < c_{i'}$ and $c_{j'} < c_j$.
  This relation corresponds to the *During* relation. Formally, $[c_{i'}, c_{j'}] \equiv \langle D \rangle [c_i, c_j]$.

In [89], Venema has introduced a *ternary* relation (named **A**) between intervals. Figure 3.3 exhibits the semantics of this relation [89]. Note that some logics (e.g., CDT, BCDT$^+$) are defined based on the **A** relation.

$\mathbf{A}ijk$ holds iff i **starts** k and i **meets** j and j **ends** k

where i, j, k are three intervals.

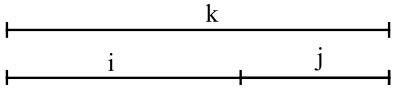

Figure 3.3: Venema relation

3.4.1 Allen's modalities

Based on Allen's relations, thirteen different modalities are defined. These modalities are $\langle A\rangle, \langle\overline{A}\rangle, \langle L\rangle, \langle\overline{L}\rangle, \langle E\rangle, \langle\overline{E}\rangle, \langle B\rangle, \langle\overline{B}\rangle, \langle D\rangle, \langle\overline{D}\rangle, \langle O\rangle, \langle\overline{O}\rangle, \langle=\rangle$ which respectively stand for meet, met-by, after, before, ends, ended-by, starts, started-by, during, contain, overlaps, overlapped-by and equal. These modalities are unary modalities. We give an example to show how these modalities are used. Let ψ and φ be two formulas. The formula $\psi = \langle B\rangle\varphi$ states that the intervals on which that ψ and φ are true start at the same time while the length of the interval on which φ is true is shorter than the length of the interval on which ψ is true (the current interval). Note that some of the modalities can be defined in terms of other modalities, e.g., $\langle B\rangle\varphi = \langle\overline{A}\rangle\langle A\rangle\varphi$, $\langle D\rangle\varphi = \langle B\rangle\langle E\rangle\varphi$, $\langle L\rangle\varphi = \langle A\rangle\langle A\rangle\varphi$ [90]. Due to this fact, some logics may contain few modalities while they are able to model many of Allen's relations. For instance, a logic which has the $\langle A\rangle, \langle\overline{A}\rangle, \langle\overline{L}\rangle, \langle L\rangle, \langle E\rangle$ and $\langle\overline{E}\rangle$ modalities, is able to model all of Allen's relations [91].

3.5 Some interval-based temporal logics

In this section, we consider some of the logics that are genuinely interval-based in the sense that they cannot be directly translated into a point-based logic, and they do not assume locality or any semantic restrictions that reduce the interval-based semantics to the point-based semantics [11].

3.5.1 Interval Temporal Logic ($\mathcal{ITL}$) [92, 93]

$\mathcal{ITL}$, introduced by Halpern, Monna and Moszkowski [93], is used to describe the behaviour of software and hardware. For instance, in [92], $\mathcal{ITL}$ has been used to describe and to reason about the structure and dynamics of a wide variety of timing-dependent digital circuits. This logic is an extension of linear time temporal logic (LTL), while the notion of time in $\mathcal{ITL}$ is bounded for the past, unbounded for the future, discrete and linear [93]. However a model of LTL consists of infinite state sequences while a model of $\mathcal{ITL}$ consists of finite state sequences, called *intervals* [94]. The length of an interval is equal to the number of states existing in a the sequence. We present the syntax of propositional $\mathcal{ITL}$ ($\mathcal{PITL}$) below.

Definition 9 ([92])**.** Let p be a proposition, and let $n \in \mathbb{N}$. A $\mathcal{PITL}$ formula, denoted by P, is defined inductively as follows.

$$P ::= p \mid false \mid \neg P \mid P \vee P \mid \bigcirc P \mid P; P$$

Since $\mathcal{ITL}$ is considered as an extension of LTL with the *semicolon* operator, the semantics of the $\neg$,$\vee$ and $\bigcirc$ operators is the same as the semantics of these operators in LTL (see [95]). The semantics of the *semicolon* operator is the same as for the semantics of the Venema relation which we described in the previous section. Note that the truth of formulas does not depend on states, but on intervals.

However because of the existence of the *semicolon* operator, $\mathcal{ITL}$ is undecidable [92, 93], the decidability of $\mathcal{PITL}$ can be recovered by constraining atomic propositions to be point-wise and by adopting the locality assumption [76]. In other words, the main ontological element in the decidable $\mathcal{PITL}$ is a time point. It follows that the decidable $\mathcal{PITL}$ is not genuinely an interval-based temporal logic.

3.5.2 The Halpern-Shoham logic ($\mathcal{HS}$)

$\mathcal{HS}$ is an interval-based temporal logic that allows one to express all Allen's relations between intervals [96]. This logic does not assume anything about the nature of an interval structure; so the interval structure can be discrete or dense, linear or branching, dedekind complete or not [97].

Let $\mathcal{AP}$ be a set of propositional variables, and $p \in \mathcal{AP}$. A formula of $\mathcal{HS}$, ψ, is defined inductively as follows [98].

$$\psi ::= p \mid \neg\psi \mid \psi \wedge \psi \mid \langle A \rangle\ \psi \mid \langle B \rangle\ \psi \mid \langle E \rangle\ \psi \mid \langle \overline{\mathrm{A}} \rangle\ \psi \mid \langle \overline{\mathrm{B}} \rangle\ \psi \mid \langle \overline{\mathrm{E}} \rangle\ \psi$$

As seen in the syntax of the logic, there are only 6 modalities for defining a formula. Recall that the other Allen's modalities are definable based on these 6 modalities. A model for an $\mathcal{HS}$ formula is a 3 tuple $M = (\mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{V})$ where $(\mathbb{D}, \mathbb{I}(\mathbb{D}))$ is an interval structure and $\mathcal{V} : \mathbb{I}(\mathbb{D}) \to 2^{\mathcal{AP}}$ is a valuation function that assigns to every interval the set of propositions which are true there. The semantics of $\mathcal{HS}$ is defined based on the satisfiability relation ($\models$) as follows [88]. Let $[i, j]$ be an interval.

- $M, [i, j] \models p$ iff $p \in V([i, j])$, for any $p \in \mathcal{AP}$;
- $M, [i, j] \models \neg\psi$ iff it is not the case that $M, [i, j] \models \psi$;
- $M, [i, j] \models \psi_1 \wedge \psi_2$ iff $M, [i, j] \models \psi_1$ *and* $M, [i, j] \models \psi_2$;
- $M, [i, j] \models \langle A \rangle \psi$ iff there exists $h \geq j$ such that $M, [j, h] \models \psi$;
- $M, [i, j] \models \langle B \rangle \psi$ iff there exists $i \leq h < j$ such that $M, [i, h] \models \psi$;
- $M, [i, j] \models \langle E \rangle \psi$ iff there exists $i < h \leq j$ such that $M, [h, j] \models \psi$;
- $M, [i, j] \models \langle \overline{\mathrm{A}} \rangle \psi$ iff there exists $h \leq i$ such that $M, [h, i] \models \psi$.
- $M, [i, j] \models \langle \overline{\mathrm{B}} \rangle \psi$ iff there exists $h > j$ such that $M, [i, h] \models \psi$.

- $M, [i, j] \models \langle \overline{\mathrm{E}} \rangle \psi$ iff there exists $h < i$ such that $M, [h, j] \models \psi$.

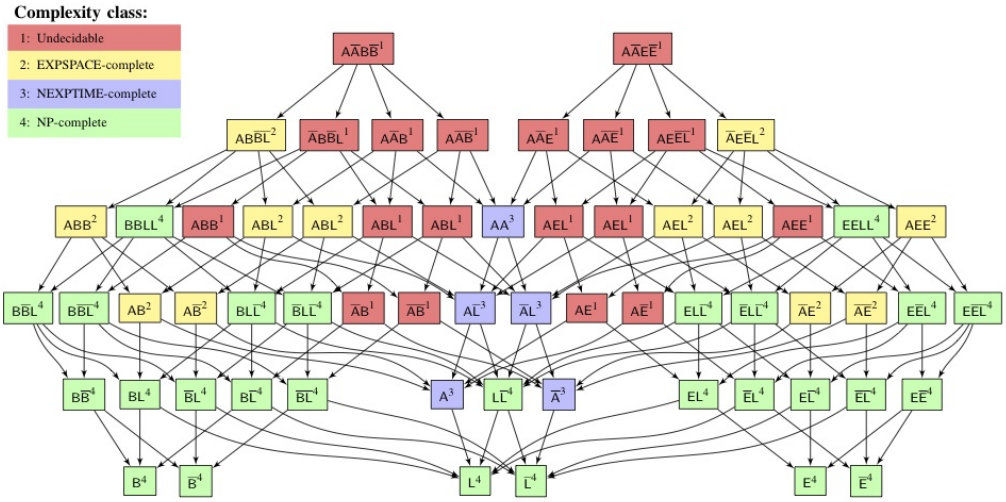

Figure 3.4: Fragments of $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ over strongly discrete linear orders [99]

$\mathcal{HS}$ is undecidable for most classes of interval structures (e.g., any class of ordered structures with an infinitely ascending sequence [88]) . Although most fragments of $\mathcal{HS}$ on most natural classes of interval structures (e.g., $\mathbb{N}, \mathbb{Z}$) are undecidable [90, 100], there are still many decidable fragments [99]. For example, the logic which contains the $\langle D \rangle, \langle \overline{D} \rangle, \langle B \rangle, \langle \overline{B} \rangle, \langle L \rangle, \langle \overline{L} \rangle$ modalities is the maximal decidable fragment over dense interval structures [11]. In terms of expressivity, there are exactly 44 expressively different decidable fragments of $\mathcal{HS}$ over discrete linear orders (see Figure 3.4), and their complexity ranges from NP to ExpSpace [99]. Later, we discuss one of the decidable fragments of $\mathcal{HS}$, called PNL.

3.5.3 CDT and BCDT$^+$ [83, 88, 89]

CDT, proposed by Venema in [89], is the most expressive propositional interval logic over a (non-strict) linear ordering [83, 88]. BCDT$^+$ is an extension of CDT to partial orderings with the linear interval property. Besides the usual boolean operators, the language of these logics contains three binary modal operators, C, D and T and a propositional constant π. A formula of these logics, denoted by ψ, is defined recursively as follows:

$$\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \pi \mid \psi \ C \ \psi \mid \psi \ D \ \psi \mid \psi \ T \ \psi$$

Figure 3.5 intuitively exhibits the semantics of the C, D and T operators.

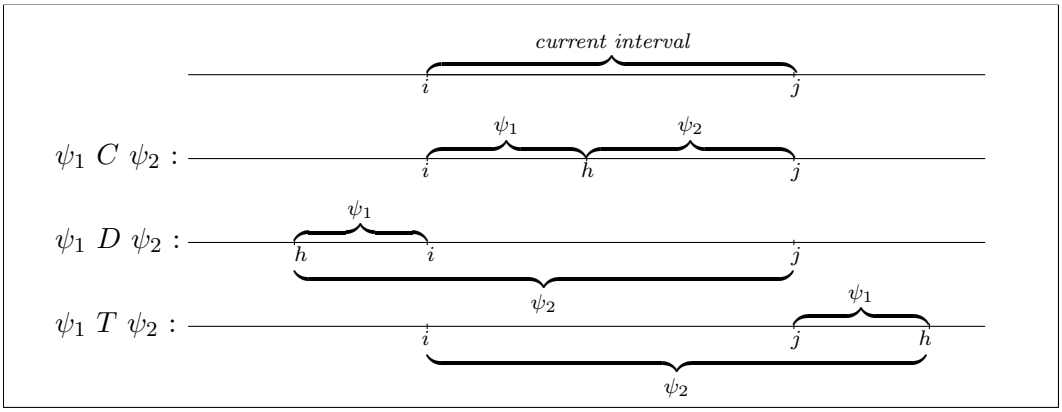

Figure 3.5: Intuitive semantics of C, D and T modalities [89]

The satisfiability relation of π and the C, D and T modalities is defined as follows [88]. Let M be a model similar to what we defined for the $\mathcal{HS}$ logic, and let $[i, j]$ be an interval.

- $M, [i, j] \models \pi$ iff $i = j$;
- $M, [i, j] \models \psi_1 \ C \ \psi_2$ iff there exists $i \leq h \leq j$ such that $M, [i, h] \models \psi_1$ and $M, [h, j] \models \psi_2$;

- $M, [i, j] \models \psi_1\ D\ \psi_2$ iff there exists $h \leq i$ such that $M, [h, i] \models \psi_1$ and $M, [h, j] \models \psi_2$;
- $M, [i, j] \models \psi_1\ T\ \psi_2$ iff there exists $h \geq j$ such that $M, [j, h] \models \psi_1$ and $M, [i, h] \models \psi_2$;

Note that in the following sense, $\mathcal{HS}$ logic is considered as a subset of CDT [83, 88].

- $\langle B \rangle \psi = \psi\ C\ (\neg\pi)$
- $\langle \overline{\mathrm{B}} \rangle \psi = (\neg\pi)\ T\ \psi$
- $\langle E \rangle \psi = (\neg\pi)\ C\ \psi$
- $\langle \overline{\mathrm{E}} \rangle \psi = \psi\ D\ (\neg\pi)$
- $\langle A \rangle \psi = (\neg\pi \wedge \psi)\ T\ \top$
- $\langle \overline{\mathrm{A}} \rangle \psi = (\neg\pi \wedge \psi)\ D\ \top$

Since $\mathcal{HS}$ logic is undecidable, CDT and BCDT$^+$ are also undecidable. For instance, CDT is undecidable over almost all interesting classes of linear orderings, e.g., $\mathbb{N}, \mathbb{Z}$ [88].

3.5.4 Propositional Neighbourhood Logic (PNL)

PNL [101] is nearly a maximal decidable fragment [102] of $\mathcal{HS}$ logic which consists of two temporal modalities ($\langle A \rangle, \langle \overline{A} \rangle$).

The language of PNL consists of a set $\mathcal{AP}$ of atomic propositions (e.g., p), logical operator, *or* ($\vee$) and *negation* ($\neg$) and two temporal operators *meet* ($\Diamond_r$) and *met-by* ($\Diamond_l$) [103]. The formulas of PNL are generated by the following syntax rules [101, 102].

$$\psi ::= p \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \Diamond_r \psi \mid \Diamond_l \psi$$

Note that in some of the literature $\langle A \rangle$ (resp. $\langle \overline{\mathrm{A}} \rangle$) is used to model the meet (resp. met-by) relation between intervals in the context of strict interval

structures while $\Diamond_r$ ($\Diamond_l$) is used in the context of non-strict interval structures. In this thesis, we use these notations interchangeably.

Two modalities $\Box_r$ and $\Box_l$ are respectively defined as the dual operators of $\Diamond_r$ and $\Diamond_l$, i.e., $\Box_r\psi = \neg\Diamond_r\neg\psi$ and $\Box_l\psi = \neg\Diamond_l\neg\psi$. Also, two propositional constants *true* ($\top$) and *false* ($\bot$) can be defined as usual.

Given a model M (which is the same as a model for a $\mathcal{HS}$ formula), the semantics of the logic is defined as follows [101, 102, 104].

- $M, [i, j] \models p$ iff $p \in V([i, j])$, for any $p \in \mathcal{AP}$;
- $M, [i, j] \models \neg\psi$ iff it is not the case that $M, [i, j] \models \psi$;
- $M, [i, j] \models \psi_1 \vee \varphi_2$ iff $M, [i, j] \models \psi_1$ *or* $M, [i, j] \models \psi_2$;
- $M, [i, j] \models \psi_1 \wedge \varphi_2$ iff $M, [i, j] \models \psi_1$ *and* $M, [i, j] \models \psi_2$;
- $M, [i, j] \models \Diamond_r\psi$ iff there exists $h \geq j$ such that $M, [j, h] \models \psi$;
- $M, [i, j] \models \Diamond_l\psi$ iff there exists $h \leq i$ such that $M, [h, i] \models \psi$;
- $M, [i, j] \models \Box_r\psi$ iff for every $h \geq j$ we have $M, [j, h] \models \psi$;
- $M, [i, j] \models \Box_l\psi$ iff for every $h \leq i$ we have $M, [h, i] \models \psi$.

To have a better intuition about the usage of PNL, suppose, we want to encode the statement *Right after John completes the writing of his thesis, he has to submit it to the graduate office*, where we know that *John is completing the writing of his thesis now*. The formula $\psi = \varphi_1 \wedge \Diamond_r\varphi_2$ describes the statement: φ_1 is *John is completing the writing of his thesis* and φ_2 is *John submits his thesis to the graduate office*.

It is worthwhile mentioning that in the context of non-strict (resp. strict) interval structures, PNL is called PNL$^+$ (resp. PNL$^-$). PNL$^+$ can be extended

with a propositional constant, π, in order to specify a point interval, i.e., an interval whose end points coincide. The satisfiability of this constant over an interval is defined as follows:

- $M, [i, j] \models \pi$ iff $i = j$.

3.5.4.1 A Tableau-based algorithm for PNL$^+$

The content of this section is essentially the same as Section 9 in [101]. In this section we work in the context of a non-strict interval structure and present an algorithm to check the satisfiability of a PNL$^+$ formula. The algorithm can be modified in order to obtain an algorithm for checking the satisfiability of a PNL$^-$ formula. We refer the reader to [101] to see the necessary modifications.

Definition 10. We recall some basic definitions:

- A *finite tree* is a finite directed connected graph in which every node (except the *root*) has exactly one incoming edge.
- A *successor* of a node $\mathbf{n}$ is a node $\mathbf{n}'$ s.t. there is an edge from $\mathbf{n}$ to $\mathbf{n}'$.
- A *leaf* is a node which has no successor.
- A *path* is a sequence of nodes $\mathbf{n}_0$,...,$\mathbf{n}_k$ such that, for all $i = 0, ..., k-1$, $\mathbf{n}_{i+1}$ is a successor of $\mathbf{n}_i$.
- A *branch* is a path from the root to a leaf.
- The *height* of a node $\mathbf{n}$ is the maximum number of edges of a path from $\mathbf{n}$ to a leaf. If $\mathbf{n}$, $\mathbf{n}'$ belong to the same branch and the height of $\mathbf{n}$ is less than or equal to the height of $\mathbf{n}'$, we write $\mathbf{n} \prec \mathbf{n}'$. We remark that we follow [105] in using a nonstandard definition of height.

Definition 11. Let $\mathbb{C} = \langle C, < \rangle$ be a finite partial order.

- A *labeled formula*, with label in $\mathbb{C}$, is a pair $(\psi,[c_i, c_j])$, where $\psi \in \text{PNL}^+$ and $[c_i, c_j] \in \mathbb{I}(C)^-$ and $c_i < c_j$.
- The *decoration* $v(\mathbf{n})$ for a node $\mathbf{n}$ in a tree $\mathcal{T}$ is a triple $((\psi, [c_i, c_j]),\mathbb{C},u_n)$, where $(\psi, [c_i, c_j])$ is a labeled formula, with label in $\mathbb{C}$, and u_n is a local flag function which associates a value, 0 or 1, with each branch B containing $\mathbf{n}$.
- A *decorated tree* is a tree in which every node has a decoration $v(\mathbf{n})$.

The formula in $v(\mathbf{n})$ is denoted by $\Phi(\mathbf{n})$. Note that the value 0 for a node $\mathbf{n}$ with respect to a branch B indicates that $\mathbf{n}$ can be expanded on B. For every decorated tree, we define a *global flag function* u acting on pairs (node, branch through that node) as $u(\mathbf{n}, B) = u_n(B)$. Sometimes, for convenience, we will include in the decoration of the nodes the global flag function instead of the local ones.

If B be a branch in a tree, $\mathbb{C}_{\mathcal{B}}$ is the (partially) ordered set in the decoration of the leaf of B. Also, $B.\mathbf{n}$ denotes the result of the expansion of B with the node $\mathbf{n}$ (i.e., the addition of an edge connecting the leaf of B to $\mathbf{n}$). Similarly, $B.\mathbf{n}_1.\mathbf{n}_2$ denotes the expansion of the branch $B.\mathbf{n}_1$ with $\mathbf{n}_2$. Also, $B.\mathbf{n}_1|...|\ \mathbf{n}_k$ denotes the result of the expansion of B with k direct successor nodes $\mathbf{n}_1$... $\mathbf{n}_k$. A tableau for a set of PNL^+ formulas is a special decorated tree.

Definition 12. Given a *decorated tree* $\mathcal{T}$, a branch B in $\mathcal{T}$, and a node $\mathbf{n} \in B$ such that $v(\mathbf{n})$ =$((\psi, [c_i, c_j]),\mathbb{C}_{\mathcal{B}},u\ )$, with $u(\mathbf{n},B) = 0$, the *expansion rule* for B and $\mathbf{n}$ is defined as follows (in all the cases considered, $u(\mathbf{n}^{'}, B^{'}) = 0$ for all new pairs $(\mathbf{n}^{'}, B^{'})$ of nodes and branches).

$R_{\neg\neg}$: If $\psi = \neg\neg\varphi$, then expand the branch to $B.\mathbf{n}_1$, with $v(\mathbf{n}_1) = ((\varphi,[c_i, c_j]), \mathbb{C}_{\mathcal{B}}, u)$;

$R_{\wedge}$: If $\psi = \varphi_0 \wedge \varphi_1$, then expand the branch to $B.\mathbf{n}_0.\mathbf{n}_1$, with $v(\mathbf{n}_0) = ((\varphi_0,[c_i, c_j]), \mathbb{C}_{\mathcal{B}}, u)$ and $v(\mathbf{n}_1) = ((\varphi_1,[c_i, c_j]), \mathbb{C}_{\mathcal{B}}, u)$;

$R_{\vee}$: If $\psi = \varphi_0 \vee \varphi_1$, then expand the branch to $B.\mathbf{n}_0|\mathbf{n}_1$, with $v(\mathbf{n}_0) = ((\varphi_0,[c_i, c_j]), \mathbb{C}_{\mathcal{B}}, u)$ and $v(\mathbf{n}_1) = ((\varphi_1,[c_i, c_j]), \mathbb{C}_{\mathcal{B}}, u)$;

$R_{\Box_r}$: If $\psi = \Box_r\varphi$ and c is the least element of $\mathbb{C}_{\mathcal{B}}$ such that $c_j \leq c$ and c has not beed used yet to expand $\mathbf{n}$ on B, then expand the branch to $B.\mathbf{n}_1$, with $v(\mathbf{n}_1) = ((\varphi,[c_j, c]), \mathbb{C}_{\mathcal{B}}, u)$;

$R_{\Box_l}$: If $\psi = \Box_l\varphi$ and c is the greatest element of $\mathbb{C}_{\mathcal{B}}$ such that $c \leq c_i$ and c has not beed used yet to expand $\mathbf{n}$ on B, then expand the branch to $B.\mathbf{n}_1$, with $v(\mathbf{n}_1) = ((\varphi,[c, c_i]), \mathbb{C}_{\mathcal{B}}, u)$;

$R_{\Diamond_r}$: If $\psi = \Diamond_r\varphi$, then expand the branch to $B.\mathbf{n}_j|\ldots\mathbf{n}_n|\mathbf{n}'_j|\ldots|\mathbf{n}'_n$, where

1. for all $j \leq k \leq n$, $v(\mathbf{n}_k) = ((\varphi,[c_j, c_k]), \mathbb{C}_{\mathcal{B}}, u)$ and
2. for all $j \leq k \leq n$, $v(\mathbf{n}'_k) = ((\varphi,[c_j, c]), \mathbb{C}_k, u)$ where, for $j \leq k \leq n-1$, $\mathbb{C}_k$ is the linear ordering obtained by inserting a new element c between c_k and c_{k+1} in $\mathbb{C}_{\mathcal{B}}$, and for $k = n$, $\mathbb{C}_k$ is the linear ordering obtained by inserting a new element c after c_n in $\mathbb{C}_{\mathcal{B}}$.

$R_{\Diamond_l}$: If $\psi = \Diamond_l\varphi$, then expand the branch to $B.\mathbf{n}_1|\ldots\mathbf{n}_i|\mathbf{n}'_1|\ldots|\mathbf{n}'_i$, where

1. for all $1 \leq k \leq i$, $v(\mathbf{n}_k) = ((\varphi,[c_k, c_i]), \mathbb{C}_{\mathcal{B}}, u)$ and
2. for all $1 \leq k \leq i$, $v(\mathbf{n}'_k) = ((\varphi,[c, c_i]), \mathbb{C}_k, u)$ where, for $2 \leq k \leq i$, $\mathbb{C}_k$ is the linear ordering obtained by inserting a new element c between c_{k-1} and c_k in $\mathbb{C}_{\mathcal{B}}$, and for $k = 1$, $\mathbb{C}_1$ is the linear ordering obtained by inserting a new element c before c_1 in $\mathbb{C}_{\mathcal{B}}$.

Note that for any node $m(\neq n)$ in B and any branch $B^{'}$ extending B, let $u(m, B^{'})$ be equal to $u(m, B)$, and for any branch $B^{'}$ extending B, $u(n, B^{'}) = 1$, unless $\psi = \Box_l\varphi$ or $\psi = \Box_r\varphi$ (in such cases $u(n, B^{'}) = 0$). The universal formula $\Box_r\varphi$ (resp. $\Box_l\varphi$) states that, for all $c_j \leq c$ (resp. $c \leq c_i$), φ holds over $[c_j, c]$ (resp. $[c, c_i]$). As a matter of fact, the expansion rule imposes such a condition for a single element c in $\mathbb{C}_{\mathcal{B}}$ (the least (resp. greatest) element which has not been used yet), and it does not change the flag (which remains equal to 0). In this way, all elements will be eventually taken into consideration, including those elements greater (resp. smaller) than c_j (resp. c_i) that will be added to $\mathbb{C}_{\mathcal{B}}$ in some subsequent steps of the tableau construction.

Definition 13. A node $\mathbf{n}$ in a decorated tree $\mathcal{T}$ is *available on a branch* B it belongs to iff $u(\mathbf{n}, B)$=0.

An expansion rule is applicable to a node $\mathbf{n}$ on a branch B if the node is available on B and the application of the rule generates at least one successor node with a new labeled formula. This second condition is needed to avoid looping of the application of the rule on universal formulas ($\Box_r\varphi$, $\Box_l\varphi$).

Definition 14. A branch B is *closed* if the following condition holds:.

$*$ There are two nodes $\mathbf{n}$, $\mathbf{n}^{'} \in B$ such that $v(\mathbf{n}) = ((\varphi,[c_i, c_j]), \mathbb{C}, u)$ and $v(\mathbf{n}^{'}) = ((\neg\varphi,[c_i, c_j]),\mathbb{C}^{'}, u)$ for some formula φ and $c_i, c_j \in \mathbb{C} \cap \mathbb{C}^{'}$;

If the above condition does not hold, the branch is *open*, which means that there is no inconsistency between the labeled formulas residing on the branch and we are able to build a class of models based on the labeled formulas.

Definition 15. For a branch B in a decorated tree $\mathcal{T}$, the *expansion strategy* is defined as follows:

1. Apply the expansion rule to a branch B only if it is open;

2. If B is open, apply the expansion rule to the closest available node to the root node (say $\mathbf{n}$) in B to which the expansion rule is applicable (if any).

Definition 16. An *initial tableau* for a given formula $\psi \in \text{PNL}^+$ is the finite decorated tree $\mathcal{T}$ shown below.

$$[root]\ ((\psi,[c_0, c_1]),\ \{c_0 < c_1\},0)$$

Definition 17. A *tableau* for a given formula $\psi \in \text{PNL}^+$ is a finite decorated tree $\mathcal{T}$ obtained by expanding the initial tableau for ψ, through successive applications of the expansion strategy to the existing branches.

Definition 18. A tableau for a formula in PNL^+ is *closed* if and only if every branch in it is closed, otherwise it is *open*.

Example 3. Figure 3.6 exhibits an example of a tableau for $\psi = \Diamond_r \Box_r p \wedge \Diamond_r \neg p$.

Note that in order to make the tableau more understandable to the reader, we have provided two descriptive elements in the tableau: a number and text surrounded by a rectangle for each node and each transition respectively; e.g., a transition label $\boxed{3\text{-}\mathcal{R}_{\Diamond_r}}$ states that the following node of the transition is produced by applying $\mathcal{R}_{\Diamond_r}$ to the node 3.

3.5.5 Metric Propositional Neighbourhood Logic (MPNL)

There is an interesting family of extensions of PNL with metricity over the natural numbers, called *Metric Propositional Neighbourhood Logics* (MPNL). Figure 3.7 exhibits the members of this family along with the expressivity relation between them.

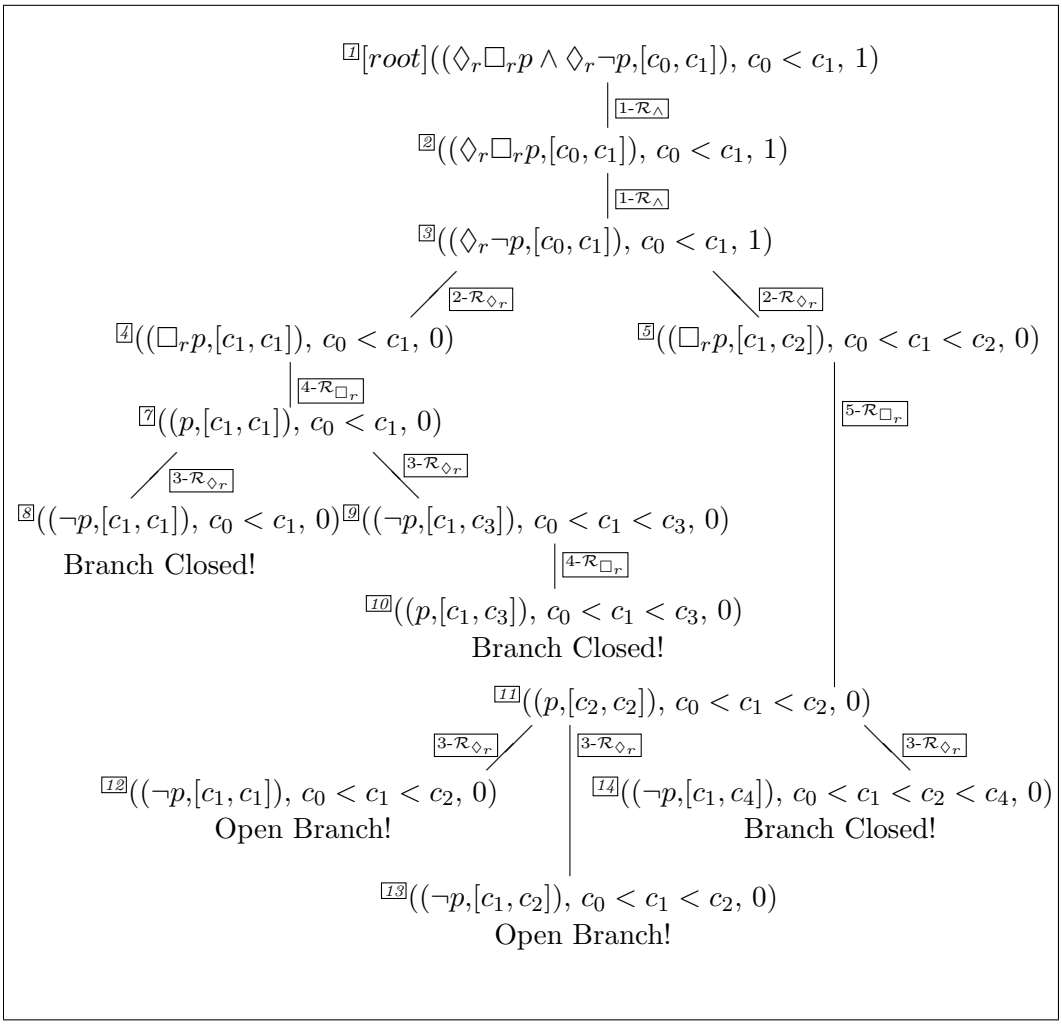

Figure 3.6: Tableau for $\psi = \Diamond_r \Box_r p \wedge \Diamond_r \neg p$

As seen in Figure 3.7, MPNL$_l$, Propositional Neighbourhood Logic, with atomic length constraints, is the most expressive logic [11] in the family. The language of MPNL$_l$ includes all syntax of PNL and is enriched as follows [12, 106].

- $\psi ::= len_{Ck}$, $k \in \mathbb{N}$, $C \in \{\leq, <, =, >, \geq\}$.

The semantics of the aforementioned case is defined as follows [107].

- $M, [i, j] \models len_{Ck}$ iff $|j - i|$ C k.

This logic is a metric logic, in the sense that a user can specify the length of an interval on which a proposition should be true. For example, a formula $\psi = (p \wedge len_{=k})$ is true when p is true and the length of the interval on which we evaluate the formula is k. Indeed, this is a crucial feature because in some domains (e.g., medicine), it is necessary to determine the length of an event. Assume *John needs to take B12 tablets for 2 weeks*; thus, in order to define this statement in MPNL$_l$, the user has to use the *len* operator to model the duration of 2 *weeks*; obviously, other operators of the logic are not able to restrict the length of an interval.

In terms of expressivity, MPNL$_l$ is able to model a metric version of all Allen's relations with the exception of the *during* relation [12].

In the next chapter, we introduce a new interval-based temporal logic, inspired by MPNL$_l$.

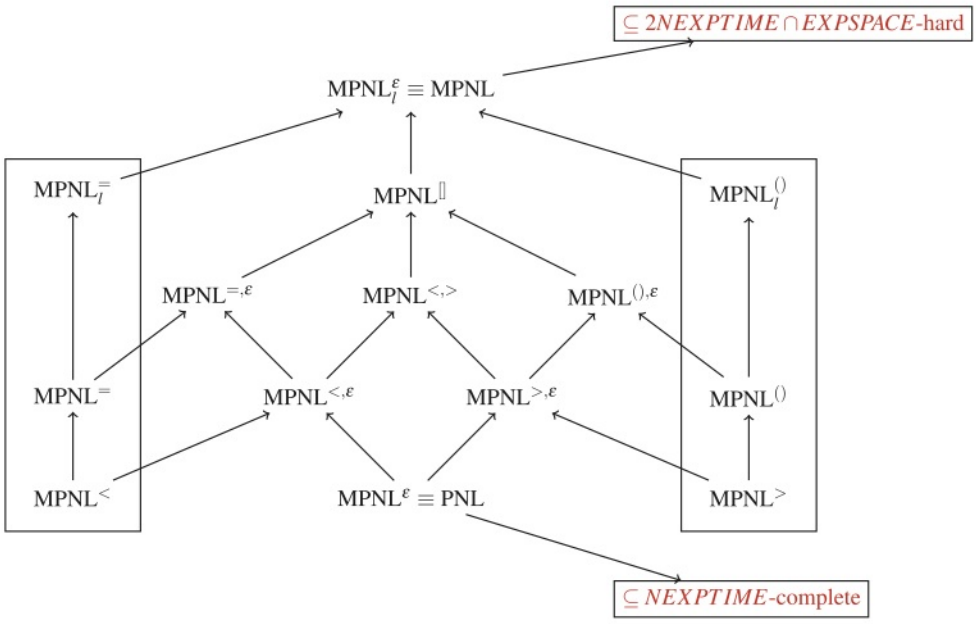

Figure 3.7: MPNL family and their expressivity [107]

Chapter 4

IMPNL: A Logic Inspired by MPNL$_l$

In this thesis, we combine an interval-based temporal logic and a description logic in order to design a interval-based temporal description logic. MPNL$_l$ is not a good candidate for this goal; because if we combine MPNL$_l$ with a DL, the combined logic is undecidable [108]. In this chapter, we introduce a new logic, named IMPNL, inspired by MPNL$_l$. Then, we design a sound and complete tableau-based algorithm for checking the satisfiability of IMPNL formulas. We show that this algorithm always terminates. Therefore, the satisfiability problem of an IMPNL formula is decidable. We propose a non-deterministic implementation for the algorithm which has PSpace complexity. In the next chapter, we combine IMPNL (without its negation operator) with the description logic $\mathcal{ALC}$ and discuss its properties.

4.1 IMPNL

The language of IMPNL consists of a set, $\mathcal{AP}$, of propositional variables, logical operators *atomic negation* ($\neg$), *or* ($\vee$), *and* ($\wedge$), and temporal operators, $\Diamond_r$, $\Diamond_l$, corresponding to Allen's relations *meet* and its inverse, *met-by*. This logic has two constants $\top$ (True) and $\bot$ (False) defined as usual.

4.1.1 Syntax and Semantics

The formulas, denoted by φ, ψ, ..., are recursively defined using BNF, where p_k is a propositional variable. Note, the subscript of an atomic formula denotes the length whereas, the subscript of a non-atomic formula denotes an index and is used to distinguish the formula from other formulas. We sometimes use $p_k \rightarrow \psi$ to denote $\neg p_k \vee \psi$.

$$\psi = p_{_k} \mid \neg p_{_k} \mid \top_{_k} \mid \bot_{_k} \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid \Diamond_r \psi \mid \Diamond_l \psi$$
$$\text{where} \qquad k \in \mathbb{N}$$

Note that in IMPNL, a user must specify the length of an interval as a subscript of a propositional variable. Therefore, it is not possible to have a propositional variable (without specifying the length) as an atomic formula.

The semantics of IMPNL is based on a 3 tuple structure $M = \langle \mathbb{D}, \mathbb{I}^-(\mathbb{D}), \mathrm{V} \rangle$ where the pair $\langle \mathbb{D}, \mathbb{I}^-(\mathbb{D}) \rangle$ is a *strict interval structure* (see Definition 7 on Page 32) and V: $\mathbb{I}^-(\mathbb{D}) \rightarrow 2^{\mathcal{AP}}$ is a valuation function that assigns to every interval the set of propositional variable which are true on that interval. A *satisfaction relation* is defined as follows:

1. $M, [i,j] \models p_{_k}$ iff $j - i = k$ *and* $\forall i', j'$, if $[i', j'] \subseteq [i,j]$ then $p_{k'} \in V([i', j'])$ where $j' - i' = k'$;

2. $M, [i,j] \models \neg p_k$ iff $j - i = k$ *and* $\forall i', j'$, if $[i', j'] \subseteq [i,j]$ then $p_{k'} \notin V([i', j'])$ where $j' - i' = k'$;

3. $M, [i,j] \models \top_k$ iff $j - i = k$;

4. $M, [i,j] \models \bot_k$ never;

5. $M, [i,j] \models \psi_1 \vee \psi_2$ iff $M, [i,j] \models \psi_1$ *or* $M, [i,j] \models \psi_2$;

6. $M, [i,j] \models \psi_1 \wedge \psi_2$ iff $M, [i,j] \models \psi_1$ *and* $M, [i,j] \models \psi_2$;

7. $M, [i,j] \models \Diamond_r \psi$ iff there exists $h > j$ such that $M, [j,h] \models \psi$;

8. $M, [i,j] \models \Diamond_l \psi$ iff there exists $h < i$ such that $M, [h,i] \models \psi$.

We say a formula ψ is satisfiable if there exists a structure M and an interval $[c_0,c_1]$ s.t. $M, [c_0, c_1] \models \psi$. Also, it is easy show that $\Diamond_z(\psi_1 \vee \psi_2) \Leftrightarrow \Diamond_z \psi_1 \vee \Diamond_z \psi_2$ $(z \in \{r, l\})$ and $\Diamond_z(\psi_1 \wedge \psi_2) \rightarrow \Diamond_z \psi_1 \wedge \Diamond_z \psi_2$ $(z \in \{r, l\})$.

Consider the differences between lMPNL and MPNL$_l$:

- IMPNL has negation only for atomic formulas.
- IMPNL has no $\Box_z$ $(z \in \{r, l\})$ (necessity) operator.
- In IMPNL, the length of every atomic proposition must be specified.
- IMPNL has a homogeneity assumption; i.e., if a formula is true (false) on an interval, it is true (false) in every subinterval of that interval.
- IMPNL has a tableau algorithm to support analysis, but to our knowledge, no tableau algorithm has been designed for MPNL$_l$.
- Checking the satisfiability of an IMPNL formula has PSPACE complexity.

4.1.2 Restrictions of IMPNL

We have some restrictions in IMPNL: it is not possible to model every natural language requirement, but sometimes we can provide a reasonable alternative, as we show below:

1. The "At least" condition, e.g., "The patient should give at least 3 sputum during the diagnosis".

   In this case we are not able to model the statement with a formula of finite length, because an upper bound is unspecified. Some cases can be fixed by making suitable assumptions because in the domain of medicine, most events have a maximum length (worst case: lifetime of a patient, e.g., 120 years). Therefore, it is frequently easy to find an upper bound for situations governed by "at least"; e.g., in the above example, we can assume that the patient gives at most 5 sputum.

2. Statements using "any", e.g., "The patient should fast for 12 hours before any blood work, which includes testing fasting sugar".

   There is no general solution for this case but we may find a solution for some particular cases; e.g., Kaletra and Rifampin are contraindicated and should not be taken simultaneously, in the sense that a patient is not allowed to take Kaletra during the period in which (s)he takes Rifampin. We are not able to say there is no interval during which the patient takes these medicines simultaneously. Therefore, we model this statement in another way, i.e., we say the patient takes neither Kaletra nor Rifampin or (s)he takes Kaletra but not Rifampin or vice versa: $\bigwedge_{k=1hour}^{z}[(\neg TakingKaletra_k \wedge \neg TakingRifampin_k) \vee (TakingKaletra_k \wedge \neg TakingRifampin_k) \vee (\neg TakingKaletra_k \wedge TakingRifampin_k)]$ where z is a natural number constant and is equal to the number of hours that

exists in 120 years.

3. A loop with an undetermined number of repetitions, e.g., one which uses "Until".

   Generally, we are not able to model this kind of loop. In some cases we can make suitable assumptions and model the guideline, but it is not always possible. For example, the CD4 level of an HIV-infected patient should be monitored until his death. The time of death is unknown so we assume that the patient will live for 120 years (maximum lifetime of a person), and model it with $HIVPatient_{1hour} \rightarrow \Diamond_l \Diamond_r \text{CD4Monitoring}_{120years}$.

4.1.3 Tableau-based algorithm for IMPNL formulas

In this section we present a tableau-based algorithm for the satisfiability checking of a formula of IMPNL. This algorithm is a modification of the algorithm which was introduced in [109]. In the algorithm, we have changed the strategy for selecting an interval required for satisfying $\Diamond_z^{z'}\varphi$ $(z \in \{r, l\}, z' \in \{+, -\})$; as a consequence, some of the expansion rules (rules for $\Diamond_z^{z'}$ $(z \in \{r, l\}, z' \in \{+, -\})$, in Definition 5 in [109]) are slightly different here in Definition 23 (on Page 66). Due to this change, we are able to apply the expansion rules in Definition 27 in an arbitrary order (on Page 68): in [109], we needed to apply the rules in a specific order. Also, in this algorithm, we have removed the notion of decoration and the notion of local flag function.

There are some tableau algorithms that use the negation of a formula to construct a tableau, while other tableau algorithms use the original formula to create the tableau. In our algorithm, we use an annotated version of the original formula to derive the tableau. The annotated version of the formula

is equi-satisfiable with the original formula (Lemma 1). The root node of the tableau is created based on Definition 28 (on Page 68). Then, the expansion strategy, defined in Definition 27 (on Page 68), indicates how the tableau rules (Definition 23 on Page 66) should be used to derive the tableau. If one of the fully expanded branches is open (Definition 25 on Page 67), the formula is satisfiable (Theorem 4 on Page 73). If all branches are closed, the formula is not satisfiable. We begin by introducing some definitions and functions needed in the tableau rules.

4.1.3.1 Annotation of Input Formula

In this section, we annotate occurrences of $\Diamond_z$ ($z \in \{r, l\}$) in a (sub) formula $\Diamond_z \varphi$ to indicate to what extent the length of the interval required for the satisfaction of φ is known. We first need to make the following definition.

Definition 19. In a formula, an occurrence of a propositional variable is called a *free occurrence* if it is not bound by any temporal operators.

Example 4. In the formula $p_3 \vee (q_8 \wedge \Diamond_r(r_7 \vee p_3))$, the first occurrence of propositional variable p_3 and the only occurrence of propositional variable q_8 are free occurrences.

Generally, the operator $\Diamond_z$ ($z \in \{r, l\}$) can appear in a formula ψ in three different ways.

I. The length of the interval required for the satisfaction of the formula following $\Diamond_z$ is known.

   For example, in the formula $\Diamond_r(p_k \wedge \Diamond_r q_m)$, the length of an interval required for the satisfaction of $p_k \wedge \Diamond_r q_m$ is k. In other words, there is an interval with length k right after the current interval on which $p_k \wedge \Diamond_r q_m$ is true.

II. The length of the interval required for the satisfaction of the formula following $\Diamond_z$ is not yet determined.
For example, in the formula $\Diamond_r(\Diamond_r p_k)$, the length of an interval needed for the satisfaction of $\Diamond_r p_k$ is not yet determined.

III. The combination of the two previous cases, which means that the length of the interval required for satisfaction of part of the formula following $\Diamond_z$ is known and is not yet determined for the other part.
For example, in the formula $\Diamond_r(p_k \vee \Diamond_r q_m)$, the length of the interval required for the satisfaction of p_k is known, but it is not yet determined for the satisfaction of $\Diamond_r q_m$, and we only need to satisfy one of them.

Based on the aforementioned cases, we annotate a formula ψ so that if ψ has a subformula $\Diamond_z\varphi$ of Form I, it will be changed to $\Diamond_z^*\varphi$; if it has a subformula $\Diamond_z\varphi$ of Form II, it will be changed to $\Diamond_z^-\varphi$, and if it has a subformula $\Diamond_z\varphi$ of Form III, it will be changed to $\Diamond_z^+\varphi$, using the following steps. Let ψ be a formula with at least one subformula of the form $\Diamond_z\varphi$.

1. If φ is an atomic formula, change all instances of $\Diamond_z\varphi$ in ψ to $\Diamond_z^*\varphi$.

2. If $\varphi = \varphi_1 \wedge \varphi_2$ and φ has at least one free occurrence of any propositional variable, change all instances of $\Diamond_z\varphi$ in ψ to $\Diamond_z^*\varphi$.

3. If $\varphi = \varphi_1 \vee \varphi_2$ and both φ_1 and φ_2 have at least one free occurrence of any propositional variable, change all instances of $\Diamond_z\varphi$ in ψ to $\Diamond_z^*\varphi$.

4. If $\varphi = \varphi_1 \vee \varphi_2$ and (wlog) φ_1 has at least one free occurrence of any propositional variable and φ_2 has no free occurrence of any propositional variable, change all instances of $\Diamond_z\varphi$ in ψ to $\Diamond_z^+\varphi$.

5. If none of the above rules is applicable to $\Diamond_z\varphi$ in ψ, change it to $\Diamond_z^-\varphi$.

Let IMPNL^a denote the set of annotated versions of formulas built using the syntax of IMPNL.

Example 5. Let $\psi = \Diamond_r(p_3 \wedge \Diamond_r((\Diamond_r q_1 \vee \Diamond_r r_7) \vee p_4)) \vee \Diamond_l(\Diamond_r p_9 \wedge \Diamond_l q_2)$. The annotated version of ψ is $\Diamond_r^*(p_3 \wedge \Diamond_r^+((\Diamond_r^* q_1 \vee \Diamond_r^* r_7) \vee p_4)) \vee \Diamond_l^-(\Diamond_r^* p_9 \wedge \Diamond_l^* q_2)$. The annotating steps are as follows.

Step 1. Based on rule 1, we have, $\Diamond_r(p_3 \wedge \Diamond_r((\Diamond_r^* q_1 \vee \Diamond_r^* r_7) \vee p_4)) \vee \Diamond_l(\Diamond_r^* p_9 \wedge \Diamond_l^* q_2)$.

Step 2. Since rule 1 is no longer applicable, we use rule 2.

– Rule 2 is not applicable on $\Diamond_l(\Diamond_r^* p_9 \wedge \Diamond_l^* q_2)$, since both operands of the $\wedge$-operator have no free occurrence.

– The occurrence p_3 is a free occurrence in $\Diamond_r(p_3 \wedge \Diamond_r((\Diamond_r^* q_1 \vee \Diamond_r^* r_7) \vee p_4))$. Because one of the operands of the $\wedge$-operator is a free occurrence, we change the subformula to $\Diamond_r^*(p_3 \wedge \Diamond_r((\Diamond_r^* q_1 \vee \Diamond_r^* r_7) \vee p_4))$.

Step 3. Since rule 2 is no longer applicable, we use rule 3. In the subformula $\Diamond_r((\Diamond_r^* q_1 \vee \Diamond_r^* r_7) \vee p_4)$, the occurrence p_4 is a free occurrence, and the subformula $(\Diamond_r^* q_1 \vee \Diamond_r^* r_7)$ contains no free occurrences, therefore rule 3 is not applicable here, and the result of applying rule 4 is $\Diamond_r^+((\Diamond_r^* q_1 \vee \Diamond_r^* r_7) \vee p_4)$.

Step 4. Based on the annotation rules (rule 5), the unchanged cases of $\Diamond_z$ must be changed to $\Diamond_z^-$. Finally, the annotated version of the formula is $\Diamond_r^*(p_3 \wedge \Diamond_r^+((\Diamond_r^* q_1 \vee \Diamond_r^* r_7) \vee p_4)) \vee \Diamond_l^-(\Diamond_r^* p_9 \wedge \Diamond_l^* q_2)$.

Lemma 1. A formula ψ and the annotated version of ψ are equi-satisfiable.

Proof. Each of the annotations $\{*, +, -\}$ is syntactic sugar, so, if ψ is satisfiable, then the annotated version of ψ is satisfiable and vice versa. □

4.1.3.2 A Finder Function (FF)

Let $2^{\mathbb{N}}_{\mathrm{O}}$ denote the set of non descending ordered lists built from the subsets of $\mathbb{N}$. Candidates for the length of the interval for the satisfaction of a formula are determined using a function $FF : \mathrm{IMPNL}^a \to 2^{\mathbb{N}}_{\mathrm{O}}$, which assigns to every formula, a non descending ordered list of interval lengths. The formula itself provides constraints that exclude most lengths from being candidates for the interval required for its satisfaction. The following example explains the idea.

Example 6. Suppose we want to determine if a formula is satisfiable, e.g., the formula $\psi = p_k \wedge \Diamond_r(q_m \vee s_n)$ $(m \neq n)$. To do so, we need an interval. This interval is called the *current interval*; i.e., given $M, [i, j] \models \psi$, the interval $[i, j]$ is the current interval for ψ. Trivially, we could test all possible intervals, but this is not pragmatic; we require a function in order to find a candidate length for an interval that makes the formula satisfiable, before we begin the satisfiability checking. Intuitively, an interval $[c_0, c_1]$ with length k is a good choice here, since p_k could be true on it. If $p_{k'} \in V[c_0^{'}, c_1^{'}]$ where $k^{'} = c_1^{'} - c_0^{'}$ for all $[c_0^{'}, c_1^{'}] \subseteq [c_0, c_1]$, we construct a model M, s.t. $M, [c_0, c_1] \models p_k$. We could test intervals with lengths $\neq k$ but none of them would satisfy p_k.

Based on rule 4 of the semantics, in order to satisfy ψ, we should have $M, [c_0, c_1] \models p_k$ and $M, [c_0, c_1] \models \Diamond_r(q_m \vee s_n)$. The latter case is satisfied if we find an h s.t. $M, [c_1, h] \models (q_m \vee s_n)$. We can test all different intervals starting at c_1 to find the suitable interval to check the satisfiability of $(q_m \vee s_n)$. Obviously, this is not a reasonable way forward; we need a function to find a suitable h for us. Equivalently, the function finds the length of $[c_1, h]$ in order

to determine h. Since in order to satisfy q_m (resp. s_n) we need an interval with length m (resp. n), the function should return a list of possible lengths which, for this example, is $\langle m, n\rangle$ (wlog $m \leq n$).

The function FF is defined below. We use $FF^i(\psi)$ to denote the i^{th} element of $FF(\psi)$. The $\circledast$ operator creates a non descending ordered list by merging two non descending ordered lists. We note that for the $\wedge$-operator, using the lengths of both operands does not change the final result of satisfiability checking; however, using the length of one operand is enough since the length of the two operands must be equal; otherwise the formula is unsatisfiable.

1. $FF(\psi)$=$\langle$k$\rangle$ *if* $\psi \in \{p_k, \neg p_k, \top_k, \bot_k\}$
2. $FF(\psi)$=$\langle\rangle$ *if* $\psi = \Diamond_z^{z'} \varphi$ and $z \in \{r, l\}$ and $z' \in \{*, -, +\}$
3. $FF(\psi)$=$FF(\varphi_1) \circledast FF(\varphi_2)$ *if* $\psi = \varphi_1\ \omega\ \varphi_2$ where $\omega \in \{\vee, \wedge\}$

4.1.3.3 Tableau Construction.

The tableaux idea and proof of its soundness are in many ways analogous to [101, 105]. We have adapted some steps and some details of the proof in [105], and have augmented them with metricity-related details.

Recall Definition 10. Let C be a set, and a distance function f ($f : C \times C \to \mathbb{N}; f(i, j) = |j - i|$ where $i, j \in C$) can be defined on the set, and let $L_{\mathbb{C}}$ be a ternary relation, and $L_{\mathbb{C}} \subseteq C \times C \times \mathbb{N}$, consisting of tuples (i, j, k) where $[i, j]$ is a strict interval and $k = f(i, j)$. This relation is used in the following definition to keep track of the lengths of the intervals which are considered in the construction of a tableau.

Definition 20. Let $\mathbb{C} = \langle C, <, L_{\mathbb{C}}\rangle$ be a finite partial order equipped with the $L_{\mathbb{C}}$ relation.

- A *labeled formula*, with label in $\mathbb{C}$, is a pair $(\psi,\ [c_i, c_j])$, where $\psi \in$ IMPNL^a and $[c_i, c_j] \in \mathbb{I}^-(C)$ and $c_i < c_j$ and $(c_i, c_j, |c_j - c_i|) \in L_{\mathbb{C}}$.
- A node $\mathbf{n}$ in a tree $\mathcal{T}$ is a pair $((\psi, [c_i, c_j]),\ \mathbb{C})$, where $(\psi, [c_i, c_j])$ is a labeled formula, with label in $\mathbb{C}$.
- For any branch B in a tree, $\mathbb{C}_{\mathcal{B}}$ is the partially ordered set in the leaf of B.

Example 7. $\mathbb{C} = (\{c_0, c_1, c_2, c_3\}, \{c_0 < c_1 < c_2 < c_3\}, \{(c_1, c_2, 4), (c_0, c_3, 12), (c_2, c_3, 8)\})$.

Note that the main ontological element in IMPNL is an interval, and we are not able to identify a time point in the logic. Therefore the closedness or openness of intervals in $\mathbb{I}^-(C)$ is not important, and has no effect on the satisfiability of any formula.

Henceforth, we use a compact representation of $\mathbb{C}$ in the sense that we just mention the last two components (which are sets) of the 3 tuple $\mathbb{C}$. It is easy for the reader to find the first component based on the other two components.

Definition 21. Let $\mathrm{N}_{\mathcal{T}}$ and $\mathrm{B}_{\mathcal{T}}$, respectively, be the set of nodes and the set of branches of a tree $\mathcal{T}$. A *flag function* $u : \mathrm{N}_{\mathcal{T}} \times \mathrm{B}_{\mathcal{T}} \to \{0, 1\}$ is a function which assigns a value 0 to a pair ($\mathbf{n}$,B) consisting of a node $\mathbf{n}$ and a branch B through $\mathbf{n}$, if $\mathbf{n}$ is expandable (i.e., an expansion rule (Definition 23) can be applied to $\mathbf{n}$) w.r.t. branch B, and assigns a value 1 to the pair otherwise.

For every tree we define a flag function associated with it, and we use it during the expansion of the nodes of the tree. If B is a branch, then $B.\mathbf{n}$ denotes the result of the expansion of B with the node $\mathbf{n}$ (i.e., the addition of an edge connecting the leaf of B to $\mathbf{n}$). Also, $B.\mathbf{n}_1|...|\ \mathbf{n}_k$ denotes the result of the expansion of B with k direct successor nodes $\mathbf{n}_1, ..., \mathbf{n}_k$.

Before we present the tableau rules, we should explain the way in which we deal with some operators, in particular, with $\{\Diamond_r^-, \Diamond_l^-, \Diamond_r^+, \Diamond_l^+\}$. While it is straightforward to determine the successors of a node when we expand it using the rules for an operator in $\{\wedge, \vee, \Diamond_r^*, \Diamond_l^*\}$, there is a subtle point for the expansion of the node when we use the rules for the remaining operators. The problem is to find a way to specify a length for an interval that can be used during the application of tableau rules to a subformula $\Diamond_z^+\varphi$ or $\Diamond_z^-\varphi$ of a formula ψ when the length is not determined. We select a length for these intervals in such a way that no overlap exists between the intervals already used in the process of satisfying ψ and the intervals needed for the satisfaction of φ. If two intervals overlap, the probability of having a clash increases: intervals may individually contain consistent propositions, but when we check their satisfiability in a common interval, they may be inconsistent.

Let ψ be a formula, and suppose we wish to determine if we can satisfy ψ. We present a strategy for selecting suitable intervals. $LN(\psi)$ denotes the sum of the lengths of propositions appearing in ψ. In other words, $LN(\psi)$ determines the maximum time needed for satisfying the propositions (assuming that the required intervals are laid end to end, in some order). We select an arbitrary constant $\kappa \in \mathbb{N} \geq LN(\psi)$. We use this constant when we want to find a candidate interval for the satisfaction of $\Diamond_z^{z'}$ $(z \in \{r, l\}, z' \in \{+, -\})$. The use of this constant alone is not sufficient to find the interval. We define a sequence of constants $b_1, b_2, \ldots$, where $b_v = 2^{v-1}$, i.e., every element of this sequence is the sum of the previous elements plus one. We assign a superscript b_v to every $\Diamond_z^{z'}$ $(z \in \{r, l\}, z' \in \{+, -\})$ so that the v reflects their order of appearance in the formula. For convenience, we put the constants as left superscripts of the operator, e.g., ${}^1\Diamond_r^-\varphi$. Now, we use the following strategy to find a suitable interval to check the satisfiability of $\Diamond_z^{z'}$ $(z \in \{r, l\}, z' \in \{+, -\})$.

Definition 22. (Selection Strategy). To check the satisfiability of ${}^{b_v}\Diamond_r^-\varphi$ (resp. ${}^{b_v}\Diamond_l^-\varphi$) on $[c_i, c_j]$, we select an interval $[c_j, c_k]$ (resp. $[c_k, c_i]$) where $c_k = b_v * \kappa + c_j$ (resp. $c_k = c_i - b_v * \kappa$), and use it to check the satisfiability of φ.

Our selection strategy assures that there is no possibility of overlap between intervals used for satisfaction of any subformula of φ and the intervals already used in the process of satisfaction. We prove the soundness of this in Lemma 2. However, we note that sometimes the strategy of selecting intervals is not important, and the satisfaction of φ is completely independent of the selection strategy: The length of the interval used to satisfy $\Diamond_r^{z'}$ succeeded by $\Diamond_l^{z''}\varphi$ or $\Diamond_l^{z'}$ succeeded by $\Diamond_r^{z''}\varphi$ is not important. For example, the following figure exhibits the satisfaction of $\Diamond_r^-\Diamond_l^* p_{10}$ on $[c_0, c_1]$. As can be seen in the figure, the length of $[c_1, c_2]$ is not important, and has no effect on the satisfiability of $\Diamond_l^* p_{10}$ or p_{10}. We are able to select any c_2 greater than c_1, and the result is the same.

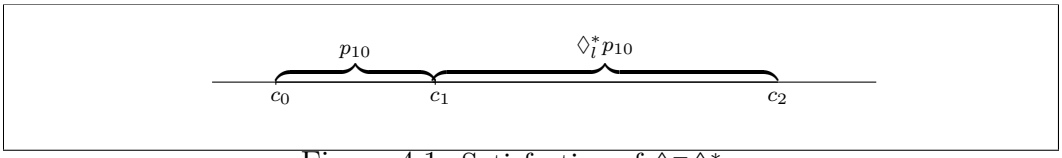

Figure 4.1: Satisfaction of $\Diamond_r^-\Diamond_l^* p_{10}$

Lemma 2. Let ψ be a formula, and let $\Diamond_z^{z'}\varphi$ ($z \in \{r, l\}$,$z' \in \{+, -\}$) be one of its subformulas. When we use our selection strategy, there is no overlap between intervals used for the satisfaction of φ and the intervals already used in the process of satisfaction of ψ when we check the satisfiability of $\Diamond_z^{z'}\varphi$.

Proof. Suppose we want to check the satisfiability of $\Diamond_z^{z'}\varphi$ on $[c_i, c_j]$ where c_s and c_f are, respectively, the minimum and the maximum time points which are used in the process of satisfaction of ψ before we begin to check the satisfiability of $\Diamond_z^{z'}\varphi$ (see Figure 4.2). Also, assume M is the sum of the lengths of

propositions satisfied before we begin to check the satisfiability of $\Diamond_z^{z'}\varphi$. Let $h = |c_f - c_s|$, and let κ be a positive integer constant $\kappa \geq LN(\psi)$. In Figure 2, Q is the length of a gap appearing due to using our selection strategy. In order to prove the theorem, it is sufficient to show that $Q \geq LN(\varphi)$. Since $LN(\varphi) = LN(\psi) - M$, we can equivalently prove $Q \geq (LN(\psi) - M)$.

We prove $Q \geq (LN(\psi) - M)$ when Q is defined as follows:

- if we want to check the satisfiability of $\Diamond_r^{z'}\varphi$, $Q = |c_j + 2^{v-1} * \kappa - c_f|$ (see Figure 4.2 (top)).
- if we want to check the satisfiability of $\Diamond_l^{z'}\varphi$, $Q = |c_s - (c_i - 2^{v-1} * \kappa)|$ (see Figure 4.2 (bottom)).

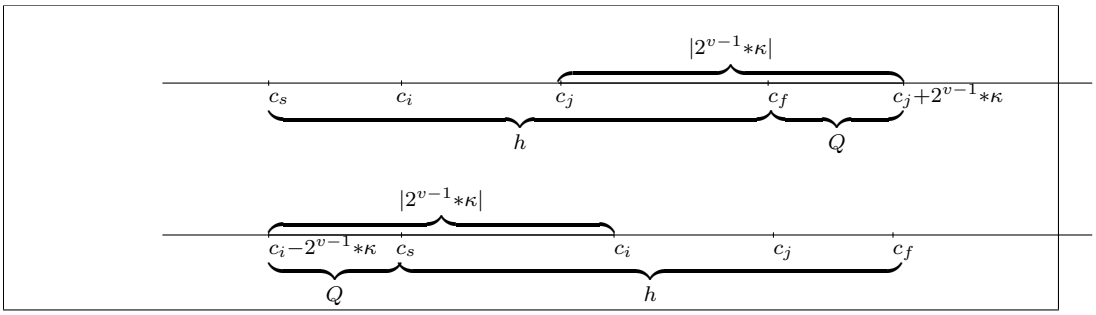

Figure 4.2: Satisfaction of $\Diamond_r^{z'}\varphi$ (top),$\Diamond_l^{z'}\varphi$ (bottom)

The worst scenario for the first case is to have $c_j = c_s$. Since we have already satisfied $v - 1$ occurrences of $\Diamond_z^{z'}$, we have $h \leq M + \Sigma_{y=1}^{v-1}(2^{y-1} * \kappa)$. Proof of the second case is analogous to the following, but where the worst case scenario is $c_i = c_f$.

$$
\begin{aligned}
Q &= |c_s + 2^{v-1} * \kappa - c_f| && (c_s = c_j) \\
&= |2^{v-1} * \kappa - h| && (h = |c_f - c_s|) \\
&\geq |2^{v-1} * \kappa - M - \Sigma_{y=1}^{v-1}(2^{y-1} * \kappa)| && (h \leq M + \Sigma_{y=1}^{v-1}(2^{y-1} * \kappa)) \\
&= |2^{v-1} * \kappa - M - (2^{v-1} * \kappa - \kappa)| && (\Sigma_{y=1}^{v-1}(2^{y-1} * \kappa) = 2^{v-1} * \kappa - \kappa) \\
&= |\kappa - M| \\
&\geq |LN(\psi) - M| && (\kappa \geq LN(\psi))
\end{aligned}
$$

□

Theorem 2. (Soundness of our selection strategy.) If a formula ψ is not satisfiable when our selection strategy is used to specify the length of intervals, it is not satisfiable with any other selection strategy.

Proof. By Lemma 2, our selection strategy provides enough of a gap for the satisfaction of a subformula, and prevents the overlap between the intervals required for the satisfaction of the subformula and the intervals already used in the process of satisfaction of the formula, therefore we are sure that the inconsistency is not removed by expanding, shrinking or removing the gap. □

Formally, LN : IMPNL^a $\rightarrow$ $\mathbb{N}$ calculates the sum of the lengths of the propositions appearing in the formula, and is inductively defined as follows.

1. $LN(\psi)$=$\langle$k$\rangle$ *if* $\psi \in \{p_k, \neg p_k, \top_k, \bot_k\}$
2. $LN(\psi)$=$LN(\varphi)$ *if* $\psi \in \{\Diamond_z^* \varphi, \Diamond_z^+ \varphi, \Diamond_z^- \varphi\}$
3. $LN(\psi)$=$LN(\varphi_1)$+$LN(\varphi_2)$ *if* $\psi = \varphi_1\ \omega\ \varphi_2$ where $\omega \in \{\vee, \wedge\}$

Definition 23. Given a *tree* $\mathcal{T}$, a branch B in $\mathcal{T}$, and a node $\mathbf{n} \in B$ such that $\mathbf{n} =((\psi, [c_i, c_j]), \mathbb{C}_\mathcal{B})$, with $u(\mathbf{n},B) = 0$, the *expansion rule* for B and $\mathbf{n}$ is defined as follows. Note, we use our interval selection strategy in the definition of the rules.

$\mathrm{R}_\wedge$: If $\psi = \varphi_0 \wedge \varphi_1$, then expand the branch to $B.\mathbf{n}_0.\mathbf{n}_1$, s.t. $\mathbf{n}_0 = ((\varphi_0,[c_i, c_j]), \mathbb{C}_\mathcal{B})$ and $\mathbf{n}_1 = ((\varphi_1,[c_i, c_j]), \mathbb{C}_\mathcal{B})$;

$\mathrm{R}_\vee$: If $\psi = \varphi_0 \vee \varphi_1$, then expand the branch to $B.\mathbf{n}_0|\mathbf{n}_1$, s.t. $\mathbf{n}_0 = ((\varphi_0,[c_i, c_j]), \mathbb{C}_\mathcal{B})$ and $\mathbf{n}_1 = ((\varphi_1,[c_i, c_j]), \mathbb{C}_\mathcal{B})$;

R^*_r: If $\psi = \Diamond^*_r\varphi$, then expand the branch to $B.\mathbf{n}_1|...|\mathbf{n}_f$, s.t. $\mathbf{n}_1 = ((\varphi, [c_j, c_{k_1}]), \mathbb{C}_{\mathcal{B}_1})$ and $c_j < c_{k_1}$, and $(c_j, c_{k_1}, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f = ((\varphi,[c_j, c_{k_f}]), \mathbb{C}_{\mathcal{B}_f})$ and $c_j < c_{k_f}$, and $(c_j, c_{k_f}, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$.
For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_\mathcal{B}} = L_{\mathbb{C}_{\mathcal{B}_m}}$ then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_\mathcal{B}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_{k_m} in $\mathbb{C}_\mathcal{B}$ and $L_{\mathbb{C}_{\mathcal{B}_m}}$ by inserting $(c_j, c_{k_m}, FF^m(\varphi))$ in $L_{\mathbb{C}_\mathcal{B}}$.

* Note that R^*_l, R^+_l, R^-_l are respectively analogous to R^*_r, R^+_r, R^-_r with a small difference in the intervals used for the satisfaction of formulas.

R^*_l: If $\psi = \Diamond^*_l\varphi$, then expand the branch to $B.\mathbf{n}_1|...|\mathbf{n}_f$, s.t. $\mathbf{n}_1 = ((\varphi, [c_{k_1}, c_i]), \mathbb{C}_{\mathcal{B}_1})$ and $c_{k_1} < c_i$, and $(c_{k_1}, c_i, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f = ((\varphi,[c_{k_f}, c_i]), \mathbb{C}_{\mathcal{B}_f})$ and $c_{k_f} < c_i$, and $(c_{k_f}, c_i, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$. For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_\mathcal{B}} = L_{\mathbb{C}_{\mathcal{B}_m}}$, then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_\mathcal{B}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_i and c_{k_m} in $\mathbb{C}_\mathcal{B}$, and $L_{\mathbb{C}_{\mathcal{B}_m}}$ is obtained by inserting $(c_{k_m}, c_i, FF^m(\varphi))$ in $L_{\mathbb{C}_\mathcal{B}}$.

R^-_r: If $\psi = {}^{b_v}\Diamond^-_r\varphi$, then expand the branch to $B.\mathbf{n}_0$, s.t. $\mathbf{n}_0 = ((\varphi,[c_j, c_{k_0}]), \mathbb{C}_{\mathcal{B}_0})$ where $c_j < c_{k_0}$ and $c_{k_0} = c_j + b_v * \kappa$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_{k_0} in $\mathbb{C}_\mathcal{B}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting $(c_j, c_{k_0}, b_v * \kappa)$ in $L_{\mathbb{C}_\mathcal{B}}$.

R_l^-: If $\psi = {}^{b_v}\Diamond_l^-\varphi$, then expand the branch to $B.\mathbf{n}_0$, s.t. $\mathbf{n}_0 = ((\varphi,[c_{k_0}, c_i])$, $\mathbb{C}_{\mathcal{B}_0})$ where $c_{k_0} < c_i$ and $c_{k_0} = c_i - b_v * \kappa$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_{k_0} in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting $(c_{k_0}, c_i, b_v * \kappa)$ in $L_{\mathbb{C}_{\mathcal{B}}}$.

R_r^+: If $\psi = {}^{b_v}\Diamond_r^+\varphi$, then expand the branch to $B.\mathbf{n}_0|(\mathbf{n}_1|...|\mathbf{n}_f)$, s.t. $\mathbf{n}_0 = ((\varphi,[c_j, c_{k_0}])$, $\mathbb{C}_{\mathcal{B}_0})$ and $c_j < c_{k_0}$ and $c_{k_0} = c_j + b_v * \kappa$ and $\mathbf{n}_1 = ((\varphi,[c_j, c_{k_1}])$, $\mathbb{C}_{\mathcal{B}_1})$ and $c_j < c_{k_1}$, and $(c_j, c_{k_1}, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f = ((\varphi,[c_j, c_{k_f}])$, $\mathbb{C}_{\mathcal{B}_f})$ and $c_j < c_{k_f}$, and $(c_j, c_{k_f}, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_j, c_{k_0} in $\mathbb{C}_{\mathcal{B}}$, and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting $(c_j, c_{k_0}, b_v * \kappa)$ in $L_{\mathbb{C}_{\mathcal{B}}}$. For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_{\mathcal{B}}} = L_{\mathbb{C}_{\mathcal{B}_m}}$, then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_{\mathcal{B}}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_j and c_{k_m} in $\mathbb{C}_{\mathcal{B}}$, and $L_{\mathbb{C}_{\mathcal{B}_m}}$ is obtained by inserting $(c_j, c_{k_m}, FF^m(\varphi))$ in $L_{\mathbb{C}_{\mathcal{B}}}$.

R_l^+: If $\psi = {}^{b_v}\Diamond_l^+\varphi$, then expand the branch to $B.\mathbf{n}_0|(\mathbf{n}_1|...|\mathbf{n}_f)$, s.t. $\mathbf{n}_0 = ((\varphi,[c_{k_0}, (c_i)])$, $\mathbb{C}_{\mathcal{B}_0})$ and $c_{k_0} < c_i$ and $c_{k_0} = c_i - b_v * \kappa$ and $\mathbf{n}_1 = ((\varphi,[c_{k_1}, c_i])$, $\mathbb{C}_{\mathcal{B}_1})$ and $c_{k_1} < c_i$, and $(c_{k_1}, c_i, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f = ((\varphi,[c_{k_f}, c_i])$, $\mathbb{C}_{\mathcal{B}_1})$ and $c_{k_f} < c_i$, and $(c_{k_f}, c_i, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_i, c_{k_0} in $\mathbb{C}_{\mathcal{B}}$, and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting $(c_{k_0}, c_i, b_v * \kappa)$ in $L_{\mathbb{C}_{\mathcal{B}}}$. For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_{\mathcal{B}}} = L_{\mathbb{C}_{\mathcal{B}_m}}$, then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_{\mathcal{B}}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_{k_m} in $\mathbb{C}_{\mathcal{B}}$, and $L_{\mathbb{C}_{\mathcal{B}_m}}$ is obtained by inserting $(c_{k_m}, c_i, FF^m(\varphi))$ in $L_{\mathbb{C}_{\mathcal{B}}}$.

In all the cases considered, $u(\mathbf{n}', B') = 0$ for all new pairs $(\mathbf{n}', B')$ of nodes and branches and $u(\mathbf{n},B) = 1$ for the node which is expanded.

Definition 24. A node $\mathbf{n}$ in a tree $\mathcal{T}$ is expandable with respect to a branch B iff $u(\mathbf{n}, B)$=0.

Definition 25. A branch B is *closed* if at least one of the following conditions holds:

1. There are two nodes $\mathbf{n}$, $\mathbf{n}^{'} \in B$ such that $\mathbf{n} = ((p_l,[c_{i_0}, c_{j_0}]), \mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}^{'} = ((\neg p_m,[c_{i_1}, c_{j_1}]),\mathbb{C}^{'}_{\mathcal{B}})$ for some atomic formula p and $[c_{i_0}, c_{j_0}] \cap [c_{i_1}, c_{j_1}] \neq \emptyset$;

2. There is a node $\mathbf{n} \in B$ such that $\mathbf{n} = ((p_k,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ and $(c_i, c_j, k) \in L_{\mathbb{C}_{\mathcal{B}}}$ and $|c_j - c_i| \neq k$;

3. There is a node $\mathbf{n} \in B$ s.t. $\mathbf{n} = ((p_l,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ while p_l is an atomic formula and $\exists k_1, k_2 \in \mathbb{C}_{\mathcal{B}}$ *s.t.* $(c_i, c_j, k_1) \in L_{\mathbb{C}_{\mathcal{B}}}$ and $(c_i, c_j, k_2) \in L_{\mathbb{C}_{\mathcal{B}}}$ and $k_1 \neq k_2$.

If none of the above conditions hold, the branch is *open*, which means that there is no inconsistency between the labeled formulas residing on the branch, and as we shall see by Lemma 5 and Theorem 4, we are able to build a class of models satisfying the labeled formulas on the branch.

Definition 26. A tableau for a formula in IMPNL^a is *closed* if and only if every branch in it is closed, otherwise it is *open*.

Definition 27. For a branch B in a tree $\mathcal{T}$, the *expansion strategy* is defined as follows:

1. Apply an expansion rule to a branch B only if it is open;

2. If B is open, apply the expansion rule to any node (say $\mathbf{n}$) in B to which the expansion rule is applicable (if any).

Definition 28. An *initial tableau* for a given formula $\psi \in$ IMPNL^a is a finite tree $\mathcal{T}$ shown below.

$[root]$ $((\psi,[c_0, c_1])$, $\{\{c_0 < c_1\},\{(c_0, c_1, k)\}\})$
where $k \in \mathbb{N}$ is an arbitrary constant.

Lemma 3. If $\psi \in$ IMPNL, ψ and $\Diamond_r\psi$ are equi-satisfiable over $(\mathbb{Z}, <)$.

Proof. We prove the first case. Suppose that ψ is satisfiable on $[c_i,c_j]$. Since c_i and c_j are finite positive integers, we can find a $c_k \in \mathbb{Z}$ such that $c_k < c_i$. Based on the semantics of IMPNL, $\Diamond_r\psi$ is satisfiable on $[c_k,c_i]$. Now, assume $\Diamond_r\psi$ is satisfiable on $[c_m, c_n]$. This means there is an interval $[c_n, c_{k_0}]$ such that ψ is satisfiable. □

Since in this thesis we use $\mathbb{Z}$ as our linear interval structure, we proved the above lemma for $(\mathbb{Z}, <)$. The main idea for proving the lemma for $(\mathbb{R}, <)$, $(\mathbb{Q}, <)$, $(\mathbb{N}, <)$ is the same.

Definition 29. A *tableau* for a given formula $\psi \in$ IMPNL is a finite tree $\mathcal{T}$ obtained by expanding the initial tableau for the annotated version of $\Diamond_r\psi$ through successive applications of the expansion strategy to the existing branches.

4.1.4 Soundness of the tableau algorithm for IMPNL formulas

Definition 30. A *strict model* is a model in which every interval is a strict interval.

Definition 31. Given a set S of labeled formulas with labels in $\mathbb{C}$, we say that S is satisfiable over $\mathbb{C}$ if there exists a strict model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}^-(\mathbb{D}), V\rangle$ such that $\mathbb{D}$ is an extension of $\mathbb{C}$ and $\mathrm{M},[c_i, c_j] \models \psi$ for all $(\psi, [c_i, c_j]) \in S$.

Theorem 3. (Soundness). If $\psi \in$ IMPNL and a tableau $\mathcal{T}$ for the annotated version of $\Diamond_r\psi$ is closed, then ψ is not satisfiable.

Proof. Let $\mathbb{C}$ be the interval structure in $\mathbf{n}$. $P(m)$ is the statement: if the following conditions hold:

1. $\mathbf{n}$ is a node;
2. the height of $\mathbf{n}$ is m;
3. every branch through $\mathbf{n}$ is closed;

then the set $S(\mathbf{n})$ of all labeled formulas in the nodes between $\mathbf{n}$ and the root is not satisfiable over $\mathbb{C}$. We will prove $P(m)$ is true for all $m \geq 0$ using strong induction. We present the general sketch of the induction here. We refer the reader to Appendix A for the details of proof.

(Base case) If $m = 0$, then $\mathbf{n}$ is a leaf, and the unique branch B containing $\mathbf{n}$ is closed. Then, we have one of the following cases.

1. $S(\mathbf{n})$ contains both the labeled formulas $(p_s,[c_{k_1}, c_{l_1}])$ and $(\neg p_r,[c_{k_2}, c_{l_2}])$ where $([c_{k_1}, c_{l_1}] \cap [c_{k_2}, c_{l_2}] \neq \emptyset)$.
2. $S(\mathbf{n})$ contains the labeled formula $(p_s,[c_k, c_{k_0}])$ and $(c_{k_0}, c_k, s) \in L_{\mathbb{C}}$ and $|c_k - c_{k_0}| \neq s$.

In both cases, we are not able to construct a model for the labeled formulas in set S($\mathbf{n}$). Hence, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

(Induction Case) Assume $P(m)$ holds for all m, $0 \leq m \leq t$. We want to prove $P(t+1)$ holds. Suppose the height of $\mathbf{n}$ is $t+1$, and C=$\{c_0,...,c_n\}$. There are two cases to consider; (1) when $\mathbf{n}$ is the direct successor which results after applying the $\mathcal{R}_\wedge$ rule on node $\mathbf{g}$ s.t. $\mathbf{g} = (\varphi_0 \wedge \varphi_1,[c_i, c_j], \mathbb{C})$, and (2) when an expansion rule is applied to $\mathbf{n}$ s.t. $\mathbf{n} = ((\psi,[c_i, c_j]), \mathbb{C})$ or an expansion rule is applied to some labeled formula $(\psi,[c_i, c_j]) \in S(\mathbf{n}) - \{\Phi(\mathbf{n})\}$ (i.e. the existing formula in $\mathbf{n}$) to extend the branch at $\mathbf{n}$. In both cases $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$ (see the details in Appendix A). □

4.1.5 Completeness of the tableau algorithm for IMPNL formulas

Definition 32. A set S of labeled formulas is a *downward saturation set* if and only if it has the following properties.

1. if $(p_k, [c_i, c_j]) \in S$, then $|c_j - c_i| = k$, and there is no $(\neg p_{k'}, [c_{i'}, c_{j'}]) \in S$ s.t. $[c_{i'}, c_{j'}] \subseteq [c_i, c_j]$;
2. if $(\neg p_k, [c_i, c_j]) \in S$, then $|c_j - c_i| = k$, and there is no $(p_{k'}, [c_{i'}, c_{j'}]) \in S$ s.t. $[c_{i'}, c_{j'}] \subseteq [c_i, c_j]$;
3. if $(\varphi_1 \wedge \varphi_2, [c_i, c_j]) \in S$, then $(\varphi_1, [c_i, c_j]) \in S$ and $(\varphi_2, [c_i, c_j]) \in S$;
4. if $(\varphi_1 \vee \varphi_2, [c_i, c_j]) \in S$, then $(\varphi_1, [c_i, c_j]) \in S$ or $(\varphi_2, [c_i, c_j]) \in S$;
5. if $(\Diamond_r^{u'} \varphi, [c_i, c_j]) \in S$ where $u' \in \{*, +, -\}$, then $(\varphi, [c_j, c_m]) \in S$ for some c_m;
6. if $(\Diamond_l^{u'} \varphi, [c_i, c_j]) \in S$ where $u' \in \{*, +, -\}$, then $(\varphi, [c_n, c_i]) \in S$ for some c_n.

Note that when the expansion of the tableau for ψ is finished, there is no node in any open branch B to which any expansion rule is applicable.

Lemma 4. The set of labeled formulas occurring on an open branch B forms a downward saturation set.

Proof. Let S be the set of labeled formulas occurring on B. Since B is an open branch, in particular, it should not satisfy the first two conditions mentioned in Definition 25 (on Page 67). Equivalently, S should satisfy the first and second properties of a downward saturation set. Suppose $(\psi, [c_i, c_j]) \in S$; based on Definition 27 (on Page 68), we have the following cases:

- $\psi = (\varphi_1 \wedge \varphi_2, [c_i, c_j]) \in S$, then based on $R_\wedge$ in Definition 23 (on Page 66), $(\varphi_1, [c_i, c_j]) \in S$ and $(\varphi_2, [c_i, c_j]) \in S$;
- $\psi = (\varphi_1 \vee \varphi_2, [c_i, c_j]) \in S$, then based on $R_\vee$ in Definition 23, $(\varphi_1, [c_i, c_j]) \in S$ or $(\varphi_2, [c_i, c_j]) \in S$;
- $\psi = (\Diamond_r^{u'} \varphi, [c_i, c_j]) \in S$ where $u' \in \{*, +, -\}$, then based on R_r^d in Definition 23, $(\varphi, [c_j, c_m]) \in S$ for some c_m;
- $\psi = (\Diamond_l^{u'} \varphi, [c_i, c_j]) \in S$ where $u' \in \{*, +, -\}$, then based on R_l^d in Definition 23, $(\varphi, [c_n, c_i]) \in S$ for some c_n.

Therefore, S satisfies the properties of a saturation set. □

Lemma 5. (Hintikka-style Lemma) Every downward saturation set [110] occurring on an open branch B is satisfiable.

Proof. Recall, length of a formula, ψ, is the cardinality of the multi set of operators occurring in it. $P(m)$ is the statement: There is a model M for the labeled formulas of length at most m in a downward saturation set S. We will prove $P(m)$ is true for all $m \geq 0$ using strong induction.

(Base Case) In this case (m=0), we consider atomic formulas in S. We construct a model M: for every labeled formula $(p_k, [c_i, c_j]) \in S$ ($(\neg p_k, [c_i, c_j]) \in S$), let $V([c_i', c_j']) = \{p_{k'}\}(\{\neg p_{k'}\})$ where $k' = c_j' - c_i'$ and $[c_i', c_j'] \subseteq [c_i, c_j]$. Such a construction is possible, and $M, [c_i, c_j] \models p_k(\neg p_k)$, since based on Definition 32, item 1 (Definition 32, item 2), $|c_j - c_i| = k$ and there is no labeled atomic formula which causes an inconsistency.

(Induction Case) Assume that $P(m)$ is true for all $0 \leq m \leq t$. We prove $P(t+1)$ is true. For every $(\psi, [c_i, c_j]) \in S$ such that the length of ψ is $t+1$, consider the following cases.

- $(\psi = \varphi_1 \wedge \varphi_2, [c_i, c_j]) \in S$. Based on Definition 32, item 3, we know $(\varphi_1, [c_i, c_j]) \in S$ and $(\varphi_2, [c_i, c_j]) \in S$. Since the length of φ_1 (φ_2) is less than the length of ψ, by the induction assumption, there is a model M such that $M, [c_i, c_j] \models \varphi_1$ and $M, [c_i, c_j] \models \varphi_2$, so $M, [c_i, c_j] \models \psi$.

- $(\psi = \varphi_1 \vee \varphi_2, [c_i, c_j]) \in S$. Based on Definition 32, item 4, we know either $(\varphi_1, [c_i, c_j]) \in S$ or $(\varphi_2, [c_i, c_j]) \in S$. Since the length of φ_1 (φ_2) is less than the length of ψ, by the induction assumption, there is a model M such that $M, [c_i, c_j] \models \varphi_1$ or $M, [c_i, c_j] \models \varphi_2$, so $M, [c_i, c_j] \models \psi$.

- $(\psi = \Diamond_r^{u'} \varphi, [c_i, c_j]) \in S$ where $u' \in \{*, +, -\}$. Based on Definition 32, item 5, we know $(\varphi, [c_j, c_m]) \in S$ for some c_m. Since the length of φ is less than the length of ψ, by the induction assumption, there is a model M such that $M, [c_j, c_m] \models \varphi$, $M, [c_i, c_j] \models \psi$.

- $(\psi = \Diamond_l^{u'} \varphi, [c_i, c_j]) \in S$ where $u' \in \{*, +, -\}$. Proof of this case is analogous to the previous case; use Definition 32, item 6 and change $[c_j, c_m]$ to $[c_n, c_i]$.

□

Corollary 1. Based on Lemma 5, we can find a model for the set of formulas residing on the nodes of an open branch, in particular the formula ψ which resides on the root of the tableau.

Theorem 4. (Completeness). If $\psi \in$ IMPNL and a tableau $\mathcal{T}$ for the annotated version of $\Diamond_r \psi$ is open, then ψ is satisfiable.

Proof. Recall that φ, $\Diamond_r\varphi$ and the annotated version of $\Diamond_r\varphi$ are equi-satisfiable. The result follows by Corollary 1. □

4.1.6 Complexity of the tableau algorithm for IMPNL formulas

Definition 33. The *string length* of a formula ψ is the cardinality of the multi set consisting of propositional variables and operators appearing in ψ; it is denoted by $STLength(\psi)$. The length of the longest branch in a complete tableau for formula ψ is denoted by $Length_{br}(\psi)$, and is inductively defined as follows:

- $Length_{br}(\psi) = 1$ iff $\psi \in \{p_k, \top_k, \bot_k, \neg p_k\}$
- $Length_{br}(\psi) = Max\{Length_{br}(\psi_1), Length_{br}(\psi_2)\}+1$ iff $\psi = \psi_1 \vee \psi_2$
- $Length_{br}(\psi) = Length_{br}(\psi_1)+Length_{br}(\psi_2)+1$ iff $\psi = \psi_1 \wedge \psi_2$
- $Length_{br}(\psi) = Length_{br}(\psi_1)+1$ iff $\psi \in \{\Diamond_r^*\psi_1, \Diamond_r^-\psi_1, \Diamond_r^+\psi_1\}$

Theorem 5. For every formula ψ, $Length_{br}(\psi) \leq STLength(\psi)$.

Proof. This theorem is easy to prove by induction on the string length of the formula. □

Lemma 6. The longest branch is finite.

Proof. Since the string length of a formula is finite, by Theorem 5, the length of the longest branch is finite. □

Theorem 6. The complexity of satisfiability reasoning in IMPNL is PSpace when the length of an interval is a constant.

Proof. Recall that every labeled formula consists of an IMPNL formula and an interval. Obviously, the space required for storing an interval is a constant. Since the string length of an IMPNL formula in any node of a tableau is equal or less than the string length of the formula in the initial node of the tableau, and the lengths of intervals are fixed constants, the space required for storing a node is PSPACE. Now, the theorem is a direct consequence of Theorem 5 and Lemma 6. □

4.2 Conclusion

In this chapter, we introduced a metric interval-based temporal logic suitable for modeling many processes in different domains (e.g., CPGs in the domain of medicine). Then, we presented a decidable tableau-based algorithm for checking the satisfiability of a formula of IMPNL. The soundness of our algorithm implies that any temporal inconsistency in a process is detectable, and completeness of the algorithm ensures that we are able to find a (class of) model(s) for a process when there is an open branch in its tableau.

In IMPNL, modeling the static aspects of a process is an issue which we cannot easily deal with. For example, we are not able to model drug contraindications in the domain of medicine. In the next chapter, we will introduce a metric interval-based temporal description logic which addresses this problem.

Chapter 5

Metric Interval-based Temporal Description Logic

As we mentioned in the previous chapter, there are some issues which we cannot easily deal with when we use IMPNL, e.g., drug contraindications in the domain of medicine. In this chapter, we combine IMPNL with the description logic $\mathcal{ALC}$. The combined logic, called *Metric Interval-based Temporal Description Logic* (MITDL), is powerful enough to model both the dynamic aspects (e.g., time constraints) and the static aspects (e.g., relation between concepts in a domain) of many domains, e.g., medicine. More precisely, MITDL, is a combination of a description logic, $\mathcal{ALC}$, and a restricted version of IMPNL (IMPNL without its negation operator).

5.1 Syntax

A *signature* Σ of MITDL is a 5 tuple Σ=(**C, R, I,** $\top$**,** $\bot$) where **C** is a set of concept names, **R** is a set of role names, **I** is a set of individual names, and $\top$ (Top concept) and $\bot$ (Bottom concept) are two constant concept names.

Definition 34. Let D be a concept description (see Definition 3 on page 17); let R be a role name; let a, b be two individual names, and let $k \in \mathbb{N}$; a MITDL-*formula* ψ over Σ is defined recursively as follows:
$\psi = (D = \top)_k \mid (a : D)_k \mid R(a, b)_k \mid \top_k \mid \bot_k \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid \Diamond_r \psi \mid \Diamond_l \psi$

Let D and E be two concept descriptions. A GCI $D \sqsubseteq E$, can be written in an equivalent form $\neg D \sqcup E = \top$. Also, $(D = E) \equiv ((D \sqsubseteq E) \wedge (E \sqsubseteq D)) \equiv ((\neg D \sqcup E = \top) \wedge (D \sqcup \neg E = \top))$; so in MITDL, we can write $(\neg D \sqcup E = \top)_k \wedge (D \sqcup \neg E = \top)_k$ in order to model $(D = E)_k$.

5.2 Semantics

The semantics of MITDL is based on a structure $M = (\mathbb{D}, \mathbb{I}^-(\mathbb{D}), \mathbb{S})$ where the pair $(\mathbb{D}, \mathbb{I}^-(\mathbb{D}))$ is a strict interval structure, and $\mathbb{S}$ is a set of $\mathcal{ALC}$ interpretations over the intervals defined: $\mathbb{S} = \{\mathbb{S}[i, j] \mid [i, j] \in \mathbb{I}(\mathbb{D})$ and $\mathbb{S}[i, j] = (\Delta^{[i,j]}, D_0^{[i,j]}, D_1^{[i,j]}, D_2^{[i,j]}, ..., R_0^{[i,j]}, R_1^{[i,j]}, R_2^{[i,j]}, ..., a_0^{[i,j]}, a_1^{[i,j]}, a_2^{[i,j]}, ...)$ is a DL-interpretation on interval $[i, j]\}$. Recall that a DL-interpretation is a DL structure (DL model) in which $\Delta^{[i,j]}$ is a non-empty set of domain elements, $D_m^{[i,j]}$ is a DL-concept description, $R_m^{[i,j]}$ is a DL role name and $a_m^{[i,j]}$ is a DL-individual name.

Remark 1. In this thesis, we assume that $\mathbb{D} = \mathbb{Z}$.

Remark 2. In this thesis, we adopt the expanding domain assumption [111], i.e., $\Delta^{[i,j]} \subseteq \Delta^{[c,d]}$ where $d \geq j$ and $i \geq c$.

Given an interpretation $\mathbb{S}[i, j]$, the concept descriptions are interpreted as follows:

- $\top^{[i,j]} = \Delta^{[i,j]}$
- $\bot^{[i,j]} = \emptyset$
- $D^{[i,j]} \subseteq \Delta^{[i,j]}$
- $(\neg D)^{[i,j]} = \Delta^{[i,j]} \backslash D^{[i,j]}$
- $R^{[i,j]} \subseteq \Delta^{[i,j]} \times \Delta^{[i,j]}$
- $(D \sqcap E)^{[i,j]} = D^{[i,j]} \cap E^{[i,j]}$
- $(D \sqcup E)^{[i,j]} = D^{[i,j]} \cup E^{[i,j]}$
- $(\exists R.D)^{[i,j]} = \{x | (x,y) \in R^{[i,j]} \ and \ y \in D^{[i,j]}\}$
- $(\forall R.D)^{[i,j]} = \{x | (x,y) \in R^{[i,j]} \ implies \ y \in D^{[i,j]}\}$

The satisfaction relation for a given model M and an interval $[i,j]$ is defined as follows:

- $M, [i,j] \models \top_k$ iff $j - i = k$
- $M, [i,j] \models \bot_k$ never
- $M, [i,j] \models (D = \top)_k$ iff $j - i = k$ and $\forall i', j'$, if $[i', j'] \subseteq [i,j]$ then $D^{[i',j']} = \top^{[i',j']}$
- $M, [i,j] \models (a : D)_k$ iff $j - i = k$ and $\forall i', j'$, if $[i', j'] \subseteq [i,j]$ then $a^{[i',j']} \in D^{[i',j']}$
- $M, [i,j] \models R(a,b)_k$ iff $j - i = k$ and $\forall i', j'$, if $[i', j'] \subseteq [i,j]$ then $(a^{[i',j']}, b^{[i',j']}) \in R^{[i',j']}$
- $M, [i,j] \models \psi_1 \vee \psi_2$ $(\psi_1 \wedge \psi_2)$ iff $M, [i,j] \models \psi_1$ *or* (and) $M, [i,j] \models \psi_2$;
- $M, [i,j] \models \Diamond_r \psi$ iff there exists $h > j$ such that $M, [j,h] \models \psi$;
- $M, [i,j] \models \Diamond_l \psi$ iff there exists $h < i$ such that $M, [h,i] \models \psi$.

Note that $\Diamond_z(\psi_1 \vee \psi_2) \Leftrightarrow \Diamond_z \psi_1 \vee \Diamond_z \psi_2$ $(z \in \{r,l\})$ and $\Diamond_z(\psi_1 \wedge \psi_2) \rightarrow \Diamond_z \psi_1 \wedge \Diamond_z \psi_2$ $(z \in \{r,l\})$.

Definition 35. A *simple* formula is a MITDL formula which does not contain a formula $(D = \top)_k$ as a subformula, where D is a concept description, and k is a positive integer. An arbitrary MITDL formula may be referred to as a generic formula.

5.3 Tableau-based algorithm for checking the satisfiability of a MITDL formula

In this section we present two tableau-based algorithms for the satisfiability checking of MITDL formulas. The first algorithm is an extension of the algorithm which we introduced in Section 4.1.3. We have added new expansion rules, called DL rules, required for dealing with DL-formulas (Definition 36 on Page 81). We designed these rules by adding a temporal dimension to the rules presented in [112]. The first algorithm is used for checking the satisfiability of a *simple* formula. We will later prove that this algorithm has PSpace complexity.

We then extend and modify the first algorithm to obtain an algorithm (second algorithm) for checking the satisfiability of a generic MITDL formula. To accomplish this, and motivated by [25], we have designed a temporal version of the subset blocking technique (Definition 43 on page 110) required for ensuring the termination of the second algorithm in the case that the algorithm falls into a cycle. We will show that the complexity of this algorithm is 2ExpTime.

Similar to the proposed tableau algorithm for IMPNL (Section 4.1.3 on Page 55), we here use the annotated version of the original formula to derive the tableau. Note that the annotation process of a MITDL formula is exactly the same as the annotation process for an IMPNL formula (see Section 4.1.3.1

on Page 56). Recall that the annotated version of the formula is equi-satisfiable with respect to the original formula (Lemma 1 on Page 58). The root node of the tableau is created based on Definition 28 (on Page 68). Then, the expansion strategy, defined in Definition 27 (on Page 68), indicates how the tableau rules (Definition 44 on Page 111) should be used to derive the tableau. If one of the fully expanded branches is open (Definition 37 on Page 85), the formula is satisfiable (Theorem 13 on Page 114). If all branches are closed, the formula is not satisfiable. We begin by introducing some definitions and functions needed in the tableau rules. Recall, it is more efficient to assume that all concept descriptions appearing in a formula are in NNF (see Definition 4 on Page 23).

5.3.1 Satisfiability checking of a simple formula

In this section, we give the details of the tableau construction for checking the satisfiability of a simple formula. Let MITDL^a denote the set of annotated versions of formulas built using the syntax of MITDL. The definition of a *labelled formula* is analogous to Definition 20 (on Page 60) with the exception that $\psi \in$ MITDL^a. Also, the definition of the FF function is analogous to the definition of the FF function in Section 4.1.3.2 (on Page 59) with the exception that

- FF : MITDL$^a \rightarrow 2^{\mathbb{N}}_{\mathrm{O}}$, and
- the first line of the definition is changed to: FF(ψ)=$\langle k \rangle$ where $\psi \in \{(a : D)_k, R(a,b)_k, (D = \top)_k, \top_k, \bot_k\}$ and a, b are two individuals, D is a concept description and R is a role name.

Morover, the definition of the LN function is analogous to the definition of the LN function on page 65 with the exception that

- LN : MITDL$^a \rightarrow \mathbb{N}$, and
- the first line of the definition is changed to: LN(ψ)=$\langle k \rangle$ where $\psi \in \{(a : D)_k, R(a,b)_k, (D = \top)_k, \top_k, \bot_k\}$ and a, b are two individuals, D is a concept description and R is a role name.

Suppose that (wlog) all concept descriptions are in NNF. The following definition presents the tableau expansion rules. Note that the temporal rules mentioned in the following definition are identical to the temporal rules presented in Definition 23 (on Page 66) except that we have $\psi, \varphi, \varphi_1, \varphi_2 \in$ MITDL here.

Definition 36. Given a tree $\mathcal{T}$, a branch B in $\mathcal{T}$, and a node $\mathbf{n} \in B$ such that $\mathbf{n}$ =$((\psi, [c_i, c_j])$,$\mathbb{C}_\mathcal{B})$, with $u(\mathbf{n},B) = 0$, the expansion rule for B and $\mathbf{n}$ is defined as follows.

- Temporal Rules

  $\mathcal{R}_\wedge$: If $\psi = \varphi_0 \wedge \varphi_1$, then expand the branch to $B.\mathbf{n}_0.\mathbf{n}_1$, s.t. $\mathbf{n}_0 =$ $((\varphi_0,[c_i, c_j])$, $\mathbb{C}_\mathcal{B})$ and $\mathbf{n}_1 = ((\varphi_1,[c_i, c_j])$, $\mathbb{C}_\mathcal{B})$ and let $u(\mathbf{n},B) = 1$;

  $\mathcal{R}_\vee$: If $\psi = \varphi_0 \vee \varphi_1$, then expand the branch to $B.\mathbf{n}_0|\mathbf{n}_1$, s.t. $\mathbf{n}_0 =$ $((\varphi_0,[c_i, c_j])$, $\mathbb{C}_\mathcal{B})$ and $\mathbf{n}_1 = ((\varphi_1,[c_i, c_j])$, $\mathbb{C}_\mathcal{B})$ and let $u(\mathbf{n},B) = 1$;

  $\mathcal{R}_{\Diamond^*_r}$: If $\psi = \Diamond^*_r\varphi$, then expand the branch to $B.\mathbf{n}_1|...|\mathbf{n}_f$, s.t. $\mathbf{n}_1 =$ $((\varphi,[c_j, c_{k_1}])$, $\mathbb{C}_{\mathcal{B}_1})$ and $c_j < c_{k_1}$, and $(c_j, c_{k_1}, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f = ((\varphi,[c_j, c_{k_f}])$, $\mathbb{C}_{\mathcal{B}_f})$ and $c_j < c_{k_f}$, and $(c_j, c_{k_f}, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$.

  For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_\mathcal{B}} = L_{\mathbb{C}_{\mathcal{B}_m}}$, then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_\mathcal{B}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_{k_m} in $\mathbb{C}_\mathcal{B}$ and $L_{\mathbb{C}_{\mathcal{B}_m}}$ is obtained by inserting $(c_j, c_{k_m}, FF^m(\varphi))$ in $L_{\mathbb{C}_\mathcal{B}}$. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\Diamond_r^-}$: If $\psi = {}^{b_v}\Diamond_r^-\varphi$, then expand the branch to $B.\mathbf{n}_0$, s.t. $\mathbf{n}_0 = ((\varphi,[c_j, c_{k_0}]), \mathbb{C}_{\mathcal{B}_0})$ where $c_j < c_{k_0}$ and $c_{k_0} = c_j + b_v * \kappa$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_{k_0} in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting $(c_j, c_{k_0}, b_v * \kappa)$ in $L_{\mathbb{C}_{\mathcal{B}}}$. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\Diamond_r^+}$: If $\psi = {}^{b_v}\Diamond_r^+\varphi$, then expand the branch to $B.\mathbf{n}_0|(\mathbf{n}_1|...|\mathbf{n}_f)$, s.t. $\mathbf{n}_0 = ((\varphi,[c_j, c_{k_0}]), \mathbb{C}_{\mathcal{B}_0})$ and $c_j < c_{k_0}$ and $c_{k_0} = c_j + b_v * \kappa$ and $\mathbf{n}_1 = ((\varphi,[c_j, c_{k_1}]), \mathbb{C}_{\mathcal{B}_1})$ and $c_j < c_{k_1}$, and $(c_j, c_{k_1}, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f = ((\varphi,[c_j, c_{k_f}]), \mathbb{C}_{\mathcal{B}_f})$ and $c_j < c_{k_f}$, and $(c_j, c_{k_f}, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_j, c_{k_0} in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting $(c_j, c_{k_0}, b_v * \kappa)$ in $L_{\mathbb{C}_{\mathcal{B}}}$. For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_{\mathcal{B}}} = L_{\mathbb{C}_{\mathcal{B}_m}}$ then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_{\mathcal{B}}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_j and c_{k_m} in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_m}}$ is obtained by inserting $(c_j, c_{k_m}, FF^m(\varphi))$ in $L_{\mathbb{C}_{\mathcal{B}}}$. Let $u(\mathbf{n},B) = 1$;

* $\mathcal{R}_{\Diamond_l^*}$, $\mathcal{R}_{\Diamond_l^+}$, $\mathcal{R}_{\Diamond_l^-}$ are respectively analogous to $\mathcal{R}_{\Diamond_r^*}$, $\mathcal{R}_{\Diamond_r^+}$, $\mathcal{R}_{\Diamond_r^-}$ with a small difference in the intervals used for the satisfaction of formulas.

$\mathcal{R}_{\Diamond_l^*}$: If $\psi = \Diamond_l^*\varphi$, then expand the branch to $B.\mathbf{n}_1|...|\mathbf{n}_f$, s.t. $\mathbf{n}_1 = ((\varphi,[c_{k_1}, c_i]), \mathbb{C}_{\mathcal{B}_1})$ and $c_{k_1} < c_i$, and $(c_{k_1}, c_i, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f = ((\varphi,[c_{k_f}, c_i]), \mathbb{C}_{\mathcal{B}_f})$ and $c_{k_f} < c_i$, and $(c_{k_f}, c_i, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$.

For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_{\mathcal{B}}} = L_{\mathbb{C}_{\mathcal{B}_m}}$, then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_{\mathcal{B}}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_i and c_{k_m} in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_m}}$ is obtained by inserting $(c_{k_m}, c_i, FF^m(\varphi))$ in $L_{\mathbb{C}_{\mathcal{B}}}$. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\Diamond_l^-}$: If $\psi = {}^{b_v}\Diamond_l^-\varphi$, then expand the branch to $B.\mathbf{n}_0$, s.t. $\mathbf{n}_0 = ((\varphi,[c_{k_0}, c_i]),$

$\mathbb{C}_{\mathcal{B}_0}$) where $c_{k_0} < c_i$ and $c_{k_0} = c_i - b_v * \kappa$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_{k_0} in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting $(c_{k_0}, c_i, b_v * \kappa)$ in $L_{\mathbb{C}_{\mathcal{B}}}$. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\Diamond_l^+}$: If $\psi = {}^{b_v}\Diamond_l^+\varphi$, then expand the branch to $B.\mathbf{n}_0|(\mathbf{n}_1|...|\mathbf{n}_f)$, s.t. $\mathbf{n}_0$ = $((\varphi,[c_{k_0}, c_i])$, $\mathbb{C}_{\mathcal{B}_0})$ and $c_{k_0} < c_i$ and $c_{k_0} = c_i - b_v * \kappa$ and $\mathbf{n}_1$ = $((\varphi,[c_{k_1}, c_i])$, $\mathbb{C}_{\mathcal{B}_1})$ and $c_{k_1} < c_i$, and $(c_{k_1}, c_i, FF^1(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_1}}$; ... ; $\mathbf{n}_f$ = $((\varphi,[c_{k_f}, c_i])$, $\mathbb{C}_{\mathcal{B}_f})$ and $c_{k_f} < c_i$, and $(c_{k_f}, c_i, FF^f(\varphi)) \in L_{\mathbb{C}_{\mathcal{B}_f}}$; here $f = |FF(\varphi)|$. $\mathbb{C}_{\mathcal{B}_0}$ is obtained by inserting c_{k_0}, c_i in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ by inserting $(c_{k_0}, c_i, b_v * \kappa)$ in $L_{\mathbb{C}_{\mathcal{B}}}$. For any m, $1 \leq m \leq f$, if c_{k_m} is already in $\mathbb{C}_{\mathcal{B}_m}$ and $L_{\mathbb{C}_{\mathcal{B}}} = L_{\mathbb{C}_{\mathcal{B}_m}}$ then $\mathbb{C}_{\mathcal{B}_m} = \mathbb{C}_{\mathcal{B}}$; otherwise $\mathbb{C}_{\mathcal{B}_m}$ is obtained by inserting c_{k_m} and c_i in $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_m}}$ is obtained by inserting $(c_{k_m}, c_i, FF^m(\varphi))$ in $L_{\mathbb{C}_{\mathcal{B}}}$. Let $u(\mathbf{n},B) = 1$;

- DL Rules

$\mathcal{R}_{\sqcap}$: If $\psi = (a : C \sqcap D)_k$, then if B does not contain both (or one of the) two nodes (say $\mathbf{n}_0$, $\mathbf{n}_1$) s.t. $\mathbf{n}_0$ = $(((a : C)_k,[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1$ = $(((a : D)_k,[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$, then expand the branch to $B.\mathbf{n}_0.\mathbf{n}_1$. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\sqcup}$: If $\psi = (a : C \sqcup D)_k$, then if B does not contain both two nodes (say $\mathbf{n}_0$, $\mathbf{n}_1$) s.t. $\mathbf{n}_0$ = $(((a : C)_k,[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1$ = $(((a : D)_k,[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$, then expand the branch to $B.\mathbf{n}_0|\mathbf{n}_1$. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\exists}$: If $\psi = (a : \exists R.C)_k$, then if there is no individual (say b) s.t. two nodes (say $\mathbf{m}$, $\mathbf{s}$) s.t. $\mathbf{m}$ = $(((R(a, b))_{k'},[c_{i_0}, c_{j_0}])$, $\mathbb{C}_{\mathcal{B}})$ where $[c_i, c_j] \subseteq [c_{i_0}, c_{j_0}]$ and $\mathbf{s}$ = $(((b : C)_{k''},[c_{i_1}, c_{j_1}])$, $\mathbb{C}_{\mathcal{B}})$ where $[c_i, c_j] \subseteq [c_{i_1}, c_{j_1}]$ exist in the branch B, then expand the branch to $B.\mathbf{n}_0.\mathbf{n}_1$, $\mathbf{n}_0$ = $(((b : C)_k,[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1$ = $((R(a, b)_k,[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$ [113]. Note,

individual b has not appeared elsewhere in the tree. In other words, it is a new individual name. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_\forall$: If $\psi = (a : \forall R.C)_k$, then if there is a node (say $\mathbf{m}$) in branch B s.t. $\mathbf{m} = (((R(a,b))_{k'},[c_{i_0},c_{j_0}]),\ \mathbb{C}_\mathcal{B})$ where $[c_i,c_j] \cap [c_{i_0},c_{j_0}] = [c_{i_2},c_{j_2}] \neq \emptyset$, and there is no node (say $\mathbf{l}$) in branch B s.t. $\mathbf{l} = (((b : C)_{k'''},[c_{i_3},c_{j_3}]),\ \mathbb{C}_\mathcal{B})$ where $[c_{i_3},c_{j_3}] \subseteq [c_{i_2},c_{j_2}]$, then expand the branch to $B.\mathbf{n}_0$, where $\mathbf{n}_0 = (((b : C)_{k''},[c_{i_2},c_{j_2}]),\ \mathbb{C}_\mathcal{B})$, and $k'' = |c_{j_2} - c_{i_2}|$. $\mathbb{C}_{\mathcal{B}_2}$ is equal to $\mathbb{C}_\mathcal{B}$ and $L_{\mathbb{C}_{\mathcal{B}_2}}$ is obtained by inserting (c_{i_2},c_{j_2},k'') in $L_{\mathbb{C}_\mathcal{B}}$.

In order to apply a rule, the conditions mentioned in the description of the rule must be satisfied; otherwise the rule is not applicable. Also, note that $\mathcal{R}_\forall$ can be applied many times, each time adding a new node to the branch; therefore, we have not mentioned the value of $u(\mathbf{n},B)$ in the definition of the rule.

Remark 3. Let B be a branch; let φ be a formula; let $[c_{i_0},c_{j_0}],[c_{i_1},c_{j_1}]$ be two intervals, and let $\mathbf{n}_1=((\varphi,[c_{i_1},c_{j_1}]),\mathbb{C}_\mathcal{B})$ be a node added by applying a rule to a node residing on B. If there is a predecessor of node $\mathbf{n}_1$ (say $\mathbf{n}_0=((\varphi,[c_{i_0},c_{j_0}]),\mathbb{C}_\mathcal{B})$) s.t. $[c_{i_1},c_{j_1}] \subseteq [c_{i_0},c_{j_1}]$, let $u(\mathbf{n}_1,B)=1$. By homogeneity (see Page 53), this remark indicates that it is not necessary to expand a node which carries no new information.

Remark 4. The lemma 3 (on Page 69) and its proof and the definitions 24 (on Page 67), 26 (on Page 68), 27 (on Page 68), 28 (on Page 68), 29 (on Page 29) should be redefined for MITDL formula and for a tableau for a MITDL formula. Since the new definitions and the new lemma and its proof would be identical to the ones which we have mentioned above, we do not redefine them.

Definition 37. A branch B is *closed* if at least one of the following conditions holds:

1. There is a node $\mathbf{n} \in B$ such that $\mathbf{n} = ((\mathrm{p}_k,[c_i, c_j]),\mathbb{C}_\mathcal{B})$ and $(c_i, c_j, k) \in L_{\mathbb{C}_\mathcal{B}}$ and $|c_j - c_i| \neq k$ where p_k is an atomic formula;

2. There is a node $\mathbf{n} \in B$ s.t. $\mathbf{n} = ((\mathrm{p}_l,[c_i, c_j]),\mathbb{C}_\mathcal{B})$ while p_l is an atomic formula and $\exists k_1, k_2 \in \mathbb{C}_\mathcal{B}$ s.t. $(c_i, c_j, k_1) \in L_{\mathbb{C}_\mathcal{B}}$ and $(c_i, c_j, k_2) \in L_{\mathbb{C}_\mathcal{B}}$ and $k_1 \neq k_2$.

3. There are two nodes $\mathbf{n}$,$\mathbf{n}^{'} \in B$ such that $\mathbf{n} = (((a : C)_l,[c_i, c_j]),\mathbb{C}_\mathcal{B})$ and $\mathbf{n}^{'} = (((a : \neg C)_m,[c_{i'}, c_{j'}]),\mathbb{C}^{'}_\mathcal{B})$ for some concept description C and $[c_i, c_j] \cap [c_{i'}, c_{j'}] \neq \emptyset$;

If none of the above conditions holds, the branch is *open*, i.e., there is no node with inconsistent information and, further, there is no inconsistency among the labeled formulas residing on the branch. We will show that we are able to build a class of models satisfying the labeled formula on the branch.

Since the soundness proof and the completeness proof of this algorithm is similar to the proofs of these theorems for the algorithm for generic formulas (see Section 5.3.5 - Page 107), we will respectively prove the soundness and the completeness of the algorithm for the generic formulas in Section 5.3.6 (Page 113) and Section 5.3.7 (Page 114). The only difference between the proofs of soundness and completeness for these algorithms is to consider the application of the $\mathcal{R}_=$ rule in the soundness proof and the completeness proof for the generic algorithm.

5.3.2 Termination of the tableau algorithm for simple MITDL formulas

Theorem 7. (Termination). There cannot be an infinite sequence of rule applications when we build a tableau for a simple formula.

General sketch of proof. We will show that in order to prove this theorem, it is sufficient to demonstrate the number of nodes residing on a branch is finite (see just below, paragraph titled "Proof in detail"). We first prove that the branching factor of a tableau is finite. Then, we define the notion of temporal degree (Definition 39 on Page 88) and use it to divide the nodes of a branch into two categories: temporal nodes and non-temporal nodes. We prove that the number of temporal nodes on a branch, as well as the number of non-temporal nodes on the branch, are finite.

- The number of temporal nodes is finite.

  Every application of a temporal rule adds a finite number of (non-) temporal nodes to a branch. Also, the (string) length of the formula in a node, added by applying a temporal rule on a temporal node (say **m**), is strictly less than the (string) length of the formula in node **m**. Based on these facts, the number of application of temporal rules in the construction of a branch is finite. Having this and the fact that the only way to add a temporal node to a branch is to apply a temporal rule on a temporal node (Lemma 8 on Page 89), we conclude that the number of temporal nodes on a branch is finite.

- The number of non-temporal nodes is finite.

  We first define the notions of *Temporal label* and *Full Label* of an individual w.r.t. a branch (Definition 40 on Page 90). A full label of an

individual is a multi-set that records the concept descriptions of which the individual is a member. Also, we define the notion of *repetitive nodes*, i.e., the nodes that contain no new information (Definition 42 on Page 94). To be able to demonstrate the finiteness of the number of non-temporal nodes, we prove the following facts.

1. The number of distinct intervals appearing in the nodes of a tableau is finite (Lemma 10 on Page 92).

2. A full label of an individual contains a finite number of concept descriptions (Corollary 4 on Page 95).

   We first define the notion of sub-description of a concept description (Definition 41 on Page 92). Based on the definition of this notion we show that the number of distinct concept descriptions in the nodes of a tableau is finite (Lemma 12 on Page 94). Also, we prove that the number of repetitive nodes is at most twice the number of non-repetitive nodes (Theorem 8 on Page 95). Finiteness of a full label of an individual is an easy consequence of these facts.

3. The number of individuals appearing in the nodes of a tableau is finite.

   We divide the individuals into two categories: *old individuals* and *new individuals*. Based on the definition of old individuals, the number of old individuals is finite. We prove that the number of new individuals is finite (Lemma 13 on Page 95).

Having these facts, we show that the number of non-temporal nodes is finite (Theorem 9 on Page 97). Finally, we present the proof of termination.

Proof in detail. Consider the following lemma.

Lemma 7. The branching factor of a tableau for a MITDL formula is finite.

Proof. Recall the definition of a tableau for a simple formula (see Remark 4 on Page 84 and Definition 29 on Page 69). Also, note that by the definition, the FF function (on Page 60) always returns a finite list. By the definition of expansion rules, it is easy to show that every expansion rule adds a finite number of branches to the tableau. Therefore, the branching factor of a tableau is finite. □

Recall that a tableau is a tree and contains no cycle. On the other hand, each application of a rule on a node adds one or more nodes to the branch on which the node resides. Also, by Lemma 7, the branching factor of a tableau is finite. Therefore, if the number of nodes residing on any branch of the tableau is finite, the total number of nodes in the tableau is finite, and consequently the tableau construction terminates. Hence, in order to prove the theorem, it is sufficient to show that the number of nodes residing on an arbitrary branch of the tableau is finite. Consider the following theorem, lemmas and definitions required for the proof.

Definition 38. The *length* of a formula (resp. axiom, concept description), denoted by $|.|$, is its string length.

Definition 39.

- The *Temporal Degree* of a formula is the number of temporal operators it contains, i.e. $\{\wedge, \vee, \Diamond_r^z, \Diamond_l^z\}$ $(z \in \{*, +, -\})$, appearing in the formula.
- A node (say $\mathbf{n}$) in a tableau is called a *temporal node* if the temporal degree of $\Phi(\mathbf{n}) > 0$; otherwise it is called a *non-temporal* node.

Lemma 8. The following claims hold.

1. A temporal rule cannot be applied on a non-temporal node.
2. A DL rule cannot be applied on a temporal node.
3. Applying a DL rule on a non-temporal node cannot add a temporal node to a tableau.
4. Let $\mathbf{s}$ be a node which is added to a branch by applying a temporal rule on a temporal node (say $\mathbf{n}$). $|\Phi(\mathbf{s})| < |\Phi(\mathbf{n})|$.
5. Due to the application of a rule (whether a temporal rule or a DL rule) on a node residing on a branch, one or more nodes is added to the branch.

Proof. Let $\mathbf{m}$ (resp. $\mathbf{n}$) be a temporal (resp. non-temporal) node residing on a branch of a tableau. Also, let φ_0 and φ_1 be two formulas; let D be a concept description; let a, b be two individuals; let R be a DL role, and let $k \in \mathbb{N}$.

1. Since the temporal degree of $\mathbf{n} = 0$, $\Phi(\mathbf{n}) \in \{(a : D)_k, R(a, b)_k\}$. Because of the form of $\Phi(\mathbf{n})$, none of the temporal rules is applicable to $\mathbf{n}$ (see Definition 36 on Page 81).
2. Since the temporal degree of $\mathbf{m} > 0$, $\Phi(\mathbf{m}) \in \{\varphi_0 \wedge \varphi_1, \psi_0 \vee \psi_1, \Diamond_r^* \varphi_0, \Diamond_r^+ \varphi_0, \Diamond_r^- \varphi_0, \Diamond_l^* \varphi_0, \Diamond_l^+ \varphi_0, \Diamond_l^- \varphi_0\}$. Because of the form of $\Phi(\mathbf{n})$, none of the DL rules is applicable to $\mathbf{n}$ (see Definition 36).
3. Let $\mathbf{n}_0$ be a node which is added to a branch by applying a DL rule to $\mathbf{n}$. Based on the definition of DL rules, $\Phi(\mathbf{n}_0) \in \{(a : D)_k, R(a, b)_k\}$; so the temporal degree of $\mathbf{n}_0 = 0$, which indicates that $\mathbf{n}_0$ is certainly a non-temporal node.

4. Since, based on the definition of temporal rules, $\Phi(\mathbf{s})$ is a subformula of $\Phi(\mathbf{n})$, and $\Phi(\mathbf{s}) \neq \Phi(\mathbf{n})$, we have $|\Phi(\mathbf{s})| < |\Phi(\mathbf{n})|$.

5. Obvious from the definition of expansion rules (see Definition 36). □

First we prove that the number of temporal nodes is finite. Then, we will prove that the number of non-temporal nodes is also finite.

Lemma 9. Let ψ be a formula. Temporal rules can be applied only finitely often during the construction of a tableau for ψ.

Proof. As a corollary of Lemma 8, the only way to add a temporal node to a branch is to apply a temporal rule on a temporal node. Also, based on Lemma 8 (claims 4,5), when a rule is applied on a node (say $\mathbf{n}$), it always adds one or two nodes to the tableau each containing a shorter formula than the formula in $\mathbf{n}$ to which the rule was applied. Therefore, since the length of ψ is finite, the sequence of the application of temporal rules is finite. □

Corollary 2. The number of temporal nodes in a tableau is finite.

Corollary 3. The number of non-temporal nodes added by the application of temporal rules is finite.

We define below the notions of *temporal label* and *full label* for an individual occurring in the formula of a node. We use these notions in order to prove the finiteness of the number of non-temporal nodes.

Definition 40. Let B be a branch of a tableau; let D be a concept description; let a be an individual, and let $[i, j]$ be an interval.

- A *temporal label* of a on an interval $[i, j]$ w.r.t. B, denoted by $\mathcal{L}_B^{[i,j]}(a)$, is a multi-set $\{\mathrm{D} \mid (a : D)_k$ occurs as the formula in a node (say $\mathbf{n}$) on B, and $[i, j]$ is a subinterval of the interval in $\mathbf{n}$ $\}$.

- The *full label* of an individual a w.r.t. B (denoted by $\mathcal{L}_B(a)$) is a multi-set $\{D \mid D$ occurs in a temporal label of a w.r.t. $B\}$.

Example 8. Figure 5.1 (top part) exhibits some nodes of a branch of a tableau. Also, the bottom part of the figure represents the information contained in these nodes in a more intuitive manner. The temporal label of a on $[c_2, c_6]$ is $\{E \sqcap \forall R.F, \exists R.\exists S.D, E \sqcap \forall R.F\}$. The full label of a is $\{E \sqcap \forall R.F, \exists R.D, \exists R.\exists S.D, E \sqcap \forall R.F, C \sqcup D\}$.

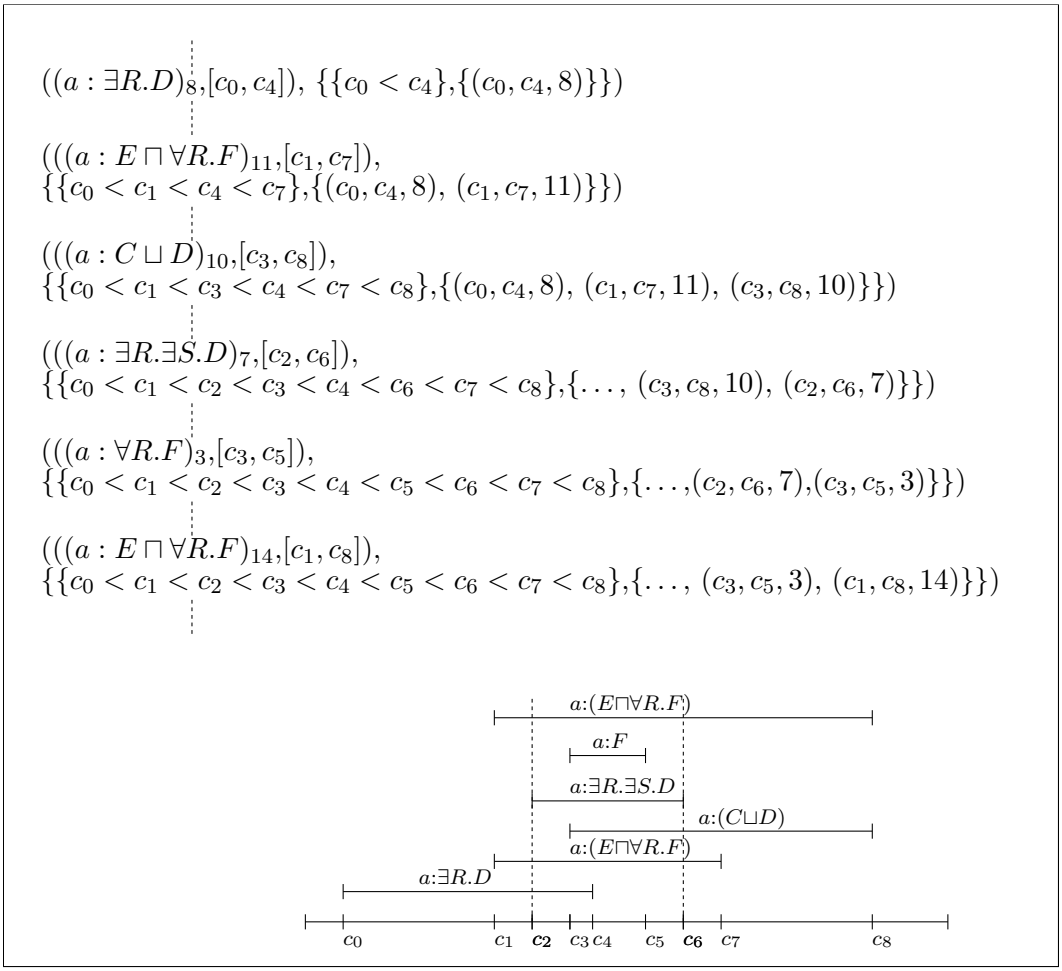

Figure 5.1: The temporal label of a on $[c_2, c_6]$ is $\{E \sqcap \forall R.F, \exists R.\exists S.D, E \sqcap \forall R.F\}$

Before we prove that the number of non-temporal nodes is finite, we need

to show that (1) the number of distinct intervals in the nodes of a tableau is finite, (2) a full label of an individual contains a finite number of concept descriptions, and (3) the number of individuals appearing in the nodes of a tableau is finite.

Lemma 10. The number of distinct intervals in the nodes of a tableau for a simple formula is finite.

Proof. Based on Lemma 9, the number of distinct intervals introduced by applying the temporal rules on the temporal nodes is finite. Let $\mathbf{n}$ be a node. We have two cases to consider: (1) Apply one of the rules $\{\mathcal{R}_{\sqcap}, \mathcal{R}_{\sqcup}, \mathcal{R}_{\exists}\}$ to $\mathbf{n}$, (2) Apply the $\mathcal{R}_{\forall}$ rule to $\mathbf{n}$.

1. Applying one of the rules $\{\mathcal{R}_{\sqcap}, \mathcal{R}_{\sqcup}, \mathcal{R}_{\exists}\}$ to $\mathbf{n}$ can add two nodes to a tableau. Based on the definition of these rules, the interval in each of the two added nodes is the same as the interval in $\mathbf{n}$.

2. The interval in a node added by applying $\mathcal{R}_{\forall}$ to $\mathbf{n}$ is a subinterval of the interval in $\mathbf{n}$. Since in this thesis we are assuming that time is discrete (see Remark 1 on Page 77), the number of distinct subintervals of an interval is finite.

Therefore, applying DL rules to the nodes of a tableau cannot introduce an infinite number of distinct intervals. □

Analogous to the notion of subformula of a formula in first order logic, we define below the notion of sub-description of a concept description.

Definition 41. Let D, E, F be three concept descriptions, and let R be a DL role. The sub-descriptions of D, denoted by $SubD(D)$, are recursively defined as follows:

- $SubD(D) = \{D\}$ where D is a concept name
- $SubD(D) = \{E\} \cup \{\mathrm{F}\} \cup SubD(E) \cup SubD(F)$ where $D = E \sqcap F$
- $SubD(D) = \{E\} \cup \{\mathrm{F}\} \cup SubD(E) \cup SubD(F)$ where $D = E \sqcup F$
- $SubD(D) = \{E\} \cup SubD(E)$ where $D = \exists R.E$
- $SubD(D) = \{E\} \cup SubD(E)$ where $D = \forall R.E$

Remark 5. Let D be a concept description. It is easy to show by induction that the cardinality of $SubD(D)$, denoted by $\#SubD(D)$, is less than or equal to $|D|$.

Lemma 11. Let ψ be a formula. The number of distinct (sub-) descriptions of the concept descriptions in ψ is less than $|\psi|$.

Proof. The total length of the concept descriptions in ψ is less than $|\psi|$. Therefore based on Remark 5, the number of distinct (sub-) descriptions of the concept descriptions in ψ is less than $|\psi|$. □

Consider the following facts derived from the definition of the expansion rules (Definition 36 on Page 81). We will use these facts in the proof of subsequent theorems.

- (Monotonicity) Applying a DL rule to a node adds one (or two) node(s) to a branch, and consequently adds a (two) new concept description(s) to some temporal labels of an individual, and never removes any concept description from any temporal label of the individual.
- (Shrinkage) Let $\mathcal{T}$ be a tableau; let B be a branch of $\mathcal{T}$; let **m, n** be two nodes; let a, b be two individuals; let D, E be two concept descriptions, and let k, $k^{'} \in \mathbb{N}$. Here, a (resp. k) can be equal to b (resp. $k^{'}$).

Assume $\mathbf{m}$ ($\Phi(\mathbf{m}) = (b : E)_{k'}$) is added to B by applying a DL rule to $\mathbf{n}$ ($\Phi(\mathbf{n}) = (a : D)_k$). Based on the definition of DL rules, $E \in SubD(D)$. If a non-temporal node $\mathbf{n}$ ($\Phi(\mathbf{n}) = (a : D)_k$) is added by applying a DL rule to a node, the concept description D is a sub-description of a concept description appearing in the root node of $\mathcal{T}$.

Lemma 12. Let ψ be a formula, and let $\mathcal{T}$ be a tableau for ψ. The number of distinct concept descriptions in the nodes of $\mathcal{T}$ is less than $|\psi|$.

Proof. Based on Shrinkage, every concept description in the labelled formula of the nodes of $\mathcal{T}$ is a (sub-) concept description of a concept in ψ. By Lemma 11, we may conclude that the number of distinct concept descriptions in the nodes of $\mathcal{T}$ is less than $|\psi|$. □

Before we prove the next theorem, we divide the individuals into the two categories.

1. Old individual: An individual which appeared in the formula of the root node.

2. New individual: An individual which is not an old individual. These individuals are created during the construction of a tableau.

Remark 6. Let ψ be a formula; the number of old individuals is less than $|\psi|$.

Below, we define the notion of a *repetitive node*. We use this notion during the proof of termination and the proof characterizing the complexity of the tableau algorithm.

Definition 42. Let $\mathbf{n}$, $\mathbf{m}$ be two non-temporal nodes s.t. $\mathbf{n}$ is a successor of $\mathbf{m}$, and both of them contain a concept assertion. $\mathbf{n}$ is a *repetitive node* if the labelled formula in $\mathbf{n}$ is exactly the same as the labelled formula in $\mathbf{m}$; otherwise $\mathbf{n}$ is a *non-repetitive* node.

Remark 7. All repetitive nodes occur in the situation described in Remark 3 (on Page 84); therefore it is not necessary to expand the repetitive nodes.

Obviously, we should consider all the nodes of a branch whether they are repetitive or not, in order to prove the termination and in order to find the complexity of the algorithm.

Theorem 8. Let B be a branch of a tableau for a simple formula; let D be a concept description. For a given individual a, the number of repetitive nodes on B that contain $a : D$ is at most twice the number of non-repetitive nodes on B that contain $a : D$.

Proof. Based on Lemma 12 and Lemma 10, the number of non-repetitive nodes on the branch B which contain $a : D$ is finite. Let w be the number of these nodes. By Remark 7, repetitive nodes cannot be expanded. As can be seen in the definition of the $\mathcal{R}_\forall$ and $\mathcal{R}_\exists$ rules, application of these rules does not add any repetitive nodes. Therefore, the only way to add a repetitive node is to apply the $\mathcal{R}_\wedge$ or $\mathcal{R}_\vee$ rule to the non-repetitive nodes which contain $a : D$. Based on the definition of these rules, only one rule is applicable to a node; then the node cannot be expanded again. Hence, the number of applications of the $\mathcal{R}_\wedge$ and $\mathcal{R}_\vee$ rules is w. Therefore, the number of repetitive nodes is at most $2w$. □

Corollary 4. For any individual a and any branch B, the full label of a w.r.t. B is finite.

Lemma 13. Let ψ be a simple formula, and let $\mathcal{T}$ be a tableau for ψ. The number of new individuals appearing in the nodes of $\mathcal{T}$ is finite.

Proof. Let a be an old individual; let b be a new individual; let R be a role, and let $k, k^{'} \in \mathbb{N}$. The number of non-temporal nodes that contain $R(a, b)_k$

is bounded by the total number of existential restrictions existing in all full labels of a. Therefore, the number of new individuals that are R-successors of the old individuals is bounded by the total number of existential restrictions in all full labels of old individuals. Let c be a new individual, and let S be a role. S (resp. k) can be equal to R (resp. $k^{'}$). For any S, the number of non-temporal nodes that contain $S(b,c)_{k'}$ is bounded by the total number of existential restrictions existing in all full labels of b. Therefore, the number of new individuals which are S-successors of the existing new individuals (the new individuals that are R-successors of the old individuals) is bounded by the number of existential restrictions in any full labels of existing new individuals. We can use similar reasoning to prove that the new individuals that are role-successors of other new individuals is also finite. Now, it remains to show that the successor chains of new individuals is finite. Since b is a new individual, it is only related to a with the relation R. In fact, b is not related to another individual or even to a by any relation other than R. Because of this fact and based on the definition of DL rules, the length of the maximal concept description in any full label of b is strictly less than the length of the maximal concept description in any full label of a; hence, since the length of a concept description is greater than zero, a successor chain of new individuals is finite. Therefore, the number of new individuals is finite.

□

Example 9. As can be seen in Figure 5.2, there is only one branch in the tableau. The full label of a w.r.t. the existing branch is $\{\exists R.\forall S.\exists P.D$, $\forall R.\forall P.(D \sqcup \exists S.(D \sqcap E))\}$ and the full label of b w.r.t. the existing branch is $\{\forall S.\exists P.D, \forall P.(D \sqcup \exists S.(D \sqcap E))\}$. Consider the two following points:

1. Obviously, the length of the maximal concept description in the full

label of b, i.e., $\forall P.(D \sqcup \exists S.(D \sqcap E))$ is strictly less than the length of the maximal concept description in the full label of a, i.e., $\forall R.\forall P.(D \sqcup \exists S.(D \sqcap E))$.

2. (Example of Shrinkage fact). The concept assertion $b : \forall P.(D \sqcup \exists S.(D \sqcap E))$ in Node 7 is achieved by applying the $\mathcal{R}_\forall$ rule to node 4, which contains $a : \forall R.\forall P.(D \sqcup \exists S.(D \sqcap E))$. Clearly, $\forall P.(D \sqcup \exists S.(D \sqcap E))$ is a sub-description of $\forall R.\forall P.(D \sqcup \exists S.(D \sqcap E))$.

$\boxed{1}[root]$ $((\Diamond_r^*((a : \exists R.\forall S.\exists P.D)_5 \wedge (a : \forall R.\forall P.(D \sqcup \exists S.(D \sqcap E)))_5),[c_0, c_1]),$
$\{\{c_0 < c_1\},\{(c_0, c_1, 2)\}\})$
$\quad\Big|\ \boxed{1\text{-R}_r^*}$
$\boxed{2}(((a : \exists R.\forall S.\exists P.D)_5 \wedge (a : \forall R.\forall P.(D \sqcup \exists S.(D \sqcap E)))_5,[c_1, c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0, c_1, 2),\ (c_1, c_2, 5)\}\})$
$\quad\Big|\ \boxed{2\text{-}\mathcal{R}_\wedge}$
$\boxed{3}(((a : \exists R.\forall S.\exists P.D)_5,[c_1, c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0, c_1, 2),\ (c_1, c_2, 5)\}\})$
$\quad\Big|\ \boxed{2\text{-}\mathcal{R}_\wedge}$
$\boxed{4}(((a : \forall R.\forall P.(D \sqcup \exists S.(D \sqcap E))_5,[c_1, c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0, c_1, 2),\ (c_1, c_2, 5)\}\})$
$\quad\Big|\ \boxed{3\text{-}\mathcal{R}_\exists}$
$\boxed{5}(((b : \forall S.\exists P.D)_5,[c_1, c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0, c_1, 2),\ (c_1, c_2, 5)\}\})$
$\quad\Big|\ \boxed{3\text{-}\mathcal{R}_\exists}$
$\boxed{6}((R(a, b)_5,[c_1, c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0, c_1, 2),\ (c_1, c_2, 5)\}\})$
$\quad\Big|\ \boxed{4\text{-}\mathcal{R}_\forall}$
$\boxed{7}((b : \forall P.(D \sqcup \exists S.(D \sqcap E)))_5,[c_1, c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0, c_1, 2),\ (c_1, c_2, 5)\}\})$
No more rules are applicable! The formula is satisfiable.

Figure 5.2: Tableau for $\Diamond_r^*((a : \exists R.\forall S.\exists P.D)_5 \wedge (a : \forall R.\forall P.(D \sqcup \exists S.(D \sqcap E)))_5)$

Theorem 9. Let ψ be a simple formula, and let B an arbitrary branch in a tableau for ψ. The number of non-temporal nodes on B is finite.

Proof. Let ψ be a formula; let a, b be two old individuals; let c, d be two new individuals; let D be a concept description; let R be a role, and let

$k, k' \in \mathbb{N}$. The finiteness of the number of non-temporal nodes on B is an easy consequence of the following facts.

- The number of non-temporal nodes that contain $R(a,b)_k$ is finite.
  Since no DL rule adds a node that contains a role assertion about two old individuals, such a role assertion should be a subformula of ψ in order to appear in the tableau. Therefore, the number of such non-temporal nodes is bounded by the number of non-temporal nodes which are added to a tableau by the application of temporal rules. Based on Corollary 3 (on Page 90), the number of non-temporal nodes that contain $R(a,b)_k$ is finite.

- The number of non-temporal nodes (on B) that contain a concept assertion (e.g., $a : D$) is finite. Since the number of individuals is finite, and the number of concept description in each full label of each individual is finite (Corollary 4 on Page 95), the number of non-temporal nodes that contain a concept assertion (e.g., $a : D$) is thus finite.

- The number of non-temporal nodes (on B) that contain $R(a,c)_{k'}$ (or $R(c,d)_{k'}$) is finite.
  Only $\mathcal{R}_\exists$ rules can add nodes that contain a role assertion between an old individual and a new individual (or two new individuals). Every application of the $\mathcal{R}_\exists$ rule introduces a new individual and adds two nodes to a branch. One node contains a concept assertion about the new individual and the other node contains a role assertion about that individual. The number of $\mathcal{R}_\exists$ rule applications is bounded by the number existential restrictions in the full labels of all individuals. Since the number of the individuals is finite (see Remark 6 (on Page 94) and Lemma 13 (on Page 95)), the number of non-temporal nodes (on B) that contain $R(a,c)_{k'}$ or

$R(c,d)_{k'}$ is finite. □

Proof. (Theorem 7 on Page 86.) Assume that there can be an infinite sequence of rule applications during the construction of a tableau. Also, the branching factor of the tableau is a finite number (see Lemma 7 on Page 88), and every rule application adds one (or more) nodes to the tableau. Therefore, there should be a branch with an infinite number of nodes. Since the number of temporal nodes as well as the number of non-temporal nodes residing on this branch are both finite, the number of nodes of the branch cannot be infinite. Therefore, there cannot be an infinite sequence of rule applications, and consequently, the tableau construction terminates.

□

5.3.3 Complexity of the tableau algorithm for simple MITDL formulas

Before we present our implementation of the tableau algorithm for checking the satisfiability of a simple formula, we prove the following theorems, which we will use to show that our implementation has PSPACE complexity.

Lemma 14. Let ψ be a formula; let B be a branch of a tableau for ψ. The number of temporal nodes on B is $\mathcal{O}(|\psi|)$.

Proof. The number of temporal rule applications for constructing B is equal to the number of temporal operators in ψ (see the definition of temporal rules).

This number is less than $|\psi|$. On the other hand, each temporal rule application may add at most two nodes to B. Hence, after applying the temporal rules, the number of existing nodes on B is $\mathcal{O}(|\psi|)$. □

Suppose that in order to construct a tableau for a simple formula we first exhaustively apply all the temporal rules and then we apply the DL rules. Now consider the following theorems.

Lemma 15. Let ψ be a formula. After applying the temporal rules (before applying the DL rules), the number of non-temporal nodes on a branch is $\mathcal{O}(|\psi|)$.

Proof. Applying a temporal rule may add at most two non-temporal nodes to the branch. By Lemma 14, before applying the DL rules, the number of non-temporal nodes on a branch is $\mathcal{O}(|\psi|)$. □

Lemma 16. Let ψ be a simple formula. The number of distinct intervals in the nodes of a branch is $\mathcal{O}(|\psi|^2)$.

Proof. Applying $\mathcal{R}_{\sqcup}$ or $\mathcal{R}_{\sqcap}$ does not introduce any new interval. Thus, the only way to have a new interval is to apply the $\mathcal{R}_{\forall}$ rule on a node. Before applying any DL rules, the number of non-temporal nodes is $\mathcal{O}(|\psi|)$ (see Lemma 15). This indicates that the number of distinct intervals in these non-temporal nodes is $\mathcal{O}(|\psi|)$. Based on the definition of $\mathcal{R}_{\forall}$, the interval in a node added by the application of the $\mathcal{R}_{\forall}$ rule is an intersection of two existing intervals. This means that by the application of the $\mathcal{R}_{\forall}$ rule at most $\mathcal{O}(|\psi|^2)$ new intervals are introduced. Therefore, the number of distinct intervals in the nodes is $\mathcal{O}(|\psi|^2)$. □

5.3.4 A PSPACE implementation of the tableau algorithm

In this section, we describe MITDL_SAT algorithm (Algorithm 1), an implementation of the tableau algorithm for checking the satisfiability of a simple formula. Given a formula, this non-deterministic algorithm decides whether the formula is satisfiable. Algorithm 1 uses a recursive routine, called SAT (Algorithm 2), which is responsible for adding new nodes that contain new individuals, and for expanding them as much as possible.

The algorithm expands a tableau in a depth first manner in the sense that it builds one branch of the tableau during the execution before turning to the next. While the positive output ("satisfiable") of the algorithm shows that the formula is satisfiable, the negative output ("not satisfiable") only indicates that the branch is closed.

Algorithm 1 PSPACE implementation of tableau algorithm

procedure MITDL_SAT(ψ)
    Build a tableau $\mathcal{T}$ which contains exactly one node (root node) (see Definition 28)
    while a temporal rule is applicable to $\mathcal{T}$ **do**
        Apply the rule, if it is a rule from $\{\mathcal{R}_{\vee}, \mathcal{R}_{\Diamond_r^-}, \mathcal{R}_{\Diamond_r^+}, \mathcal{R}_{\Diamond_l^-}, \mathcal{R}_{\Diamond_l^+}\}$, non-deterministically pick one choice and add a new node to $\mathcal{T}$
        if $\mathcal{T}$ contains a clash **then**
            return "not satisfiable"
    while a DL rule, with the exception of $\mathcal{R}_{\exists}$, is applicable to $\mathcal{T}$ **do**
        Apply the rule, if it is a $\mathcal{R}_{\sqcup}$ rule, non-deterministically pick one choice and add a new node to $\mathcal{T}$
        if $\mathcal{T}$ contains a clash **then**
            return "not satisfiable"
    Let S be a list of nodes on which $\mathcal{R}_{\exists}$ is applicable
    while $S \neq \emptyset$ **do**
        $n \leftarrow$ pick one of the elements in S
        $r \leftarrow$ the leaf of the branch
        if SAT(n,$\mathcal{T}$, r) = "not satisfiable" **then**
            return "not satisfiable"
        Remove n from S
    return "satisfiable"

Algorithm 2 Subroutine SAT

```
1: procedure SAT(n, 𝒯, rl)
2:     Apply ℛ∃ rule on n
3:     Apply the ℛ∀ rule on the existing nodes
4:     while a DL rule, with the exception of ℛ∃, is applicable to fresh nodes
       in 𝒯 do
5:         Apply the rule, if it is a ℛ⊔ rule, non-deterministically pick one
       choice and add a new node to 𝒯
6:         if 𝒯 contains a clash then
7:             return "not satisfiable"
8:     Let New be the list of the nodes on which ℛ∃ is applicable and they
       are successors of n
9:     while New ≠ ∅ do
10:        m ← pick one of the elements in New
11:        r ← the leaf of the branch
12:        if SAT(m,𝒯, r) = "not satisfiable" then
13:            return "not satisfiable"
14:        Discard the nodes which are successors of rl
15:        Remove m from New
```

First, all the temporal rules are applied (Algorithm 1 - Line 3-4). After this step, we have a branch that contains both temporal nodes and non-temporal nodes. Since no more temporal rules are applicable, temporal nodes cannot be expanded anymore. On the other hand, the temporal rules are not applicable to the non-temporal nodes (Lemma 8 on Page 89); therefore, after this step, the algorithm does not consider the application of temporal rules.

Algorithm 1 exhaustively expands the non-temporal nodes by applying DL rules with the exception of the $\mathcal{R}_\exists$ rule (Line 7-10); at the completion of Line 7-10, the tableau contains some (non-)temporal nodes which are not expandable and some non-temporal nodes, called old nodes, that can be expanded, in particular, by applying the $\mathcal{R}_\exists$ rule. Note that the tableau contains only one branch. This algorithm uses Algorithm 2, the SAT routine, to expand each of the old nodes in turn. More precisely, in each round, the routine expands an old node and the nodes (called fresh nodes) that are added to the branch by the routine. First, the $\mathcal{R}_\exists$ rule is applied to the node that is passed as the input parameter of the routine (Line 2). Then the $\mathcal{R}_\forall$ is applied on all nodes of the branch (Line 3); this may add some fresh nodes to the branch. Afterward, all DL rules (except $\mathcal{R}_\exists$) are applied on the fresh nodes (Line 4-5). If there is a fresh node that contains an existential restriction in its formula, SAT is recursively called to expand that node as well. If during the expansion of a node a clash is detected, the algorithm returns "not satisfiable", which means the branch is closed, and we should execute MITDL_SAT again in order to investigate the other possible choices which were not considered during the previous execution of the algorithm. If all the old nodes are successfully expanded and no clash is detected, the algorithm returns "satisfiable" which indicates that the formula is satisfiable. Note that, based on Theorem 7 (on Page 86), this recursion always terminates.

Theorem 10. (PSPACE Complexity Theorem.) The complexity of the MITDL_SAT algorithm (resp. the SAT algorithm) is PSPACE.

Before we prove this theorem, we analyze the MITDL_SAT and SAT algorithms in detail.

- Line 1-6 of Algorithm 1: Based on Lemma 14 (on Page 99), the number of the nodes which are added to the tableau by executing these lines is $\mathcal{O}(|\psi|)$.

Theorem 11. Let ψ be a simple formula; let B be a branch, and let a be an old individual. Before the execution of Line 11 of Algorithm 1, the cardinality of the full label of a w.r.t. B is $\mathcal{O}(|\psi|^3)$.

Proof. Since the number of distinct intervals is $\mathcal{O}(|\psi|^2)$ (Lemma 16 on Page 100) and the number of distinct concept descriptions is less than $|\psi|$ (Lemma 12 on Page 94), the number of non-repetitive nodes which contain a concept assertion about a is $\mathcal{O}(|\psi|^3)$. Note, the number of repetitive nodes (Theorem 8 on Page 95) does not change the complexity of the cardinality of the full label of a. □

- Line 7-10 of Algorithm 1: Based on Remark 6 (on Page 94) and Theorem 11, the number of possible non-temporal nodes that can be added to the branch by applying DL rules (except $\mathcal{R}_\exists$) is $\mathcal{O}(|\psi|^4)$.

- Line 11 of Algorithm 1: Since the number of existing nodes is $\mathcal{O}(|\psi|^4)$, the cardinality of S is $\mathcal{O}(|\psi|^4)$.

- Line 12-17 of Algorithm 1: In order to find the complexity of executing these lines, we should analyze the complexity of executing the SAT routine. We will show that the space complexity of the SAT is PSPACE.

– Line 2 of Algorithm 2: Applying the $\mathcal{R}_\exists$ rule always adds two nodes to the branch. Based on the definition of the $\mathcal{R}_\exists$ rule, the formulas in these nodes contain a new individual (say b).

– Line 3-7 of Algorithm 2: Since the number of distinct intervals is $\mathcal{O}(|\psi|^2)$ (Lemma 16 on Page 100), and the number of distinct concept descriptions is less than $|\psi|$ (Lemma 12 on Page 94), the number of non-repetitive nodes which contain a concept assertion about b is $\mathcal{O}(|\psi|^3)$. Note, the number of repetitive nodes (Theorem 8 on Page 95) does not change the complexity of the number of the nodes which are added by the execution of these lines. Therefore, the number of the nodes added by the execution of these lines is $\mathcal{O}(|\psi^3|)$. Since $\mathcal{O}(|\psi|^4) + \mathcal{O}(|\psi|^3) = \mathcal{O}(|\psi|^4)$, the number of existing nodes is still $\mathcal{O}(|\psi|^4)$.

– Line 8 of Algorithm 2: Since the number of existing nodes is $\mathcal{O}(|\psi|^4)$, the cardinality of S is $\mathcal{O}(|\psi|^4)$.

– Line 9-14 of Algorithm 2: The number of recursive calls of SAT is bounded by the length of the longest successor chain of existential restrictions existing in the full label of b; this length is bounded by the length of ψ. This means that the algorithm adds at most $|\psi| * \mathcal{O}(|\psi|^3) = \mathcal{O}(|\psi|^4)$ nodes to the tableau. Note, SAT discards the unnecessary nodes in order to use the memory for the executing rest of algorithm.

Proof. (Theorem 10.) Based on the above analysis, the number of the nodes in the tableau is $\mathcal{O}(|\psi|^4)$. Since the memory required for storing a node is $\mathcal{O}(|\psi|)$, the memory required for checking satisfiability of ψ is $\mathcal{O}(|\psi|^5)$. Therefore, the complexity of Algorithm 1 (resp. Algorithm 2) is PSPACE. □

5.3.5 Tableau-based algorithm for checking the satisfiability of a generic MITDL formula

In this section, we explain how the previous algorithm should be modified in order to have an algorithm for checking the satisfiability of a generic formula. In fact, most of the definitions and the other details are exactly same as before. Hence, we explain the differences here.

Let D be a concept description, and let $k \in \mathbb{N}$. Since a formula may contain a general conclusion as a subformula (e.g., $(D = \top)_k$), the algorithm presented in Section 5.3.1 must be augmented; first we propose that we add the $\mathcal{R}_=$ rule to the DL rules (Definition 36 on Page 81) of the algorithm.

$\mathcal{R}_=$: If $\psi = (D = \top)_k$, then

- if there is a node (say $\mathbf{m_1}$) on the branch B s.t. $\mathbf{m_1} = (((a : E)_{k'}, [c_{i_0}, c_{j_0}]), \mathbb{C}_\mathcal{B})$ (resp. $\mathbf{m_1} = (((a : \neg E)_{k'}, [c_{i_0}, c_{j_0}]), \mathbb{C}_\mathcal{B})$) where $[c_i, c_j] \cap [c_{i_0}, c_{j_0}] = [c_{i_1}, c_{j_1}] \neq \emptyset$, and there is no node (say $\mathbf{m_2}$) on the branch s.t. $\mathbf{m_2} = (((a : D)_{k''}, [c_{i_2}, c_{j_2}]), \mathbb{C}_\mathcal{B})$ and $[c_{i_1}, c_{j_1}] \subseteq [c_{i_2}, c_{j_2}]$, expand the branch to $B.\mathbf{n}_0$, s.t. $\mathbf{n}_0 = ((a : D)_{k'''}, [c_{i_1}, c_{j_1}])$, $\mathbb{C}_{\mathcal{B}_0})$ where $k''' = |c_{j_1} - c_{i_1}|$. $\mathbb{C}_{\mathcal{B}_0}$ is equal to $\mathbb{C}_\mathcal{B}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting (c_{i_1}, c_{j_1}, k''') in $L_{\mathbb{C}_\mathcal{B}}$.

However if we check the satisfiability of $\psi = (a : D)_5 \wedge (\exists R.C = \top)_5$ with this algorithm, we will see that the algorithm does not terminate. Let $\kappa = 10$. Figure 5.3 exhibits a fragment of a tableau for ψ. As seen in the figure, the $\mathcal{R}_\exists$ rule generates a new individual, then the $\mathcal{R}_=$ rule is applied on the added node. Afterwards, the $\mathcal{R}_\exists$ rule generates another new individual, and the $\mathcal{R}_=$ rule is applied again. The b_is ($i \in \mathbb{N}$) in the figure have the same type in the sense that all of them are individuals of the same concept. The process of

generating a new individual and applying the $\mathcal{R}_=$ rule never ends; we say the algorithm has run into a cycle. Clearly, the algorithm never terminates.

Let ψ be a formula; let D, E be two concept descriptions; let R be a DL role, and let $k \in \mathbb{N}$. If both of the following conditions hold, the algorithm may run into a cycle.

1. $(D = \top)_k$ is a subformula of ψ.

2. D is equal to $\exists R.E$ or $\exists R.E$ is a sub-description of D.

If a formula ψ does not contain a subformula $(D = \top)_k$, ψ would be a simple formula, and because of the shrinkage fact (see Page 93), the algorithm never runs into a cycle. The only rule which produces a new individual is the $\mathrm{R}_\exists$ rule; therefore, if the formula ψ is equal to $\exists R.E$ or contains $\exists R.E$, the application of the $\mathrm{R}_=$ rule adds one or more nodes to the tableau s.t. the expansion of the added nodes will introduce some new individuals. By the definition, the $\mathrm{R}_=$ rule is again applicable because of the presence of the new individuals. Then, the expansion of the nodes added by application of $\mathrm{R}_=$ will introduce more new individuals and this process never ends. The above conditions are not sufficient conditions for having a cycle in a tableau. For example, a tableau for $\psi = (a : \exists R.E)_5 \wedge \Diamond_r((\exists R.E = \top)_6)$ does not run into a cycle while, obviously, the both conditions hold. We leave to the reader to investigate this situation.

In order to ensure termination of the tableau algorithm, we will define the notion of Temporal Subset Blocking (TSB) based on the notion of subset blocking [46] (see Definition 6 on Page 24). TSB detects situations in which a cycle may occur and prevents it if it occurs. In fact, the idea is to use an existing individual instead of generating a new individual. In other word, if an individual b is blocked by a, we can use an existing role successor (say

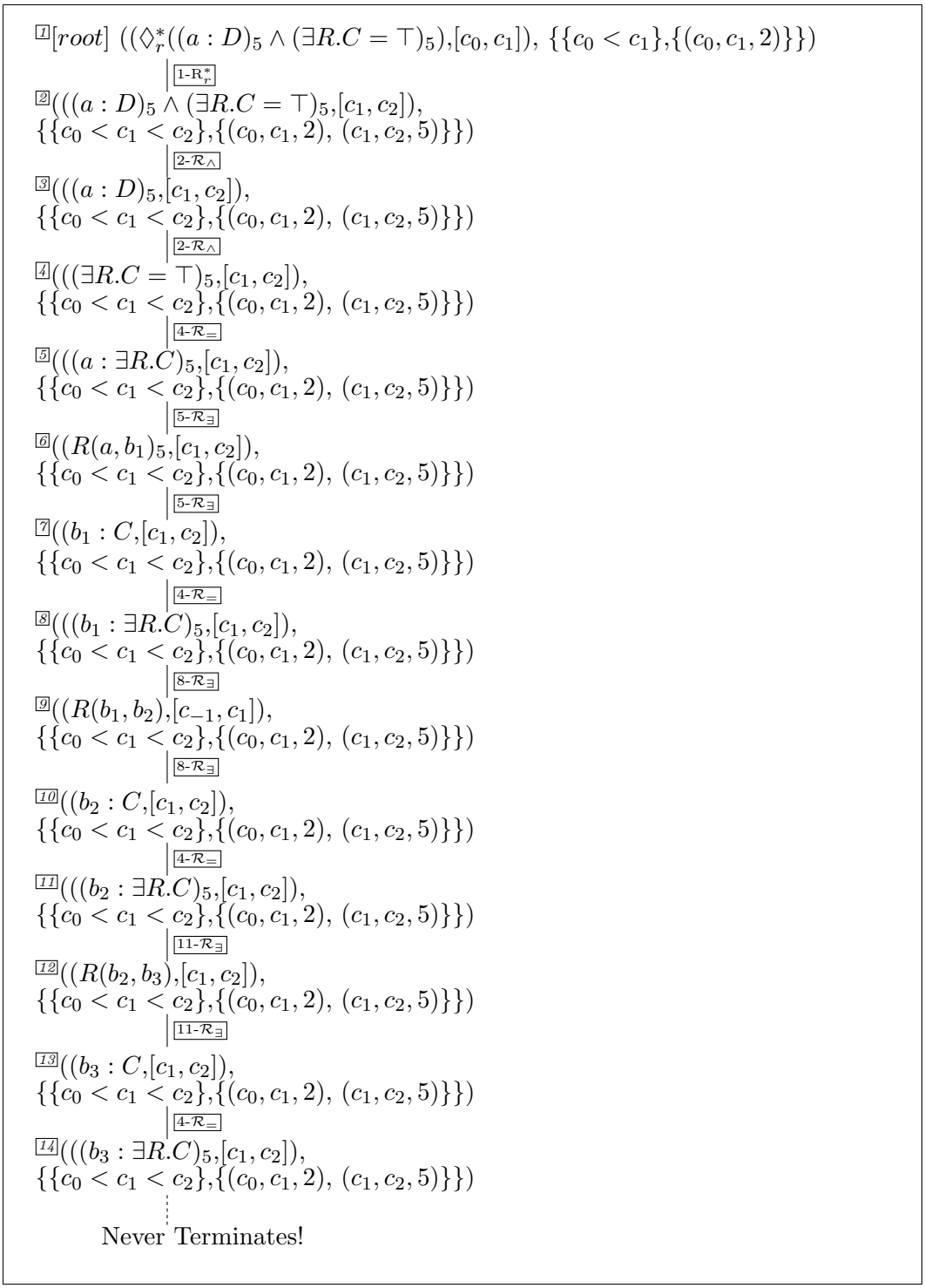

Figure 5.3: Tableau for $\Diamond_r^*((a : D)_5 \wedge (\exists R.C = \top)_5)$

R-successor) of a rather than generating a new R-successor of b. For example in Figure 5.3, instead of generating a new R-successor for b_2 (i.e., b_3), we can use the R-successor of b_1 (i.e., b_2). This yields a class of models for ψ, i.e, $(\mathbb{D}, \mathbb{I}^-(\mathbb{D}), \mathbb{S})$ where $\mathbb{D} = \{c_1, c_2\}$ s.t. $c_1 < c_2$, $\mathbb{I}^-(\mathbb{D}) = [c_1, c_2]$ and $\mathbb{S}[c_1, c_2]$ defined as follows:

- $\Delta^{[c_1,c_2]} = \{a, b_1, b_2\}$
- $D^{[c_1,c_2]} = \{a\}$
- $C^{[c_1,c_2]} = \{b_1, b_2\}$
- $R^{[c_1,c_2]} = \{(a, b_1), (b_1, b_2), (b_2, b_2)\}$

A problem which sometimes arises is *cyclic blocking*, i.e., blocking an individual a by an individual b and vice versa. Suppose that a is blocked by b. As we will see later in Definition 44, the nodes which contain concept assertions about a are not expanded, because the expansion of the nodes which contain concept assertions about b add sufficient information for checking the satisfiability of the formula in the initial node. If b is also blocked by a, the nodes which contain concept assertions about b are not expanded and some information is lost. Therefore, cyclic blocking should be avoided.

We will enumerate all individual names to avoid cyclic blocking of individuals in the algorithm. A necessary condition for blocking a by b is that a appears after b in the enumeration: we write $b \prec\prec a$ [46]. Now, consider the following definition.

Definition 43. (Temporal Subset Blocking) Let B be a branch; let D, E be two concept descriptions; let a, b be two individuals, and let $[c_i, c_j], [c_{i_0}, c_{j_0}], [c_{i_1}, c_{j_1}]$ be three intervals.

a is *blocked* on $[c_i, c_j]$ by b if

$$\{D \mid (((a:D)_k, [c_{i_0}, c_{j_0}]), \mathbb{C}_{\mathcal{B}}) \in B \text{ and } [c_i, c_j] \subseteq [c_{i_0}, c_{j_0}]\}$$
$$\subseteq$$
$$\{E \mid (((b:E)_{k'}, [c_{i_1}, c_{j_1}]), \mathbb{C}_{\mathcal{B}}) \in B \text{ and } [c_i, c_j] \subseteq [c_{i_1}, c_{j_1}] \}$$
$$\text{and } b \prec\prec a \text{ in the enumeration.}$$

Below we redefine the DL rules. The new definitions of DL rules (with the exception of the $\mathcal{R}_=$ rule) are same as the definitions of the rules in Definition 36 (on Page 81) except that we add the assumption that the individual a should not be blocked on $[c_i, c_j]$.

Definition 44. DL rules:

$\mathcal{R}_=$: If $\psi = (D = \top)_k$, then

- If there is a node (say $\mathbf{m_1}$) on the branch B s.t. $\mathbf{m_1} = (((a : E)_{k'}, [c_{i_0}, c_{j_0}]), \mathbb{C}_{\mathcal{B}})$ (resp. $\mathbf{m_1} = (((a : \neg E)_{k'}, [c_{i_0}, c_{j_0}]), \mathbb{C}_{\mathcal{B}})$) where a is a non-blocked node on $[c_{i_0}, c_{j_0}]$ and $[c_i, c_j] \cap [c_{i_0}, c_{j_0}] = [c_{i_1}, c_{j_1}] \neq \emptyset$, and there is no node (say $\mathbf{m_2}$) on the branch s.t. $\mathbf{m_2} = (((a : D)_{k''}, [c_{i_2}, c_{j_2}]), \mathbb{C}_{\mathcal{B}})$ and $[c_{i_1}, c_{j_1}] \subseteq [c_{i_2}, c_{j_2}]$, expand the branch to $B.\mathbf{n}_0$, s.t. $\mathbf{n}_0 = ((a : D)_{k'''}, [c_{i_1}, c_{j_1}]), \mathbb{C}_{\mathcal{B}_0})$ where $k''' = |c_{j_1} - c_{i_1}|$. $\mathbb{C}_{\mathcal{B}_0}$ is equal to $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_0}}$ is obtained by inserting (c_{i_1}, c_{j_1}, k''') in $L_{\mathbb{C}_{\mathcal{B}}}$.
- If no other expansion rule (with the exception of the $\mathcal{R}_=$) is applicable, then let $u(\mathbf{n}, B) = 1$;

$\mathcal{R}_\sqcap$: If $\psi = (a : D \sqcap E)_k$ and a is not blocked on $[c_i, c_j]$ then if B does not contain both (or one of the) two nodes (say $\mathbf{n}_0$,$\mathbf{n}_1$) s.t. $\mathbf{n}_0 = (((a : D)_k, [c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1 = (((a : E)_k, [c_i, c_j]), \mathbb{C}_{\mathcal{B}})$, then expand the branch to $B.\mathbf{n}_0.\mathbf{n}_1$, s.t. $\mathbf{n}_0 = (((a : D)_k, [c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1 = (((a : E)_k, [c_i, c_j]), \mathbb{C}_{\mathcal{B}})$. Let $u(\mathbf{n}, B) = 1$;

$\mathcal{R}_{\sqcup}$: If $\psi = (a : D \sqcup E)_k$ and a is not blocked on $[c_i, c_j]$ then if B does not contain both two nodes (say $\mathbf{n}_0$,$\mathbf{n}_1$) s.t. $\mathbf{n}_0 = (((a : D)_k,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1 = (((a : E)_k,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$, then expand the branch to $B.\mathbf{n}_0|\mathbf{n}_1$, s.t. $\mathbf{n}_0 = (((a : D)_k,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1 = (((a : E)_k,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\exists}$: If $\psi = (a : \exists R.D)_k$ and a is not blocked on $[c_i, c_j]$ then if there is no individual (say b) s.t. two nodes (say $\mathbf{m}$, $\mathbf{s}$) where $\mathbf{m} = (((R(a, b))_{k'},[c_{i_0}, c_{j_0}]), \mathbb{C}_{\mathcal{B}})$ where $[c_i, c_j] \subseteq [c_{i_0}, c_{j_0}]$ and $\mathbf{s} = (((b : D)_{k''},[c_{i_1}, c_{j_1}]), \mathbb{C}_{\mathcal{B}})$ where $[c_i, c_j] \subseteq [c_{i_1}, c_{j_1}]$ exist in the branch B, then expand the branch to $B.\mathbf{n}_0.\mathbf{n}_1$, s.t. $\mathbf{n}_0 = ((R(a, b)_k,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}_1 = (((b : D)_k,[c_i, c_j]), \mathbb{C}_{\mathcal{B}})$ [113]. Note, individual b is not appeared elsewhere in the tree. In other word, it is a new individual name. Let $u(\mathbf{n},B) = 1$;

$\mathcal{R}_{\forall}$: If $\psi = (a : \forall R.D)_k$ and a is not blocked on $[c_i, c_j]$ then if there is a node (say $\mathbf{m}$) in branch B s.t. $\mathbf{m} = (((R(a, b))_{k'},[c_{i_0}, c_{j_0}]), \mathbb{C}_{\mathcal{B}})$ where $[c_i, c_j] \cap [c_{i_0}, c_{j_0}] = [c_{i_2}, c_{j_2}] \neq \emptyset$, and there is no node (say $\mathbf{l}$) in branch B s.t. $\mathbf{l} = (((b : D)_{k'''},[c_{i_3}, c_{j_3}]), \mathbb{C}_{\mathcal{B}})$ where $[c_{i_3}, c_{j_3}] \subseteq [c_{i_2}, c_{j_2}]$, then expand the branch to $B.\mathbf{n}_0$, s.t. $\mathbf{n}_0 = (((b : D)_{k''},[c_{i_2}, c_{j_2}]), \mathbb{C}_{\mathcal{B}})$ where $k'' = |c_{j_2} - c_{i_2}|$. $\mathbb{C}_{\mathcal{B}_2}$ is equal to $\mathbb{C}_{\mathcal{B}}$ and $L_{\mathbb{C}_{\mathcal{B}_2}}$ is obtained by inserting (c_{i_2}, c_{j_2}, k'') in $L_{\mathbb{C}_{\mathcal{B}}}$.

By the definition of DL rules applying the $\mathcal{R}_{=}$ rule and the $\mathcal{R}_{\forall}$ rule may add many new nodes to the branch. Also, note that in all DL Rules (with the exception of the $\mathcal{R}_{=}$ rule), if a is a blocked individual, let $u(\mathbf{n},B) = 1$.

5.3.6 Soundness of the tableau algorithm for generic MITDL formulas

Definition 45. Given a set S of labeled formulas with labels in $\mathbb{C}$, we say that S is satisfiable over $\mathbb{C}$ if there exists a strict model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}^{-}(\mathbb{D}), \mathbb{S} \rangle$ such that $\mathbb{D}$ (see Definition 30 on Page 69) is an extension of $\mathbb{C}$ and M,$[c_i, c_j] \models \psi$ for all $(\psi, [c_i, c_j]) \in S$.

Theorem 12. (Soundness). If $\psi \in$ MITDL and a tableau $\mathcal{T}$ for the annotated version of $\Diamond_r \psi$ is closed, then ψ is not satisfiable.

Proof. Let $\mathbb{C}$ be the interval structure in $\mathbf{n}$. $P(m)$ is the statement: if the following conditions hold:

1. $\mathbf{n}$ is a node;
2. the height of $\mathbf{n}$ is m;
3. every branch through $\mathbf{n}$ is closed;

then set $S(\mathbf{n})$ of all labeled formulas in the nodes between $\mathbf{n}$ and the root is not satisfiable over $\mathbb{C}$. We will prove $P(m)$ is true for all $m \geq 0$ using strong induction. We present the general sketch of the induction here. We refer the reader to Appendix B for the details of proof.

(Base case) If $m = 0$, then $\mathbf{n}$ is a leaf, and the unique branch B containing $\mathbf{n}$ is closed. Then, we have one of the following cases.

1. $S(\mathbf{n})$ contains the labeled formula $(p_s, [c_k, c_{k_0}])$ and $(c_{k_0}, c_k, s) \in L_{\mathbb{C}}$ and $|c_k - c_{k_0}| \neq s$ where p_s is an atomic formula.
2. $S(\mathbf{n})$ contains both the labeled formulas $(a : C, [c_{k_1}, c_{l_1}])$ and $(a : \neg C, [c_{k_2}, c_{l_2}])$ where $([c_{k_1}, c_{l_1}] \cap [c_{k_2}, c_{l_2}] \neq \emptyset)$.

In both cases, we are not able to construct a model for the labeled formulas in set S($\mathbf{n}$). Hence, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

(Induction Case) Assume $P(m)$ holds for all m, $0 \leq m \leq t$. We want to prove $P(t+1)$ holds. Suppose the height of $\mathbf{n}$ is $t+1$ and C=$\{c_0,...,c_n\}$. There are two cases to consider; (1) when $\mathbf{n}$ is the direct successor which results after applying the $\mathcal{R}_\theta$ ($\theta \in \{\wedge, \sqcap, \exists\}$) rule on node $\mathbf{g}$ s.t. $\mathbf{g} = (\psi,[c_i, c_j], \mathbb{C})$, and (2) when an expansion rule is applied to $\mathbf{n}$ s.t. $\mathbf{n} = ((\psi,[c_i, c_j]), \mathbb{C})$ or an expansion rule is applied to some labeled formula $(\psi,[c_i, c_j]) \in S(\mathbf{n}) - \{\Phi(\mathbf{n})\}$ (i.e. the existing formula in $\mathbf{n}$) to extend the branch at $\mathbf{n}$. In both cases $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$ (see the details in Appendix B).

□

5.3.7 Completeness of the tableau algorithm for generic MITDL formulas

Theorem 13. (Completeness). If $\psi \in$ MITDL and a tableau $\mathcal{T}$ for the annotated version of $\Diamond_r\psi$ is open, then ψ is satisfiable.

General sketch of proof. Let B be an open branch in $\mathcal{T}$. We first define a set of interpretations $\mathbb{S}$ using existing information on the branch, so that, $\mathbb{S}$ satisfies all the role assertions on B. Then we use strong induction to show that $\mathbb{S}$ also satisfies all the concept assertions on B. Finally, we let $\mathbb{D}$ be a partially ordered set which consists of all points (both start points and end points of intervals) appearing on B and use strong induction to prove that a three tuple $M = (\mathbb{D}, \mathbb{I}^-(\mathbb{D}), \mathbb{S})$ is a model for the annotated version of $\Diamond_r\psi$.

Proof in detail. Recall that φ, $\Diamond_r\varphi$ and the annotated version of $\Diamond_r\varphi$ are equisatisfiable. Also, recall we are working in the context of expanding domains, i.e., $\Delta^{[i,j]} \subseteq \Delta^{[c,d]}$ where $d \geq j$ and $i \geq c$ (Remark 2 on Page 77). We prove the

theorem by constructing a class of models; we define a set of interpretations $\mathbb{S}$ where $\mathbb{S}[i,j]$ (see Section 5.2) is as follows.

1. $\Delta^{[i,j]}$=$\{\ a \mid ((a : D)_k, [i^{'}, j^{'}]) \in B$ or $((a : \neg D)_k, [i^{'}, j^{'}]) \in B$ where $[i,j] \cap [i^{'}, j^{'}] \neq \emptyset\ \}$

2. For all atomic concepts C, we define $C^{[i,j]}$=$\{\ a \mid ((a : C)_k, [i^{'}, j^{'}]) \in B$ where $[i,j] \subseteq [i^{'}, j^{'}]\ \}$

3. For all roles R, we define $R^{[i,j]} = \{\ (a,b) \mid (R(a,b)_k, [i^{'}, j^{'}]) \in B$ where $[i,j] \subseteq [i^{'}, j^{'}]\ \}$

By this definition, $\mathbb{S}[i,j] \models a : C$ where C is an atomic concept, and $((a : C)_k, [i^{'}, j^{'}]) \in B$ for $[i,j] \subseteq [i^{'}, j^{'}]$, and $\mathbb{S}[i,j] \models R(a,b)$ where $(R(a,b)_k, [i^{''}, j^{''}]) \in B$ and $[i,j] \subseteq [i^{''}, j^{''}]$. Clearly, the above definition satisfies the definition of expanding domain. Note that the length of a concept description is the cardinality of the multi set of DL-operators occurring in it. Let B be an open branch in $\mathcal{T}$; let $[i,j]$ be an interval; $P(m)$ is the statement: let D be a concept description of length m and suppose $((a : D)_k, [i^{'}, j^{'}]) \in B$ (respectively, $((a : D)_k, [i^{'}, j^{'}]) \notin B$) where $[i,j] \subseteq [i^{'}, j^{'}]$; then $\mathbb{S}[i,j] \models a : D$ (respectively, $\mathbb{S}[i,j] \models a : \neg D$). We use strong induction to prove $P(m)$ is true for all $m \geq 0$. (Base case) If $m = 0$, then D is an atomic concept and the claim is true by the definition of $D^{[i,j]}$. If $m = 1$, then $D = \neg C$ where C is an atomic concept and since $((a : \neg C)_k, [i^{'}, j^{'}]) \in B$ follows that $((a : C)_k, [i^{'}, j^{'}]) \notin B$, by the definition of $C^{[i,j]}$, $a \notin C^{[i,j]}$; hence $a \in (\Delta^{[i,j]} - C^{[i,j]})$ and the claim is true. (Induction case) Assume $P(m)$ is true for all $0 \leq m \leq t$. We prove $P(t+1)$ is true. Consider the following cases: Assume $[i,j] \subseteq [i^{'}, j^{'}]$.

- If $D = \neg E$, we have $((a : \neg E)_k, [i^{'}, j^{'}]) \in B$. It follows that $((a : E)_k, [i^{'}, j^{'}]) \notin B$ and since the length of E is less than the length of $\neg E$, by the induction assumption, $\mathbb{S}[i,j] \models a : \neg E$.

- If $D = (E \sqcap F)$, we have $((a : E \sqcap F)_k, [i^{'}, j^{'}]) \in B$. Then by the definition of the expansion rules we have $((a : E)_k, [i^{'}, j^{'}]) \in B$ and $((a : F)_k, [i^{'}, j^{'}]) \in B$. Since the length of E and the length of F are both less than the length of D, by the induction assumption, $\mathbb{S}[i, j] \models a : E$ and $\mathbb{S}[i, j] \models a : F$; therefore $\mathbb{S}[i, j] \models a : D$.

- If $D = (E \sqcup F)$, we have $((a : E \sqcup F)_k, [i^{'}, j^{'}]) \in B$. Then by the definition of the expansion rules we have $((a : E)_k, [i^{'}, j^{'}]) \in B$ or $((a : F)_k, [i^{'}, j^{'}]) \in B$. Since the length of E and the length of F are both less than the length of D, by the induction assumption, $\mathbb{S}[i, j] \models a : E$ or $\mathbb{S}[i, j] \models a : F$; therefore $\mathbb{S}[i, j] \models a : D$.

- If $D = (\exists R.E)$, we have $((a : \exists R.E)_k, [i^{'}, j^{'}]) \in B$. Then by the definition of the expansion rules we have $((R(a, b))_k, [i^{'}, j^{'}]) \in B$ and $((b : E)_k, [i^{'}, j^{'}]) \in B$. Since the length of E is less than the length of D, by the induction assumption, $\mathbb{S}[i, j] \models b : E$. Also, $\mathbb{S}[i, j] \models R(a, b)$ by the definition $\mathbb{S}[i, j]$; therefore, we have $\mathbb{S}[i, j] \models a : D$.

- If $D = (\forall R.E)$, we have $((a : \forall R.E)_k, [i^{'}, j^{'}]) \in B$. Then by the definition of the expansion rules if we have $((R(a, b))_k, [i^{'}, j^{'}]) \in B$ then $((b : E)_k, [i^{'}, j^{'}]) \in B$. $\mathbb{S}[i, j] \models R(a, b)$ by the definition $\mathbb{S}[i, j]$. Since the length of E is less than the length of D, by the induction assumption, $\mathbb{S}[i, j] \models b : E$; therefore, we have $\mathbb{S}[i, j] \models a : D$.

A three tuple $M = (\mathbb{D}, \mathbb{I}^{-}(\mathbb{D}), \mathbb{S})$ is a model for the annotated version of $\Diamond_r \psi$ denoted by γ. $P(m)$ is the statement: Let B be an open branch in a tableau for γ; let $[i, j]$ be an interval; let φ be a MITDL formula which has m occurrences of temporal operators, and suppose $(\varphi, [i, j]) \in B$; then $M, [i, j] \models \varphi$. We use strong induction to prove $P(m)$ is true for all $m \geq 0$.

Assume that $[i,j] \subseteq [i^{'}, j^{'}]$, $[i,j] \cap [i^{''}, j^{''}] = [i^{'''}, j^{'''}] \neq \emptyset$, $k = |j-i|$, $k^{'} = |j^{'} - i^{'}|$ and $k^{'''} = |j^{'''} - i^{'''}|$.

(Base case) If $m = 0$, then consider the following cases; where D is a concept description,

- Let $\varphi = (a : D)_k$; we have $((a : D)_k, [i,j]) \in B$. By previous induction, $\mathbb{S}[i,j] \models a : D$. Since we have $\mathbb{S}[i,j] \models a : D$ and $k = |j - i|$, by the definition of the satisfaction relation, $M, [i,j] \models (a : D)_k$. Note, if $k \neq |j - i|$ the branch could not be open (see the definition of an open branch in Definition 37 on Page 85).

- Let $\varphi = R(a,b)_k$. Based on the above definition of $\mathbb{S}[i,j]$, $\mathbb{S}[i,j] \models R(a,b)$. Since $\mathbb{S}[i,j] \models R(a,b)$ and $k = |j - i|$, by the definition of the satisfaction relation, $M, [i,j] \models R(a,b)_k$.

- Let $\varphi = (D = \top)_k$. Based on the $\mathcal{R}_{=}$ rule, if $((a : E)_{k''}, [i^{''}, j^{''}]) \in B$ or $((a : \neg E)_{k''}, [i^{''}, j^{''}]) \in B$, then $((a : D)_{k'''}, [i^{'''}, j^{'''}]) \in B$. Since for any individual and any concept E and any interval $[i^{''}, j^{''}]$ s.t. $((a : E)_{k''}, [i^{''}, j^{''}]) \in B$ or $((a : \neg E)_{k''}, [i^{''}, j^{''}]) \in B$, $\mathcal{R}_{=}$ is applied, then for all individuals appearing on $[i,j]$ we would have $(((a : D)_{k'''}, [i^{'''}, j^{'''}]) \in B)$. Therefore $M, [i,j] \models (D = \top)_k$.

(Induction case) Assume that $P(m)$ is true for all $0 \leq m \leq t$. We prove that $P(t+1)$ is true. Consider the following cases.

- Let $\varphi = \varphi_0 \wedge \varphi_1$. By the $\mathcal{R}_{\wedge}$ rule, $(\varphi_0, [i,j]) \in B$ and $(\varphi_1, [i,j]) \in B$. Since the number of temporal operators in $\varphi_0 (\varphi_1)$ is less than the number of temporal operators in φ, by the induction assumption, $M, [i,j] \models \varphi_0$ $(M, [i,j] \models \varphi_1)$; it follows that $M, [i,j] \models \varphi$.

- Let $\varphi = \varphi_0 \vee \varphi_1$. By the $\mathcal{R}_\vee$ rule, $(\varphi_0, [i,j]) \in B$ or $(\varphi_1, [i,j]) \in B$. Since the number of temporal operators in $\varphi_0(\varphi_1)$ is less than the number of temporal operators in φ, by the induction assumption, $M, [i,j] \models \varphi_0$ or $M, [i,j] \models \varphi_1$; it follows that $M, [i,j] \models \varphi$.

- Let $\varphi = \Diamond_r^z \varphi_0$ $(z \in \{*, +, -\})$. By the $\mathrm{R}_{\Diamond_r^z}$ rule, $(\varphi_0, [j,h]) \in B$ $(j < h)$. Since the number of temporal operators in φ_0 is less than the number of temporal operators in φ, by the induction assumption, $M, [j,h] \models \varphi_0$; it follows that $M, [i,j] \models \varphi$.

- Let $\varphi = \Diamond_l^z \varphi_0$ $(z \in \{*, +, -\})$. By the $\mathrm{R}_{\Diamond_l^z}$ rule, $(\varphi_0, [h,i]) \in B$ $(h < i)$. Since the number of temporal operators in φ_0 is less than the number of temporal operators in φ, by the induction assumption, $M, [h,i] \models \varphi_0$; it follows that $M, [i,j] \models \varphi$. □

5.3.8 Termination of the tableau algorithm for generic MITDL formulas

Theorem 14. (Termination). Let ψ be a formula; let $\mathcal{T}$ be a tableau for ψ. There cannot be an infinite sequence of rule applications in order to construct $\mathcal{T}$.

General sketch of proof. We pursue the same strategy of proof as the strategy which we had in Section 5.3.2 (on Page 86). Hence, we prove that the number of temporal nodes, as well as the number of non-temporal nodes residing on a branch are finite. Consider the following items.

- The number of temporal nodes is finite.

Since we have not changed the definition of temporal rules, based on Corollary 2 (on page 90), the number of temporal nodes is finite.

- The number of non-temporal nodes is finite.
  To demonstrate the finiteness of the number of non-temporal nodes, we prove the following facts.

  1. The number of distinct intervals in the nodes of a tableau is finite (Lemma 17 on Page 120).
  2. A full label of an individual contains a finite number of concept descriptions (Corollary 5).
  3. The number of individuals appearing in the nodes of a tableau is finite.
     We first define the notion of *non-blocked* individuals (Definition 46). We divide the individuals into two categories: old individuals and new individuals (see Page 87). Based on the definition of old individuals, the number of old individuals is finite. Then we prove that the number of non-blocked individuals is finite (Lemma 18). Consequently, the number of the application of $R_=$ is finite (Lemma 19). Eventually, we prove that the number of individuals is finite (Lemma 20 on Page 122).

  Having these facts, we show that the number of non-temporal nodes is finite (Theorem 16 on Page 124). Finally, we present the proof of termination.

Proof in detail. In order to prove the termination, we should prove the following theorems.

Lemma 17. The number of distinct intervals in the nodes of a tableau is finite.

Proof. In Lemma 10 (on Page 92), we proved that the number of intervals introduced either by applying the temporal rules or by applying $\mathcal{R}_{\sqcap}, \mathcal{R}_{\sqcup}, \mathcal{R}_{\exists}, \mathcal{R}_{\forall}$ rules is finite. Based on the definition of the $\mathcal{R}_{=}$ rule, applying this rule on a node (say $\mathbf{n}$) may introduce a new interval which is a subinterval of the interval in $\mathbf{n}$. Based on Remark 1 (on Page 77), the number of subintervals of an interval is finite. Having these facts, the number of distinct intervals in the nodes of a tableau is finite □

Theorem 15. Let B be a branch of a tableau for a formula; let D be a concept description. For a given individual a, the number of repetitive nodes on B that contain $a : D$ is at most twice the number of non-repetitive nodes on B that contain $a : D$.

Proof. Since an application of the $\mathcal{R}_{=}$ rule does not add any repetitive node to B, the proof of this theorem is similar to the proof of Theorem 8 (on Page 95). □

Corollary 5. For any individual a and any branch B, the full label of a w.r.t. B is finite.

Definition 46. An individual a is a *non-blocked* individual if there is an interval on which a is not blocked by another individual.

Lemma 18. The number of non-blocked individuals in the nodes of a tableau is finite.

Proof. Let $\mathcal{T}$ be a tableau. Let M (resp. N) be the number of distinct (sub) intervals (resp. concept descriptions) appearing in the nodes of $\mathcal{T}$. Recall

that every node that contain a concept assertion have an individual name, a concept description and an interval in its labelled formula. The number of distinct combinations of a concept description and an interval is equal to $M \times N$. In other words, given an individual a, the number of distinct labelled formulas which can contain a concept assertion about a (e.g., $((a : D)_k, [c_i, c_j])$ where D is a concept description, and $[c_i, c_j]$ is an interval) is $M \times N$. Note that, for an individual a, some of these labelled formulas usually appear in the tableau. Therefore, for a given individual a, there are $2^{M \times N}$ different sets of these labelled formulas that can appear in the tableau. Now, we show that the maximum number of non-blocked individuals is $2^{M \times N}$. Suppose that there is $2^{M \times N} + 1$ non-blocked individuals. Therefore, there must be two individuals (say a,b) s.t. the difference between the labelled formulas that contain a concept assertion about a and the labelled formulas that contain a concept assertion about b is only the individual names. Hence, based on the definition of temporal subset blocking, one of these individuals must be blocked by another individual. This means both of these individuals cannot be non-blocked individuals. Consequently, the number of non-blocked individual is bounded by $2^{M \times N}$. □

Lemma 19. The $\mathcal{R}_=$ rule cannot be applied infinitely often during the construction of a tableau for a formula.

Proof. Since the number non-blocked individuals (Lemma 18), the number of distinct concept descriptions (Lemma 12 on Page 94) and the number of distinct intervals (Lemma 17) are all finite, the number of non-temporal nodes that contain distinct concept assertions about non-blocked individuals is finite. By its definition, the $\mathcal{R}_=$ rule cannot be applied either on a node that contains a concept assertion about a blocked individual or on a repetitive node (see

Remark 7 on Page 95). Therefore, the number of the $\mathcal{R}_{=}$ rule application is bounded by the number of the aforementioned non-temporal nodes. □

Lemma 20. Let $\mathcal{T}$ be a tableau for a formula. The number of the individuals in the nodes of $\mathcal{T}$ is finite.

Proof. Let $i, s \in \mathbb{N}$; let r_i be a tableau rule, and let $r_1, r_2, r_3, \ldots$ be the sequence of rule applications for the construction of $\mathcal{T}$. Based on Lemma 9 (on Page 90) and Lemma 19, we can find a rule r_s s.t. $\forall_{i \geq s}\, r_i \in \{\mathcal{R}_{\sqcap}, \mathcal{R}_{\sqcup}, \mathcal{R}_{\exists}, \mathcal{R}_{\forall}\}$. Therefore, we divide the sequence of the rules into two subsequences, i.e., $r_1, \ldots, r_{s-1}$ and $r_s, r_{s+1}, \ldots$. Obviously, the first subsequence contains a finite number of elements. Since application of any rule adds a finite number of nodes to $\mathcal{T}$, after the application of r_{s-1}, $\mathcal{T}$ contains a finite number of temporal nodes (Corollary 2 on Page 90) and a finite number of non-temporal nodes. Some of these non-temporal nodes are not expandable because they are either repetitive nodes or they contain concept assertions about blocked individuals.

In the rest of proof, we divide the individuals appearing in the tableau into two categories: *first-generation* individuals and *non-first-generation* individuals. The individuals appearing in the tableau before the application the r_s rule (excluding the r_s rule) are called first-generation individuals. Some of these individuals are old individuals, and some of them are new individuals. Note that an old individual always is a first-generation individual. The individuals appearing in the tableau after the application the r_s rule (including the r_s rule) are called non-first-generation individuals. Since a non-first-generation individual is introduced during the construction of a tableau, it is considered a new individual.

Since right after applying the r_{s-1} rule, the number of existing non-temporal nodes is finite, the number of first-generation individuals in the existing nodes

is also finite. The number of old individuals is finite (Remark 6 on Page 94). We show that the number of non-first-generation individuals is finite.

Let a be a first-generation individual; let b be a non-first-generation individual; let R be a role, and let $k, k^{'} \in \mathbb{N}$. The number of non-temporal nodes that contain $R(a,b)_k$ is bounded by the number of existential restrictions in the full labels of a. Therefore, the number of non-first-generation individuals that are R-successors of the first-generation individuals is bounded by the number of existential restrictions in any full labels of the first-generation individuals.

Let c be a non-first-generation individual, and let S be a role. S (resp. k) can be equal to R (resp. $k^{'}$). For any S, the number of non-temporal nodes that contain $S(b,c)_{k'}$ is bounded by the number of existential restrictions in the full labels of b. Therefore, the number of non-first-generation individuals which are S-successors of the existing non-first-generation individuals (the individuals that are R-successors of the first-generation individuals) is bounded by the number of existential restrictions in any full labels of existing non-first-generation individuals. We can use similar reasoning to prove that the number of non-first-generation individuals that are role-successors of other non-first-generation individuals is also finite.

Now, it remains to show that the successor chains of non-first-generation individuals is finite. Since b is a new individual, it is only related to a by the relation R. In fact, b is not related to another individual or even to a by any relation other than R. Because of this fact and by the definition of DL rules, the length of maximal concept description in any of the full labels of b is strictly less than the length of maximal concept description in any of the full labels of a; hence, since the length of a concept description is greater than zero, a successor chain of non-first-generation individuals is finite. Therefore, the number non-first-generation individuals is finite.

□

Theorem 16. Let ψ be a formula. The number of non-temporal nodes in a tableau for ψ is finite.

Proof. Similar to the proof of Theorem 9 (on Page 97). □

Now we are able to prove the termination theorem.

Proof. (Theorem 14 on Page 118.) Similar to the proof of Theorem 7 (on Page 86). □

5.3.9 Complexity of the tableau algorithm for generic MITDL formulas

In Sections 5.3.6 and 5.3.7, we proved that the algorithm is sound and complete when the expansion rules are applied in an arbitrary order. In this section, we assume that we first exhaustively apply the temporal rules; then we apply the DL rules. Before we calculate the number of nodes in a tableau, we should prove some theorems.

Lemma 21. Let ψ be a formula. The number of distinct intervals in the nodes of a branch is $\mathcal{O}(|\psi|^2)$.

Proof. Applying $\mathcal{R}_{\sqcup}$, $\mathcal{R}_{\sqcap}$ do not introduce any new interval, but the application of the $\mathcal{R}_{\forall}$ rule or the $\mathcal{R}_{=}$ rule on a node may introduce a new interval. When applying the temporal rules is finished, the number of non-temporal nodes is $\mathcal{O}(|\psi|)$ (see Lemma 15 on Page 100). This indicates that the number of distinct intervals in these non-temporal nodes is $\mathcal{O}(|\psi|)$. Based on the

definition of the $\mathcal{R}_\forall$ (resp. $\mathcal{R}_=$) rule, the interval in a node added by the application of the $\mathcal{R}_\forall$ (resp. $\mathcal{R}_=$) rule is equal to an intersection of two existing intervals. This means that by the application of these rules at most $\mathcal{O}(|\psi|^2)$ new intervals is introduced . Therefore, the number of distinct intervals in the nodes is $\mathcal{O}(|\psi|^2)$. □

Theorem 17. Let ψ be a formula; let $\mathcal{T}$ be a tableau for ψ; let B be a branch of $\mathcal{T}$. The number of non-temporal nodes on B resulting from the application of DL rules is $\mathcal{O}(2^{|\psi|^3})$.

Proof. The number of the non-temporal nodes on B is the summation of the following numbers. Let D be a concept description; let a, b be two old individuals; let c, d be two new individuals, and let R be a role.

- $\mathcal{O}(2^{|\psi|^3})$ - The number of non-temporal nodes on B that contain a concept assertion.

  Based on Remark 5 (on Page 93), the order of concept descriptions appearing in ψ along with their sub-descriptions is $\mathcal{O}(|\psi|)$. Also, based on Lemma 21, the number of distinct intervals in the nodes of $\mathcal{T}$ is $\mathcal{O}(|\psi|^2)$. As we mentioned in the proof of Lemma 18 (on Page 120), the maximum number of non-blocked individuals is $2^{|\psi|^2 * |\psi|}$. Therefore, the number of concept assertions that can appear in the nodes is the product of these recent numbers, i.e., it is $\mathcal{O}(2^{|\psi|^3} * |\psi| * |\psi|^2)$. If due to the application of any rule, a new node, which contain a concept assertion, is added to the branch, this now is a repetitive node or it contains a concept assertion about a blocked node. In both cases, the new node is not expandable. On the other hand, based on the definition of $\mathrm{R}_=$ and $\mathrm{R}_\forall$, none of them can be applied on the existing non-temporal nodes. It is easy to see that the number of nodes added to the branch by applying $\mathrm{R}_\wedge$, $\mathrm{R}_\vee$, $\mathrm{R}_\exists$ is

twice the number of existing non-temporal nodes. Therefore, the total number of the nodes that contain a concept assertion is still $\mathcal{O}(2^{|\psi|^3})$.

- $\mathcal{O}(|\psi|)$ - The number of non-temporal nodes (on B) that contain $R(a, b)$. Since no DL rule adds a node that contain a role assertion about two old individuals, this kind of assertions should already exist in ψ (as subformulas of ψ) in order to appear in the tableau. Therefore, the number of this kind of non-temporal node is bounded by the number of non-temporal nodes which are added to a tableau by the application of temporal rules (see Lemma 15 on Page 100). In fact, these nodes are a subset of the non-temporal nodes which we considered in the previous case.

- $\mathcal{O}(2^{|\psi|^3})$ - The number of non-temporal nodes (on B) that contain $R(a, c)$ or $R(c, d)$.

  Only the $\mathcal{R}_\exists$ rule can add a node that contains a role assertion between an old individual and a new individual (resp. two new individuals). By the definition of this rule, every application of the $\mathcal{R}_\exists$ rule introduces a new individual and adds two nodes to a branch. One node contains a concept assertion about the new individual, and the other node contains a role assertion about that individual. Since the number of concept assertions in $\mathcal{T}$ is $\mathcal{O}(2^{|\psi|^3})$, the number of the $\mathcal{R}_\exists$ rule applications is $\mathcal{O}(2^{|\psi|^3})$. Therefore, the number of non-temporal nodes that contain $R(a, c)_{k'}$ or $R(c, d)_{k'}$ is $\mathcal{O}(2^{|\psi|^3})$.

Therefore, the number of the nodes on B is $\mathcal{O}(2^{|\psi|^3})$. □

Theorem 18. (Worst case complexity.) The complexity of the tableau algorithm for checking the satisfiability of a formula is 2ExpTime.

Proof. A tableau for a formula consists of two parts: the top part and the bottom part. The top part is created during the application of temporal rules while the bottom part is created by the application of DL rules. After applying the temporal rules, the algorithm expands the existing branches by applying DL rules. The length of the longest branch in the top part of the tableau is less than $|\psi|$ (see Theorem 5 on Page 74); so the maximum number of the nodes in the top part is $(|\psi|^{(|\psi|+1)})/(|\psi - 1|)$. Also, the number of the leaves of top part is $|\psi|^{|\psi|}$. In other words, there are $|\psi|^{|\psi|}$ branches that should be expanded by applying DL rules. Based on this fact and Theorem 17, the number of the nodes in the bottom part of a tableau is $\mathcal{O}(|\psi|^{|\psi|} * 2^{|\psi|^3})$. Hence, the total number of the nodes in a tableau is $\mathcal{O}(|\psi|^{|\psi|} * 2^{|\psi|^3} + (|\psi|^{(|\psi|+1)} + 1)/(|\psi| - 1))$. Since this number is $\mathcal{O}(2^{2^{|\psi|}})$, the complexity of the algorithm is 2ExpTime. □

5.4 Related Works

Because of the importance of undecidability and the concern with the high complexity of automated reasoning, a few interval-based temporal description logics have been designed. To the best of our knowledge, these logics are: $\mathcal{TL}$-$\mathcal{ALCF}$ [9, 10, 21, 108, 114, 115], $\mathcal{TL}$-$\mathcal{F}$ [9, 108], $\mathcal{TLU}$-$\mathcal{FU}$ [9, 108, 116], $\mathcal{TL}$-$\mathcal{SHOIN}(\mathcal{D})$ [115, 117], $\mathcal{TLF}_5^-$ [118, 119], Schmiedel's formalism [9, 120], $\mathcal{TALC}$ [9, 118, 119] and $\mathcal{T}$-$\mathcal{ALC}$ [121]. We consider each of them in detail.

5.4.1 Schmiedel's formalism

As a first attempt to combine an interval-based temporal logic (ITL) with a DL, in 1990, A. Schmiedel extended a terminological logic with two temporal operators (limited universal and existential quantification) over intervals

[118]. This formalism is able to encode all of Allen's relations. Moreover, it is possible to specify the length of an interval in it. There is no negation in this logic[120] and, no algorithm was designed for the subsumption reasoning in this formalism. This problem for the formalism extended by negation is Π^1_1-hard [9].

5.4.2 $\mathcal{TALC}$ and $\mathcal{TLF}_5^-$

In 1993, C. Bettini proposed a family of logics which identify intervals with their endpoints. Some of the members of this family are able to model all of Allen's relations. The major problem of this family of logics is undecidability of subsumption reasoning in $(\mathbb{R}, \leq)$,$(\mathbb{Z}, \leq)$,$(\mathbb{N}, \leq)$. Note, Bettini proposed another logic called $\mathcal{TLF}_5^-$ [118] which has polynomial time complexity. This logic does not allow disjunction and negation which is a significant restriction.

5.4.3 $\mathcal{TL}$-$\mathcal{ALCF}$

All of Allen's relations can be modeled in this logic. Unfortunately, it is not possible to bind the duration of an interval with a specific number. In other words, it is not a metric logic. Also, the temporal part of $\mathcal{TL}$-$\mathcal{ALCF}$ does not support *full negation*. While the complexity of satisfiability checking in the logic is PSPACE-complete [10], adding *full negation* to it makes the satisfiability problem undecidable.

5.4.4 $\mathcal{TL}$-$\mathcal{SHOIN}(\mathcal{D})$

This logic is achieved by extending the non-temporal part of $\mathcal{TL}$-$\mathcal{ALCF}$, by $\mathcal{SHOIN}(\mathcal{D})$ [122]. As with the previous logic, this logic is also not able

to bind a specific number for the duration of an interval. The complexity of satisfiability of a knowledge-base in this logic is NExpTime.

5.4.5 $\mathcal{TL}$-$\mathcal{F}$

This logic has no negation and disjunction; therefore, it is not expressive enough for modeling events in many domains, e.g., medicine.

5.4.6 $\mathcal{TLU}$-$\mathcal{FU}$

The logic $\mathcal{TLU}$-$\mathcal{FU}$ adds the disjunction operator to both the temporal and non-temporal parts of $\mathcal{TL}$-$\mathcal{F}$. Certainly, it is more expressive than $\mathcal{TL}$-$\mathcal{F}$, but it does not include the negation operator. The subsumption problem has NP-Complete complexity.

5.4.7 $\mathcal{T}$-$\mathcal{ALC}$

This temporal description logic was designed by Liu et al. in 2012. Indeed, a time interval is attached to every instance of the axioms in an ABox while the TBox has no temporal aspect; therefore, the main reasoning type in the logic is ABox consistency. We only have temporal instances rather than temporal concepts. Note, the complexity of instance checking in the logic is PSpace-complete.

5.5 Conclusion

In this chapter, we introduced a metric interval-based temporal description logic, called MITDL. Using the temporal part of MITDL, we were able to model the dynamic aspects of a process while the DL part of this logic encoded

the static aspects of the process. We identified a set of formulas of MITDL (simple formulas) and developed a tableau-based algorithm for checking the satisfiability of these formulas. We proved that the algorithm always terminates. Moreover, we showed that the complexity of the algorithm is PSpace. Then we extended and modified the existing algorithm to obtain an algorithm for checking the satisfiability of a generic MITDL formula. We proved the termination of the algorithm. We also proved that the algorithm is sound and complete. The soundness of our tableau algorithm states if a tableau of a formula is closed, the formula is not satisfiable while its completeness states that if there is an open branch in a tableau for a formula, the formula is satisfiable, and consequently, we are able to construct a class of models for the formula. Also, we showed that this algorithm has 2ExpTime complexity.

Chapter 6

Case Study: Modeling Clinical Practice Guidelines

In this chapter, we provide many case studies from the domain of medicine. We will show that *Clinical Practice Guidelines* (CPGs) can be modeled with IMPNL (resp. MITDL) and then checked to see if they are consistent. In other words, if there are any inconsistent conditions in a guideline modeled with IMPNL or MITDL, our algorithm determines that the guideline is not satisfiable.

We consider three CPGs, *Diagnosis and Treatment of HIV/AIDS*, *Treatment of Tuberculosis*, *Multi treatment of an HIV/AIDS-TB patient*. The first and the second guidelines are extracted from a well-known medical reference, [123] by an infection diseases specialist. He has also removed some details which were not important for us in this research. The last guideline is a combination of the other guidelines. We designed the third guideline in order to show that our algorithm can be used to detect inconsistencies when two or more guidelines are simultaneously followed for a patient with multiple diseases. We will first describe each guideline; then will model each of them with

IMPNL and MITDL. In order to have a self contained MITDL case studies, we have repeated some descriptions which we have already mentioned in IMPNL sections. Note that we assume the granularity of time is an *hour* (denoted by h), but for ease of understanding we use *day* (denoted by d), *month* (denoted by m) and *year* (denoted by yr) were applicable.

6.1 Diagnosis and Treatment of HIV/AIDS

In this guideline we need to model some periodic events; thus, we first present a technique for modelling periodic events before we explain the guideline. Both IMPNL and MITDL are able to model this kind of event if the duration of a period is constant and the number of the occurrences of periods is also known and constant, e.g., a patient should take an IBuprofen tablet every six hours for three days. In the domain of medicine, if the total number of the occurrences of the periods is unknown, we are able to assume that this periodicity would terminate within a reasonable maximum lifespan (e.g., 120 years). So we can model many periodic events in this domain. We define a non-primitive temporal operator $\circlearrowleft$ to model periodic events in this case study.

Definition 47. $\circlearrowleft^x \varphi = \underbrace{\varphi \wedge \Diamond_r(\varphi \wedge \Diamond_r(\varphi \wedge ...(\Diamond_r\varphi)...))}_{\varphi \text{ occurs } x \text{ times}}$

Generally, HIV disease has three major stages, as shown in Figure 6.1. The first stage, *Acute Infection* lasts 6-8 weeks. In this stage, a patient may have different symptoms, e.g., Fatigue, Headache [124]. After this stage, the patient goes to the next stage, *Clinical Latency*, and may have no symptoms at all. This stage may last 8-10 years. The last stage is called *AIDS*, and the patient should use different medications. Generally, an AIDS patient may live at most 20 years after this stage has started while the patient is under treatment. The

diagnostic process of the disease, as well as its treatment process, are the same for any HIV/AIDS patient in any of the stages. Figure 6.2 shows the guideline for HIV/AIDS diagnosis and treatment.

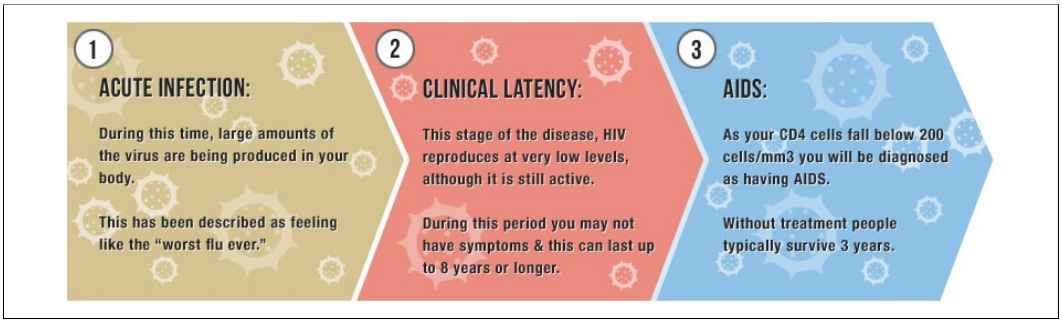

Figure 6.1: HIV Stages [125]

As can be seen in Figure 6.2, three blood-works should be performed to make sure that the patient has an HIV infection. Results for each blood-work take approximately 1 day. The registration may take 4 days. Routinely, the CD4 blood level of an infected patient should be investigated every 3 months. If the level is fine, and the patient has no sign of AIDS, i.e., no, so called *AIDS-Defining Conditions*, (e.g., Candidiasis of bronchi, trachea, or lungs, Cryptococcosis, Extrapulmonary Tuberculosis [126]), no medication is needed, but if the level of CD4 is not acceptable, or there is at least one AIDS-Defining Condition, the patient should take 3 medications every day. Also, the patient should see a medical doctor every 30 days. During these visits, the doctor ensures that the patient takes the right medicines with the right dosages and renews the prescriptions for the patient. There are different possible combinations of medicines for the treatment. We consider one combination here, i.e., *Kaletra, Tenofovir, Lamivadin*.

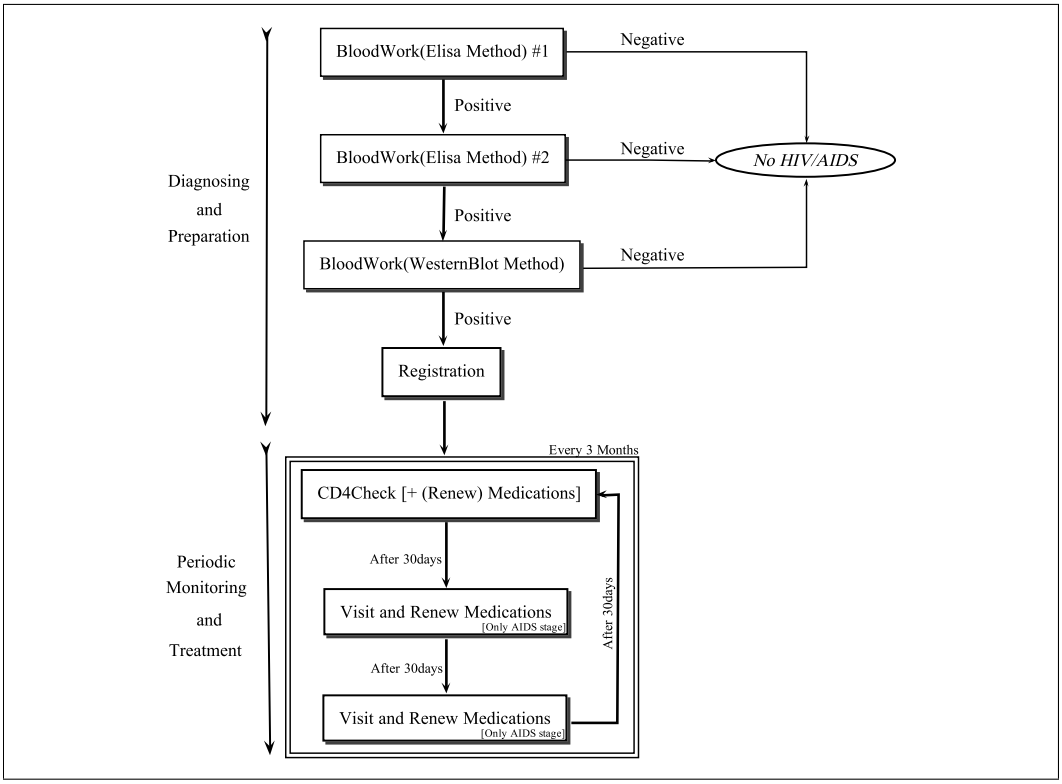

Figure 6.2: HIV/AIDS diagnosis and treatment

6.1.1 Modeling HIV/AIDS guideline with IMPNL

We have selected meaningful names for the propositions in the formula ψ_{HIV} (Formula 6.1) which describes the guideline. Formula ψ_0 describes the *diagnosing* and the *preparation* phase of the guideline. The propositions $Elisa^{\#1}$ and $Elisa^{\#2}$ respectively denote the first and second occurrences of blood-work (Elisa method).

$$\psi_{HIV} = \underbrace{Elisa^{\#1}_{1d} \wedge \Diamond_r(Elisa^{\#2}_{1d} \wedge \Diamond_r(WesternBlot_{1d} \wedge \Diamond_r(Registration_{4d}}_{\psi_0}$$
$$\wedge \Diamond_r \psi_1))) \tag{6.1}$$

Formalizing the *monitoring* and the *treatment* phase is more complicated. As we already mentioned, an HIV infected patient should take the medications if (s)he is in the AIDS stage; otherwise, no medication is needed. Formula 6.2 defines a periodic event which occurs 80 times at most because the duration of a period is 3 months, and the patient may live for at most 20 years after the AIDS stage has started.

$$\psi_1 = \circlearrowleft^{80} (\top_{90d} \wedge \Diamond_l \Diamond_r CD4Check_{1d} \wedge \Diamond_l \Diamond_r \psi_2) \tag{6.2}$$

The reason why we have used ψ_2 in the formula is because in the last stage of the disease, a patient should see the doctor every month as well as taking the medication every day. Also, (s)he should have a CD4Check every 3 months. $\top_{90d}$ in this formula specifies the length of a period. Note that $\top_{90d}$ can be removed from ψ_1 because the length of interval required for satisfying ψ_2 is 90 days which is equal to the desired period (i.e., 3 months), but it is not always the case when we model a periodic formula; so it would be always better to specify the length of a period using $\top_x$ where x is the length of the period. As can be seen in the following formula, if HasAIDS is false, no visit occurs and no medications are taken.

$$\begin{aligned}\psi_2 = \circlearrowleft^3 (\top_{30d} \\ \wedge \Diamond_l \Diamond_r (\text{HasAIDS}_{1h} \rightarrow (\text{Visit}_{1h} \\ \wedge \Diamond_l \Diamond_r (\text{TakeKaletra}_{30d} \wedge \text{TakeTenofovir}_{30d} \\ \wedge \text{TakeLamivadin}_{30d}))))\end{aligned} \tag{6.3}$$

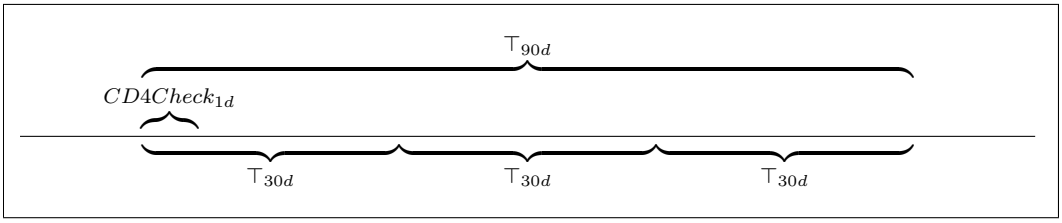

Figure 6.3: One period of ψ_1 when the patient is not in the last stage

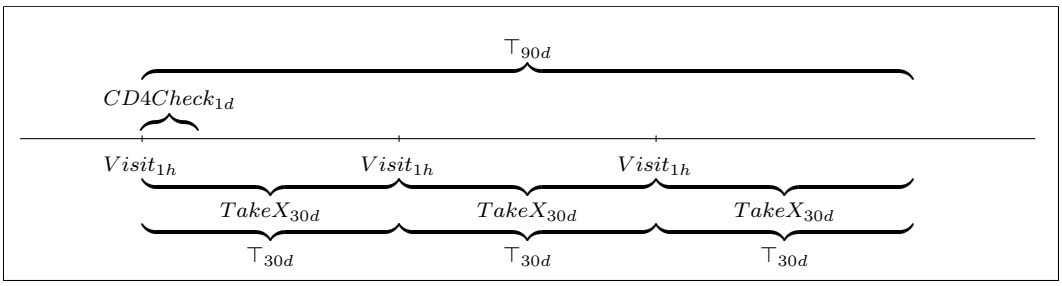

Figure 6.4: One period of ψ_1 when the patient is in the last stage

Figure 6.3 exhibits schematic presentations of ψ_1 and ψ_2 when the patient is in not in the last stage. In Figure 6.4, $TakeX_{30d} \in \{\text{TakeKaletra}_{30d}, \text{TakeTenofovir}_{30d}, \text{TakeLamivadin}_{30d}\}$. This figure shows the situation when the patient is in the last stage.

Another important point is to take into account drug contraindications when we model the guideline: two incompatible medicines should not be administrated to a patient simultaneously. We need to change the formula to reflect that this should not happen. For example, we know that Kaletra has a contraindication with some medicines, e.g., Alfuzosin, Cisapride, Rifampin, Pimozide [127]. We change Formula 6.3 to the following formula to model this fact. We leave it to the reader to update formula $\psi_2^{'}$ with respect to other drug contraindications.

$$\begin{aligned}\psi_2' = \circlearrowleft^3 (\top_{30d} \\ \wedge \Diamond_l \Diamond_r (\text{HasAIDS}_{1h} \rightarrow (\text{Visit}_{1h} \\ \wedge \Diamond_l \Diamond_r (\text{TakeKaletra}_{30d} \wedge \neg \text{TakeAlfuzosin}_{30d} \\ \wedge \neg \text{TakeCisapride}_{30d} \wedge \neg \text{TakeRifampin}_{30d} \\ \wedge \neg \text{TakePimozide}_{30d} \wedge \text{TakeTenofovir}_{30d} \\ \wedge \text{TakeLamivadin}_{30d})))) \end{aligned} \tag{6.4}$$

6.1.2 Checking the quality of HIV/AIDS Guideline

In order to use the tableau algorithm, we recall that in our language $a \rightarrow b$ denotes $\neg a \vee b$, and replace the periodic events with their expanded form. Due to lack of space, we assume that the periodic event occurs once during the diagnosis and the treatment (Formula 6.5). We use the function defined in section 4.1.3.1 to determine the annotated version of $\vartheta_{HIV} = \Diamond_r \psi_{HIV}$ (Formula 6.6). Also, we have added superscripts for $\Diamond_z^{z'}$ where $z \in \{r, l\}, z' \in \{+, -\}$ required for the selection strategy.

$$
\begin{aligned}
\psi_{HIV} = {} & \text{Elisa}^{\#1}_{1d} \wedge \\
& \Diamond_r(\text{Elisa}^{\#2}_{1d} \wedge \\
& \Diamond_r(\text{WesternBlot}_{1d} \wedge \\
& \Diamond_r(\text{Registration}_{4d} \wedge \\
& \Diamond_r(\top_{90d} \wedge \\
& \Diamond_l\Diamond_r(\text{CD4Check}_{1d} \wedge \\
& \Diamond_l\Diamond_r(\top_{30d} \wedge \\
& \Diamond_l\Diamond_r(\neg\text{HasAIDS}_{1h} \vee \\
& (\text{Visit}_{1h} \wedge \\
& \Diamond_l\Diamond_r(\text{TakeKaletra}_{30d} \wedge \neg\text{TakeAlfuzosin}_{30d} \\
& \wedge \neg\text{TakeCisapride}_{30d} \wedge \neg\text{TakeRifampin}_{30d} \wedge \neg\text{TakePimozide}_{30d} \\
& \wedge \text{TakeTenofovir}_{30d} \wedge \text{TakeLamivadin}_{30d}))))))))) \qquad (6.5)
\end{aligned}
$$

$$
\begin{aligned}
\vartheta_{HIV} = \Diamond_r^*(&\text{Elisa}_{1d}^{\#1} \wedge \\
&\Diamond_r^*(\text{Elisa}_{1d}^{\#2} \wedge \\
&\Diamond_r^*(\text{WesternBlot}_{1d} \wedge \\
&\Diamond_r^*(\text{Registration}_{4d} \wedge \\
&\Diamond_r^*(\top_{90d} \wedge \\
&{}^{1}\Diamond_l^-\Diamond_r^*(\text{CD4Check}_{1d} \wedge \\
&{}^{2}\Diamond_l^-\Diamond_r^*(\top_{30d} \wedge \\
&{}^{4}\Diamond_l^-\Diamond_r^*(\neg\text{HasAIDS}_{1h} \\
&\vee (\text{Visit}_{1h} \wedge \\
&{}^{8}\Diamond_l^-\Diamond_r^*(\text{TakeKaletra}_{30d} \wedge \\
&\neg\text{TakeAlfuzosin}_{30d} \wedge \neg\text{TakeCisapride}_{30d} \wedge \\
&\neg\text{TakeRifampin}_{30d} \wedge \neg\text{TakePimozide}_{30d} \wedge \\
&\text{TakeTenofovir}_{30d} \wedge \text{TakeLamivadin}_{30d}))))))))))
\end{aligned}
\tag{6.6}
$$

Before we build a tableau for the formula, we select a constant $\kappa = 338d2h (= LN(\vartheta_{HIV}))$. The construction of a tableau for ϑ_{HIV} is straightforward. Figures C.1, C.2 and C.3 show the tableau for the formula. As can be seen in Figure C.3, there is an open branch in the tableau, which means the formula is consistent. There is a subtle point here: the first branch determines the situation in which the patient is not in the AIDS stage, but we know that in order to have a completely consistent guideline, we should check whether the other branch remains open or not. If the branch remains open, it means we have a consistent guideline even if the patient is in the AIDS stage. We need to have both branches open because we have modelled the whole guideline

with one formula but we know a patient who comes for the diagnosis and the treatment of HIV, can be in any the stages of HIV (or does not have HIV). We could define three formulas for these three stages but in that case two of the formulas are the same since the process of the diagnosis and the treatment of HIV for a patient in the first or second stages of HIV is the same.

6.1.3 A concrete model

In this section, we construct a concrete model for the previous case study based on the nodes residing on an open branch shown in Figures C.1, C.2 and C.3. We should first select one open branch. Here, we construct a model for nodes 1 to 27. Now, it is sufficient to assign suitable values to the c_is. Let c_0 be "Feb 22, 2014 at 14:30". Based on the information existing at the nodes, we have:

- c_1 ="Feb 22, 2014 at 16:30" - Node 1
- c_2 ="Feb 23, 2014 at 16:30" - Node 2
- c_3 ="Feb 24, 2014 at 16:30" - Node 5
- c_4 ="Feb 25, 2014 at 16:30" - Node 8
- c_5 ="Mar 1, 2014 at 16:30" - Node 11
- c_6 ="May 29, 2014 at 16:30" - Node 14
- c_7 ="Mar 28, 2013 at 14:30" - Node 17
- c_8 ="Mar 2, 2014 at 16:30" - Node 18
- c_9 ="Apr 24, 2012 at 12:30" - Node 21

- c_{10} ="Mar 31, 2014 at 16:30" - Node 22
- c_{11} ="Jun 18, 2010 at 8:30" - Node 25
- c_{12} ="Mar 1, 2014 at 17:30" - Node 26

We are able to arbitrarily assign any value to other c_is which do not appear in this branch. To give better intuition, we have exhibited the information in Figure 6.5. In this scenario, the patient does not have AIDS yet, and (s)he is still in the Acute Infection or in the Clinical Latency stage. It is not so difficult to construct models for the other open branch in Figures C.1, C.2 and C.3. We leave this to the reader.

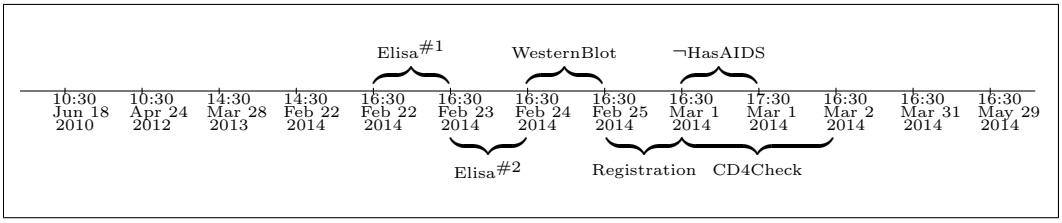

Figure 6.5: Concrete Model for HIV case study

6.2 Modeling Diagnosis and Treatment of HIV/AIDS with MITDL

In order to have an understandable formula, we have selected meaningful names for the concepts and roles used in the formula φ_{HIV} (Formula 6.7) which describes the guideline. The following formula (with the exception of the $\Diamond_r\varphi_1$ subformula) describes the *diagnosing* and the *preparation* phase of the guideline (see Figure 6.6).

$$\varphi_{HIV} = (\text{p:Patient})_{120yr} \wedge$$
$$\Diamond_l \Diamond_r ((\text{e1:Elisa})_{1d} \wedge \text{PerformTest(p, e1)}_{1d} \wedge$$
$$\Diamond_r ((\text{e2:Elisa})_{1d} \wedge \text{PerformTest(p, e2)}_{1d} \wedge$$
$$\Diamond_r ((\text{w:WesternBlot})_{1d} \wedge \text{PerformTest(p, w)}_{1d} \wedge$$
$$\Diamond_r ((\text{r:Registration})_{4d} \wedge \text{AdministrativeAction(p, r)}_{4d} \wedge \Diamond_r \varphi_1)))) \quad (6.7)$$

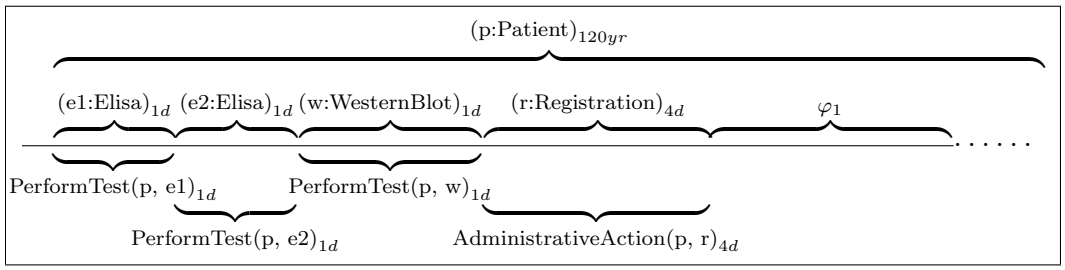

Figure 6.6: Schematic illustration of φ_{HIV}

As we already mentioned, MITDL is more expressive than IMPNL. This fact is obvious when ψ_{HIV} (Formula 6.1) and φ_{HIV} are compared. In φ_{HIV}, we have defined some binary relations (e.g., $PerformTest$) which we were not able to do in IMPNL. Also, in ψ_{HIV} we have implicitly assumed that the propositions (e.g., $WesternBlot$) are about a particular patient while in φ_{HIV} we have explicitly expressed that using $(\text{p:Patient})_{120yr}$ axiom. The same issue is true for the rest of the MITDL formulas in this chapter.

Formalizing the *monitoring* and the *treatment* phase is more complicated. As we already mentioned, an HIV infected patient should take the medications if (s)he is in the AIDS stage; otherwise, no medication is needed. Formula 6.8 defines a periodic event which occurs 80 times at most because the patient may live for at most 20 years after the AIDS stage has started, and the duration

of a period is 3 months.

$$\varphi_1 = \circlearrowleft^{80} (\top_{90d} \wedge \Diamond_l \Diamond_r ((\text{c:CD4Check})_{1d} \wedge \text{PerformTest(p,c)}_{1d}) \wedge \Diamond_l \Diamond_r \varphi_2) \tag{6.8}$$

The reason why we have used φ_2 in the formula is because in the last stage of the disease, a patient should see the doctor every month as well as should take the medication every day. Also, (s)he should have a CD4Check every 3 months. As can be seen in the following formula, if patient has no AIDS (equivalently, p is an individual of ¬HasAIDS concept), no visit or taking medications occurs.

$$\begin{aligned}
\varphi_2 = \circlearrowleft^{3} (&\top_{30d} \wedge \\
&\Diamond_l \Diamond_r ((\text{p:}\neg\text{HasAIDS})_{1h} \vee \\
&((\text{p:HasAIDS})_{1h} \wedge (\text{p:}\exists\text{Visitedby.Doctor})_{1h} \wedge \\
&\Diamond_l \Diamond_r ((\text{k:Kaletra})_{30d} \wedge (\text{t:Tenofovir})_{30d} \wedge (\text{l:Lamivadin})_{30d} \wedge \\
&\text{TakeMedicine(p,k)}_{30d} \wedge \text{TakeMedicine(p,t)}_{30d} \wedge \\
&\text{TakeMedicine(p,l)}_{30d})))
\end{aligned} \tag{6.9}$$

6.2.1 Domain information

When we model the guideline, we have to take into account the possible drug contraindications , i.e., two incompatible medicines should not be administered to a patient simultaneously, in the sense that a patient is not allowed to take the first medicine during the period in which (s)he takes the second one. We should consider this kind of fact when we model medical guidelines.

We can model the facts, separately, and consider them when we check the satisfiability of a formula. For example, we know that Kaletra, Norvir and Aptivus [128] are *Protease inhibitors* (Fact 1), and *Protease inhibitors* have a drug contraindication with some medicines (Fact 2), e.g., Rifampin [127]. In $\mathcal{ALC}$, these facts can be modeled using the following axioms. Assume that *TakeMedicine* is an $\mathcal{ALC}$ role which keeps track of the medicines which a patient may take during the treatment.

- (Fact 1) Kaletra $\sqcup$ Norvir $\sqcup$ Aptivus $\sqsubset$ ProteaseInhibitors

  This axiom indicates that Kaletra, Norvir and Aptivus [128] are *Protease inhibitors*. Equivalently, we can model this with ($\neg$Kaletra $\sqcap\neg$Norvir $\sqcap\neg$Aptivus) $\sqcup$ ProteaseInhibitors = $\top$.

- (Fact 2) $\exists$TakeMedicine.Rifampin $\sqcap\exists$TakeMedicine.ProteaseInhibitors = $\bot$

  This axiom states that there is no patient who takes Rifampin and a Protease inhibitors medicine simultaneously. The axiom can be equivalently modeled with $\forall$TakeMedicine.$\neg$Rifampin $\sqcup\forall$TakeMedicine.$\neg$ ProteaseInhibitors = $\top$.

We want to model these axioms in MITDL. These axioms are true during the lifetime of a patient (assuming that the lifespan of a patient is 120 years at most); so we model them as follows.

$$\varphi_{Fact_1} = ((\neg\text{Kaletra} \sqcap \neg\text{Norvir} \sqcap \neg\text{Aptivus}) \sqcup \text{ProteaseInhibitors} = \top)_{120yr}$$

$$\varphi_{Fact_2} = (\forall\text{TakeMedicine}.\neg\text{Rifampin} \sqcup \forall\text{TakeMedicine}.\neg\text{ProteaseInhibitors} = \top)_{120yr}$$

6.2.2 Checking the quality of the HIV/AIDS Guideline

In order to use the tableau algorithm, we replace the periodic events with their expanded form. Due to the lack of space, we assume that the periodic event occurs once during the diagnoses and the treatment (Formula 6.10). We conjoin Fact 1 and Fact 2 to φ_{HIV}, and use the function defined in Section 4.1.3.1 to determine the annotated version of $\xi_{HIV} = \Diamond_r(\varphi_{HIV} \wedge \varphi_{Fact_1} \wedge \varphi_{Fact_2})$ (Formula 6.11). Also, we have added superscripts for $\Diamond_z^{z'}$ where $z \in \{r, l\}, z' \in \{+, -\}$ required for the selection strategy.

$$
\begin{aligned}
\varphi_{HIV} = \ & (\text{p:Patient})_{120yr} \wedge \\
& \Diamond_l \Diamond_r ((\text{e1:Elisa})_{1d} \wedge \text{PerformTest(p,e1)}_{1d} \wedge \\
& \Diamond_r ((\text{e2:Elisa})_{1d} \wedge \text{PerformTest(p,e2)}_{1d} \wedge \\
& \Diamond_r ((\text{w:WesternBlot})_{1d} \wedge \text{PerformTest(p,w)}_{1d} \wedge \\
& \Diamond_r ((\text{r:Registration})_{4d} \wedge \text{AdministrativeAction(p,r)}_{4d} \wedge \\
& \Diamond_r (\top_{90d} \wedge \\
& \Diamond_l \Diamond_r ((\text{c:CD4Check})_{1d} \wedge \text{PerformTest(p,c)}_{1d} \wedge \\
& \Diamond_l \Diamond_r (\top_{30d} \wedge \\
& \Diamond_l \Diamond_r ((\text{p:}\neg\text{HasAIDS})_{1h} \vee \\
& ((\text{p:HasAIDS})_{1h} \wedge (\text{p:}\exists\text{Visitedby.Doctor})_{1h} \wedge \\
& \Diamond_l \Diamond_r ((\text{k:Kaletra})_{30d} \wedge (\text{t:Tenofovir})_{30d} \wedge (\text{l:Lamivadin})_{30d} \\
& \wedge \text{TakeMedicine(p,k)}_{30d} \wedge \text{TakeMedicine(p,t)}_{30d} \\
& \wedge \text{TakeMedicine(p,l)}_{30d}))))))))))
\end{aligned}
\tag{6.10}
$$

$$\xi_{HIV} = \Diamond_r^*(((\neg\text{Kaletra} \sqcap \neg\text{Norvir} \sqcap \neg\text{Aptivus}) \sqcup \text{ProteaseInhibitors} = \top)_{120yr} \wedge$$
$$(\forall\text{TakeMedicine}.\neg\text{Rifampin} \sqcup \forall\text{TakeMedicine}.\neg\text{ProteaseInhibitors} = \top)_{120yr}$$
$$\wedge\ (\text{p:Patient})_{120yr} \wedge$$
$$^{1}\Diamond_l^-\Diamond_r^*((\text{e1:Elisa})_{1d} \wedge \text{PerformTest(p, e1)}_{1d} \wedge$$
$$\Diamond_r^*((\text{e2:Elisa})_{1d} \wedge \text{PerformTest(p,e2)}_{1d} \wedge$$
$$\Diamond_r^*((\text{w:WesternBlot})_{1d} \wedge \text{PerformTest(p,w)}_{1d} \wedge$$
$$\Diamond_r^*((\text{r:Registration})_{4d} \wedge \text{AdministrativeAction(p,r)}_{4d} \wedge$$
$$\Diamond_r^*(\top_{90d} \wedge$$
$$^{2}\Diamond_l^-\Diamond_r^*((\text{c:CD4Check})_{1d} \wedge \text{PerformTest(p,c)}_{1d} \wedge$$
$$^{4}\Diamond_l^-\Diamond_r^*(\top_{30d} \wedge$$
$$^{8}\Diamond_l^-\Diamond_r^*((\text{p:}\neg\text{HasAIDS})_{1h} \vee$$
$$((\text{p:HasAIDS})_{1h} \wedge (\text{p:}\exists\text{Visitedby.Doctor})_{1h} \wedge$$
$$^{16}\Diamond_l^-\Diamond_r^*((\text{k:Kaletra})_{30d} \wedge (\text{t:Tenofovir})_{30d} \wedge (\text{l:Lamivadin})_{30d} \wedge$$
$$\text{TakeMedicine(p,k)}_{30d} \wedge \text{TakeMedicine(p,t)}_{30d} \wedge$$
$$\text{TakeMedicine(p,l)}_{30d})))))))))) \tag{6.11}$$

Let $\kappa = 360yr316d3h(= LN(\xi_{HIV}))$. The construction of a tableau for ξ_{HIV} is straightforward. Figures C.4, C.5, C.6, C.7 and C.8 show the tableau for the formula. As can be seen in Figure C.8, there is an open branch in the tableau which means the formula is consistent. We observe that the same subtle point exists here as with the IMPNL tableau for the guideline (see Page 139).

6.2.3 A concrete model

In this section, we construct a concrete model for the previous case study based on the nodes residing on an open branch shown in Figures C.4, C.5 , C.6, C.6 and C.8. We should first select one open branch. Here, we construct a model for the branch ending with the node 68. In this scenario, the patient does not have AIDS yet and (s)he is still in the Acute Infection or in the Clinical Latency stage. First we need to assign suitable values to the c_is. Let c_0 be "Feb 22, 2014 at 14:30". Based on the information existing at the nodes, we have the following values for the c_is. Note that we are able to arbitrarily assign any value to other c_is which do not appear in this branch.

- c_1 ="Feb 22, 2014 at 16:30" – see Node 1
- c_2 ="Feb 22, 2134 at 16:30" – see Node 2
- c_3 ="Feb 23, 2014 at 16:30" – see Node 10
- c_4 ="Feb 24, 2014 at 16:30" – see Node 15
- c_5 ="Feb 25, 2014 at 16:30" – see Node 20
- c_6 ="Mar 1, 2014 at 16:30" – see Node 25
- c_7 ="May 30, 2014 at 16:30" – see Node 30
- c_8 ="Mar 2, 2014 at 16:30" – see Node 34
- c_9 ="Mar 31, 2014 at 16:30" – see Node 39
- c_{10} ="Mar 1, 2014 at 17:30" – see Node 43
- $c_{-1} = (c_1 - \kappa)$ – see Node 9

- $c_{-2} = (c_6 - 2\kappa)$ – see Node 33
- $c_{-3} = (c_6 - 4\kappa)$ – see Node 9
- $c_{-4} = (c_6 - 8\kappa)$ – see Node 42

The interpretations can be as follows:

- $\mathbb{S}^{[c_1,c_3]}$:
  - $\Delta^{[c_1,c_3]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$
  - Patient$^{[c_1,c_3]} = \{John\}$
  - Elisa$^{[c_1,c_3]} = \{John_Elisa_Test1\}$
  - PerformTest$^{[c_1,c_3]} = \{(John, John_Elisa_Test1)\}$
  - ProteaseInhibitors$^{[c_1,c_3]} = \{John, John_Elisa_Test1, John_Elisa_Test2, John_WesternBlot, Registration_of_John, John_CD4Check\}$
- $\mathbb{S}^{[c_3,c_4]}$:
  - $\Delta^{[c_3,c_4]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$
  - Patient$^{[c_3,c_4]} = \{John\}$
  - Elisa$^{[c_3,c_4]} = \{John_Elisa_Test2\}$
  - PerformTest$^{[c_3,c_4]} = \{(John, John_Elisa_Test2)\}$
  - ProteaseInhibitors$^{[c_3,c_4]} = \{John, John_Elisa_Test1, John_Elisa_Test2, John_WesternBlot, Registration_of_John, John_CD4Check\}$
- $\mathbb{S}^{[c_4,c_5]}$:

- $\Delta^{[c_4,c_5]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$
- $\text{Patient}^{[c_4,c_5]} = \{John\}$
- $\text{Elisa}^{[c_4,c_5]} = \{John\_WesternBlot\}$
- $\text{PerformTest}^{[c_4,c_5]} = \{(John, John\_WesternBlot)\}$
- $\text{ProteaseInhibitors}^{[c_4,c_5]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$

- $\mathbb{S}^{[c_5,c_6]}$:
  - $\Delta^{[c_5,c_6]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$
  - $\text{Patient}^{[c_5,c_6]} = \{John\}$
  - $\text{Elisa}^{[c_5,c_6]} = \{Registration\_of\_John\}$
  - $\text{AdministrativeAction}^{[c_5,c_6]} = \{(John, Registration\_of\_John)\}$
  - $\text{ProteaseInhibitors}^{[c_5,c_6]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$

- $\mathbb{S}^{[c_6,c_{10}]}$:
  - $\Delta^{[c_6,c_{10}]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$
  - $\text{Patient}^{[c_6,c_{10}]} = \{John\}$
  - $\text{HasAIDS}^{[c_6,c_{10}]} = \{\}$
  - $\text{ProteaseInhibitors}^{[c_6,c_{10}]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$

- $\mathbb{S}^{[c_6,c_8]}$:

- $\Delta^{[c_6,c_8]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$
- $\text{Patient}^{[c_6,c_8]} = \{John\}$
- $\text{CD4Check}^{[c_6,c_8]} = \{John\_CD4Check\}$
- $\text{PerformTest}^{[c_6,c_8]} = \{(John, John\_CD4Check)\}$
- $\text{ProteaseInhibitors}^{[c_6,c_8]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$

- $\mathbb{S}^{[c_8,c_2]}$:
  - $\Delta^{[c_8,c_2]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$
  - $\text{Patient}^{[c_8,c_2]} = \{John\}$
  - $\text{ProteaseInhibitors}^{[c_8,c_2]} = \{John, John\_Elisa\_Test1, John\_Elisa\_Test2, John\_WesternBlot, Registration\_of\_John, John\_CD4Check\}$

As seen in the above interpretations, the interpretations of ProteaseInhibitors are wrong in the sense that none of the elements in the interpretation is a medicine. This problem arises from the incomplete description of the domain information. For instance, we should add the $\neg\text{Patient} \sqcup \neg\text{ProteaseInhibitors} = \top)_{120yr}$ axiom to the guideline in order to prevent $John$ is from being considered a ProteaseInhibitor. But for the ease of constructing a tableau for the HIV/AIDS guideline, we did not add the fact to the guideline.

6.3 Treatment of Tuberculosis

In this case study we use IMPNL (resp. MITDL) to formalize a real-life guideline which describes the process of treatment of Tuberculosis (TB) with

a positive sputum smear test, based on [129]. Figure 6.7 exhibits the guideline which was simplified by an infection disease specialist.

Generally, the TB treatment has two major phases, the *Initial phase* and the *Continuation phase* [130]. The initial phase lasts two months during which the patient should be administrated four medicines, i.e., *Rifampin, Isoniazid, Pyrazinamide, Ethambutol* plus *VitaminB6*. The patient also needs to take *Rifampin, Isoniazid* and *VitaminB6* during the continuation phase which lasts 4 months. Three sputum smear tests are performed at the end of the 2nd, 4th and last month of the treatment. Each of these tests typically takes 3 days. If the result of one of these tests is positive, the patient is directed to TB services, otherwise the treatment process is completed and the disease is successfully healed. Note, the patient still takes his/her medicines during the SputumSmearTests with the exception of the last test.

Regardless of the modeling language, this guideline is satisfiable. Since checking the satisfiability of this guideline with the proposed tableau-based algorithms has no new features, we leave to the reader to use the algorithm and check its satisfiability.

6.3.1 Modeling Treatment of TB Guideline with IMPNL

Formula ψ_{TB} (Formula 6.12) models the guideline. Similar to the previous case study, we have selected meaningful names for the propositions; so it is easy for the reader to understand the formula. Figure 6.8 exhibits a typical scenario for this guideline. Note that we should add the drug contraindications to ψ_{TB} to make it complete, then check the formula using the tableau algorithm. Since modeling the guideline using MITDL is more complex than modeling it using IMPNL, we leave the process of modeling the guideline using MITDL in detail to the next section.

$$
\begin{aligned}
\psi_{TB} = {} & \text{TakeRifampin}_{60d} \wedge \text{TakeIsoniazid}_{60d} \wedge \text{TakePyrazinamide}_{60d} \\
& \wedge \text{TakeEthambutol}_{60d} \wedge \text{TakeVitaminB6}_{60d} \wedge \\
& \Diamond_r \Diamond_l (\neg \text{SputumSmearTest}^{\#1}_{3d} \rightarrow \\
& \Diamond_r (\text{TakeRifampin}_{60d} \wedge \text{TakeIsoniazid}_{60d} \wedge \text{TakeVitaminB6}_{60d} \wedge \\
& \Diamond_r \Diamond_l (\neg \text{SputumSmearTest}^{\#2}_{3d} \rightarrow \\
& \Diamond_r (\text{TakeRifampin}_{60d} \wedge \text{TakeIsoniazid}_{60d} \wedge \text{TakeVitaminB6}_{60d} \\
& \wedge \Diamond_r (\text{SputumSmearTest}^{\#3}_{3d} \rightarrow \Diamond_r \text{TBServicesDirection}_{1h})) \\
& \wedge (\text{SputumSmearTest}^{\#2}_{3d} \rightarrow \Diamond_r \text{TBServicesDirection}_{1h}))) \\
& \wedge (\text{SputumSmearTest}^{\#1}_{3d} \rightarrow \Diamond_r \text{TBServicesDirection}_{1h}))
\end{aligned}
\tag{6.12}
$$

6.3.2 Modeling Treatment of TB Guideline with MITDL

In this case study, we use MITDL to model the TB guideline. We first need to review the guideline (see Section 6.3) and recognize possible candidates for DL concepts or DL roles. Based on the given description of the TB guideline, possible candidates for DL concepts are *Patient, Initial Phase, Continuation Phase, Rifampin, Isoniazid, Pyrazinamide, Ethambutol, VitaminB6, Positive, Negative, TB Service Direction, Sputum Smear Test*, and candidates for DL roles are *Take Medicine, Refer, Has Sputum Smear Test, Has Test Result, Has Test*. As we go further, we may find that some of these DL concepts, or DL roles are not needed. For example, we can model the concept *Negative* with $\neg$*Positive*. It is not necessary to select a time interval or a time unit (e.g., *hour, month, day*) as a DL concept, because it is considered as a length for an interval, e.g., $(a : C)_{30d}$.

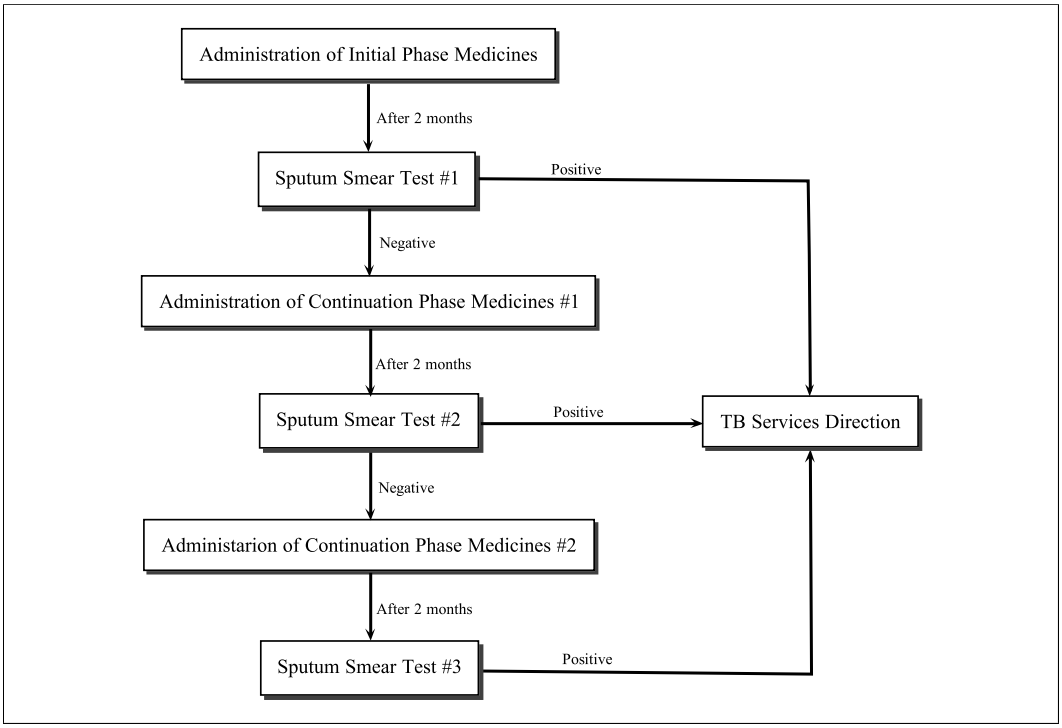

Figure 6.7: Treatment of a typical TB patient with positive sputum smear [129]

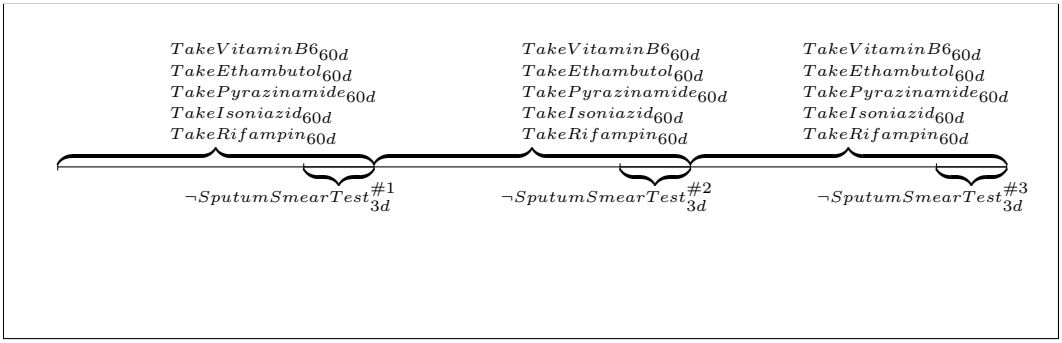

Figure 6.8: TB Treatment Timeline (A typical scenario)

The process of the treatment of a TB patient takes 6 months in total; so the axiom $(p : Patient)_{6m}$ states that an individual p is a patient for 6 months. In the initial phase of the treatment, the patient should take various medicines. In order to model this fact, we use the *TakeMedicine* role which keeps track of the medicines which a patient may take during the treatment. We know that the patient should take *Rifampin* for two months (60 days). So we define an individual of $Rifampin$ ($(r_1 : Rifampin)_{60d}$); then model the fact with $TakeMedicine(p, r_1)_{60d}$. Since $(r_1 : Rifampin)_{60d}$ and $TakeMedicine(p, r_1)_{60d}$ must be true on the same interval, we simply model it with $(r_1 : Rifampin)_{60d} \wedge TakeMedicine(p, r_1)_{60d}$. Analogously we define the axioms for taking other medicines. Since all the medicines of the initial phase should be taken simultaneously, it is enough to make a simple combination (φ_{inimed}) conjoining the axioms.

$$
\begin{aligned}
\varphi_{inimed} = {} & (\mathrm{r_1{:}Rifampin})_{60d} \wedge \mathrm{TakeMedicine(p,r_1)}_{60d} \\
& \wedge (\mathrm{is_1{:}Isoniazid})_{60d} \wedge \mathrm{TakeMedicine(p,is_1)}_{60d} \\
& \wedge (\mathrm{py{:}Pyrazinamide})_{60d} \wedge \mathrm{TakeMedicine(p,py)}_{60d} \\
& \wedge (\mathrm{et{:}Ethambutol})_{60d} \wedge \mathrm{TakeMedicine(p,et)}_{60d} \\
& \wedge (\mathrm{b6_1{:}VitaminB6})_{60d} \wedge \mathrm{TakeMedicine(p,b6_1)}_{60d}
\end{aligned}
$$

$(p : Patient)_{6m}$ and φ_{inimed} should be true on two intervals I_1 and I_2 which start at the same time. In other words, I_1 *starts with* I_2 where *starts* is one of Allen's relations [83]. Recall, $\langle B\rangle\varphi = \langle\overline{A}\rangle\langle A\rangle\varphi$, mentioned in Section 3.4.1 (on Page 36). In order to model this temporal relation, it is sufficient to add $\Diamond_l\Diamond_r$ to the second formula. Thus, $(p : Patient)_{6m} \wedge \Diamond_l\Diamond_r\varphi_{inimed}$ is the desired combination. Recall that based on the semantics of the language,

$(p : Patient)_{6m}$ can be true only on an interval of 6 months' length while φ_{inimed} can only be true on an interval of 60 days' length; so forgetting to put $\Diamond_l\Diamond_r$ before φ_{inimed} makes the combination unsatisfiable, because $(p : Patient)_{6m} \wedge \varphi_{inimed}$ indicates that $(p : Patient)_{6m}$ and φ_{inimed} are true on the same interval, and it is not possible to have an interval which has two different lengths simultaneously. In some guidelines, we may have another of Allen's relations between two intervals on which the formulas are true. Fortunately, MITDL is able to model the metric version of all of Allen's relations, with the exception of the *during* relation.

The first occurrence of Sputum Smear Test, which lasts 3 days, should be performed on the patient in the three last days of initial phase; so there is a *finishes* relation, one of Allen's relations, between interval I_2 and I_3 on which $(s_1 : SuptumSmearTest)_{3d} \wedge HasSputumSmearTest(p, s_1)_{3d}$ is true. In this formula, *HasSputumSmearTest* is a DL role which keeps track of the sputum tests which are performed for a patient during the treatment. We model the relation by $\varphi_{inimed} \wedge \Diamond_r\Diamond_l((\mathrm{s_1}{:}\mathrm{SuptumSmearTest})_{3d} \wedge \mathrm{HasSputumSmearTest}(\mathrm{p},\mathrm{s_1})_{3d})$.

We define two more DL roles, *HasTestResult* and *IsReferred*. They respectively keep track of the result of performed tests and the patients who are referred to TB Service Direction. The formula $(\mathrm{s_1} : \exists\mathrm{HasTestResult}.\neg\mathrm{Positive})_{1h}$ describes the situation where the result of test s_1 is negative. We have assumed that the result of the test appears at the last hour of the third day of the test. Therefore, there is a *finishes* relation between two intervals I_3 and I_4 on which $(\mathrm{s_1} : \exists\mathrm{HasTestResult}.\neg\mathrm{Positive})_{1h}$ are, respectively, true. We model this by $\Diamond_r\Diamond_l(\mathrm{s_1} : \exists\mathrm{HasTestResult}.\neg\mathrm{Positive})_{1h}$. A similar situation exists for the interval I_3 and the interval on which $(\mathrm{s_1} : \exists\mathrm{HasTestResult}.\mathrm{Positive})_{1h}$ is true. This last formula shows the positive result for the test. When the result is positive, the patient should be referred to TB Service Direction. We model this fact

with $(\text{p:}\exists\text{IsReferred.TBDirectionService})_{1h}$. Immediately after the end of the test, if the result of the test is negative, the first segment of the continuation phase, which lasts two months, is started. Let φ_{con1} be the formula which models this segment. There is a *meet* relation between two intervals I_4 and I_5 on which φ_{con1} is true (I_4 *meets* I_5). Recall, $\Diamond_r$ modeles this relation. See the following formula.

$$\begin{aligned}
&\varphi_{inimed} \wedge \Diamond_r\Diamond_l((\text{s}_1\text{:SuptumSmearTest})_{3d} \wedge \text{HasSputumSmearTest(p,s}_1)_{3d}) \\
&\wedge ((\Diamond_r\Diamond_l(\text{s}_1 : \exists\text{HasTestResult.}\neg\ \text{Positive})_{1h} \wedge \Diamond_r\varphi_{con1}) \\
&\quad \vee (\Diamond_r\Diamond_l(\text{s}_1 : \exists\text{HasTestResult.Positive})_{1h} \\
&\qquad \wedge \Diamond_r(\text{p} : \exists\text{IsReferred.TBDirectionService})_{1h}))
\end{aligned}$$

The process of modeling φ_{con1} is straightforward in the sense that it is sufficient to model that the patient takes three medicines for 60 days.

$$\begin{aligned}
\varphi_{con1} =&(\text{r}_3\text{:Rifampin})_{60d} \wedge \text{TakeMedicine(p,r}_3)_{60d} \\
&\wedge (\text{is}_3\text{:Isoniazid})_{60d} \wedge \text{TakeMedicine(p,is}_3)_{60d} \\
&\wedge (\text{b6}_3\text{:VitaminB6})_{60d} \wedge \text{TakeMedicine(p,b6}_3)_{60d}
\end{aligned}$$

The second Sputum Smear Test is performed in the last three days of the first segment of continuation phase. The encoding process for the second segment of the continuation phase is similar to the previous segment. We leave reader to model the rest of the guideline (φ_{rest}). The formula φ_{TB} is the formulation of the TB guideline.

$$\varphi_{TB} = (\text{p} : \text{Patient})_{6m} \wedge \Diamond_l \Diamond_r (\varphi_{inimed} \wedge$$
$$\Diamond_r \Diamond_l ((\text{s}_1\text{:SuptumSmearTest})_{3d} \wedge \text{HasSputumSmearTest}(\text{p},\text{s}_1)_{3d}) \wedge$$
$$((\Diamond_r \Diamond_l (\text{s}_1 : \exists \text{HasTestResult}.\neg\text{Positive})_{1h} \wedge \Diamond_r (\varphi_{con1} \wedge \Diamond_r \varphi_{rest})) \vee$$
$$(\Diamond_r \Diamond_l (s_1 : \exists \text{HasTestResult.Positive})_{1h} \wedge$$
$$\Diamond_r (\text{p} : \exists \text{IsReferred.TBDirectionService})_{1h})))$$

6.4 Multi treatment of an HIV/AIDS-TB patient

In two previous case studies we showed how IMPNL and MITDL may be used to formalize and to check the consistency of a guideline. In this case study we demonstrate an inconsistent guideline. We check the guideline with the proposed algorithms, and we show how an inconsistency is detected. Let us consider a patient with two diseases: HIV/AIDS and TB. While there are some specific guidelines for this kind of patient, e.g. [131, 132], we see what happens when the two guidelines (in Sections 6.1 and 6.3) are followed simultaneously. Note that since TB is considered as an AIDS-Defining condition, as soon as the patient catches TB, (s)he should start to take the AIDS medicines.

6.4.1 Modeling Multi treatment of an HIV/AIDS-TB patient with IMPNL

Since the patient is in the AIDS stage, $HasAIDS_{1h}$ is true, and the guideline of HIV/AIDS (see Formula 6.5 on Page 138) will be the following formula denoted by ψ_{HIV}^{AIDS}.

$$
\begin{aligned}
\psi_{HIV}^{AIDS} = {} & \text{Elisa}_{1d}^{\#1} \wedge \\
& \Diamond_r(\text{Elisa}_{1d}^{\#2} \wedge \\
& \Diamond_r(\text{WesternBlot}_{1d} \wedge \\
& \Diamond_r(\text{Registration}_{4d} \wedge \\
& \Diamond_r(\top_{90d} \wedge \\
& \Diamond_l\Diamond_r(\text{CD4Check}_{1d} \wedge \\
& \Diamond_l\Diamond_r(\top_{30d} \wedge \\
& \Diamond_l\Diamond_r(\text{Visit}_{1h} \wedge \\
& \Diamond_l\Diamond_r(\text{TakeKaletra}_{30d} \wedge \neg\text{TakeAlfuzosin}_{30d} \wedge \neg\text{TakeCisapride}_{30d} \\
& \wedge \neg\text{TakeRifampin}_{30d} \wedge \neg\text{TakePimozide}_{30d} \wedge \text{TakeTenofovir}_{30d} \\
& \wedge \text{TakeLamivadin}_{30d}))))))))
\end{aligned} \tag{6.13}
$$

The formalization of the new guideline is denoted by $\psi_{HIV-TB} = \Diamond_l\Diamond_r\psi_{HIV}^{AIDS} \wedge\Diamond_l\Diamond_r\psi_{TB}$. The reason for adding $\Diamond_l\Diamond_r$ to the formulation of every guideline is to prevent the occurrence of incorrect clashes. For example, if an event in the first guideline which should be true on the current interval lasts 1 day, and an event of the second guideline on the current interval lasts 3 days, the conjunction of theses events can not be true on the current intervals at the same time; thus, an incorrect clash occurs. We check the satisfiability of ψ_{HIV-TB} using the proposed tableau method. The details are as follows. We put another superscript for the propositions to make a distinction between TB propositions and HIV propositions. This is just a syntactic annotation, and does not change the semantics of the formula. The tableau for ψ_{HIV-TB} is a very big tableau, and it is not possible to show the whole tableau in this thesis.

We make another assumption to make the formula easier to deal with. We assume that the first SputumSmearTest is false. Formula 6.15 demonstrates the annotated version of $\Diamond_r \psi_{HIV-TB}$ denoted by ψ'_{HIV-TB}. Also, we replaced the implications with the equivalent form and equipped it with the required numbers for the selection strategy.

$$
\begin{aligned}
&\psi_{HIV-TB} = \\
&\Diamond_l\Diamond_r(^{HIV}\text{Elisa}^{\#1}_{1d} \wedge \\
&\Diamond_r(^{HIV}\text{Elisa}^{\#2}_{1d} \wedge \\
&\Diamond_r(^{HIV}\text{WesternBlot}_{1d} \wedge \\
&\Diamond_r(^{HIV}\text{Registration}_{4d} \wedge \\
&\Diamond_r(\top_{90days} \wedge \\
&\Diamond_l\Diamond_r(^{HIV}\text{CD4Check}_{1d} \wedge \\
&\Diamond_l\Diamond_r(\top_{30days} \wedge \\
&\Diamond_l\Diamond_r(^{HIV}\text{Visit}_{1h} \wedge \\
&\Diamond_l\Diamond_r(^{HIV}\text{TakeKaletra}_{30d} \wedge \neg^{HIV}\text{TakeAlfuzosin}_{30d} \wedge \\
&\neg^{HIV}\text{TakeCisapride}_{30d} \wedge \neg^{HIV}\text{TakeRifampin}_{30d} \wedge \\
&\neg^{HIV}\text{TakePimozide}_{30d} \wedge^{HIV} \text{TakeTenofovir}_{30d} \wedge \\
&^{HIV}\text{TakeLamivadin}_{30d}))))))))) \wedge \\
&\Diamond_l\Diamond_r(^{TB}\text{TakeRifampin}_{60d} \wedge^{TB} \text{TakeIsoniazid}_{60d} \wedge^{TB} \text{TakePyrazinamide}_{60d} \\
&\wedge^{TB} \text{TakeEthambutol}_{60d} \wedge^{TB} \text{TakeVitaminB6}_{60d} \wedge \\
&\Diamond_r\Diamond_l(\neg^{TB}\text{SputumSmearTest}^{\#1}_{3d} \rightarrow \\
&\Diamond_r(^{TB}\text{TakeRifampin}_{60d} \wedge^{TB} \text{TakeIsoniazid}_{60d} \wedge^{TB} \text{TakeVitaminB6}_{60d} \wedge \\
&\Diamond_r\Diamond_l(\neg^{TB}\text{SputumSmearTest}^{\#2}_{3d} \rightarrow \\
&\Diamond_r(^{TB}\text{TakeRifampin}_{60d} \wedge^{TB} \text{TakeIsoniazid}_{60d} \wedge \\
&^{TB}\text{TakeVitaminB6}_{60d} \wedge \\
&\Diamond_r(^{TB}\text{SputumSmearTest}^{\#3}_{3d} \rightarrow \Diamond^{TB}_r\text{TBServicesDirection}_{1h})) \\
&\wedge (^{TB}\text{SputumSmearTest}^{\#2}_{3d} \rightarrow \Diamond^{TB}_r\text{TBServicesDirection}_{1h}))) \\
&\wedge (^{TB}\text{SputumSmearTest}^{\#1}_{3d} \rightarrow \Diamond^{TB}_r\text{TBServicesDirection}_{1h})) \qquad (6.14)
\end{aligned}
$$

$$
\begin{aligned}
&\psi'_{HIV-TB} = \\
&\ {}^{1}\Diamond_r^-({}^{2}\Diamond_l^-\Diamond_r^*({}^{HIV}\text{Elisa}_{1d}^{\#1}\wedge \\
&\ \Diamond_r^*({}^{HIV}\text{Elisa}_{1d}^{\#2}\wedge \\
&\ \Diamond_r^*({}^{HIV}\text{WesternBlot}_{1d}\wedge \\
&\ \Diamond_r^*({}^{HIV}\text{Registration}_{4d}\wedge \\
&\ \Diamond_r^*(\top_{90d}\wedge \\
&\ {}^{4}\Diamond_l^-\Diamond_r^*({}^{HIV}\text{CD4Check}_{1d}\wedge \\
&\ {}^{8}\Diamond_l^-\Diamond_r^*(\top_{30d}\wedge \\
&\ {}^{16}\Diamond_l^-\Diamond_r^*({}^{HIV}\text{Visit}_{1h}\wedge \\
&\ {}^{32}\Diamond_l^-\Diamond_r^*({}^{HIV}\text{TakeKaletra}_{30d}\wedge \\
&\ \neg{}^{HIV}\text{TakeAlfuzosin}_{30d}\wedge\neg{}^{HIV}\text{TakeCisapride}_{30d}\wedge \\
&\ \neg{}^{HIV}\text{TakeRifampin}_{30d}\wedge\neg{}^{HIV}\text{TakePimozide}_{30d}\wedge \\
&\ {}^{HIV}\text{TakeTenofovir}_{30d}\wedge{}^{HIV}\text{ TakeLamivadin}_{30d})))))))))\wedge \\
&{}^{64}\Diamond_l^-\Diamond_r^*({}^{TB}\text{TakeRifampin}_{120d}\wedge{}^{TB}\text{ TakeIsoniazid}_{120d}\wedge \\
&\ {}^{TB}\text{TakePyrazinamide}_{120d}\wedge{}^{TB}\text{ TakeEthambutol}_{120d}\wedge{}^{TB}\text{ TakeVitaminB6}_{120d}\wedge \\
&\ {}^{128}\Diamond_r^-\Diamond_l^*({}^{TB}\text{SputumSmearTest}_{3d}^{\#2}\vee \\
&\ \Diamond_r^*({}^{TB}\text{TakeRifampin}_{60d}\wedge{}^{TB}\text{ TakeIsoniazid}_{60d}\wedge{}^{TB}\text{ TakeVitaminB6}_{60d}\wedge \\
&\ \Diamond_r^*(\neg{}^{TB}\text{SputumSmearTest}_{3d}^{\#3}\vee \\
&\ \Diamond_r^*({}^{TB}\text{TBServicesDirection}_{1h})))\wedge \\
&\ (\neg{}^{TB}\text{SputumSmearTest}_{3d}^{\#2}\vee\Diamond_r^*({}^{TB}\text{TBServicesDirection}_{1h})))) \qquad (6.15)
\end{aligned}
$$

In this case study $\kappa = 1127d3h$. As can be seen in the Figures C.9, C.10, C.11 and C.12, a clash was detected, due to the occurrence of ${}^{\text{TB}}\text{TakeRifampin}_{120d}$

and $\neg^{\text{HIV}}\text{TakeRifampin}_{30d}$ (Nodes 40 and 48 in Figure 6.9). Figure 6.9 clarifies the situation. A dashed line exhibits the time at which a clash occurs. This guideline states the patient should take *Rifampin* for the treatment of TB while (s)he should not take *Rifampin* since it has a contraindication with *Kaletra*. In other words, the closed branch exhibits a scenario of the treatment which can be harmful for the patient.

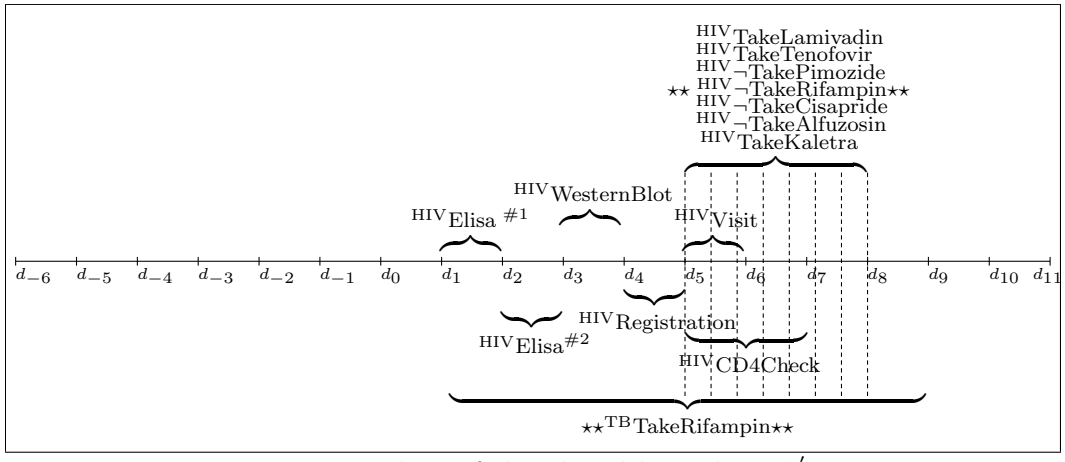

Figure 6.9: Timeline of the closed branch in ψ'_{HIV-TB}

6.4.2 Modeling Multi treatment of an HIV/AIDS-TB patient with MITDL

Since the patient is in the AIDS stage, the patient is considered as an individual of the concept HasAIDS, and the guideline of HIV/AIDS will be the following formula denoted by φ^{AIDS}_{HIV}.

$$
\begin{aligned}
\varphi_{HIV}^{AIDS} = {} & (\text{p:Patient})_{120yr} \wedge \\
& \Diamond_l \Diamond_r ((\text{e1:Elisa})_{1d} \wedge \text{PerformTest(p, e1)}_{1d} \wedge \\
& \Diamond_r ((\text{e2:Elisa})_{1d} \wedge \text{PerformTest(p,e2)}_{1d} \wedge \\
& \Diamond_r ((\text{w:WesternBlot})_{1d} \wedge \text{PerformTest(p,w)}_{1d} \wedge \\
& \Diamond_r ((\text{r:Registration})_{4d} \wedge \text{AdministrativeAction(p,r)}_{4d} \wedge \\
& \Diamond_r (\top_{90d} \wedge \\
& \Diamond_l \Diamond_r ((\text{c:CD4Check})_{1d} \wedge \text{PerformTest(p,c)}_{1d} \wedge \\
& \Diamond_l \Diamond_r (\top_{30d} \wedge \\
& \Diamond_l \Diamond_r (((\text{p:HasAIDS})_{1h} \wedge (\text{p:}\exists\text{Visitedby.Doctor})_{1h} \wedge \\
& \Diamond_l \Diamond_r ((\text{k:Kaletra})_{30d} \wedge (\text{t:Tenofovir})_{30d} \wedge (\text{l:Lamivadin})_{30d} \\
& \wedge \text{TakeMedicine(p,k)}_{30d} \wedge \text{TakeMedicine(p,t)}_{30d} \\
& \wedge \text{TakeMedicine(p,l)}_{30d}))))))))) \qquad (6.16)
\end{aligned}
$$

The formalization of the new guideline is denoted by $\varphi_{HIV-TB} = \Diamond_l \Diamond_r \varphi_{HIV}^{AIDS} \wedge \Diamond_l \Diamond_r \varphi_{TB}$. Note that the reason for adding $\Diamond_l \Diamond_r$ was explained in the previous section. We check the satisfiability of φ_{HIV-TB} using the proposed tableau method. The details are as follows. Similar to the previous section, we add superscripts to the axioms to make a distinction between TB axioms and HIV axioms. Since the tableau for φ_{HIV-TB} is very big, we assume that the first SputumSmearTest is false. Formula 6.17 demonstrates the annotated version of $\Diamond_r \varphi_{HIV-TB}$ denoted by φ'_{HIV-TB}. As before, we have replaced the implications with the equivalent form and equipped it with the required numbers for the selection strategy.

$$
\begin{aligned}
&\varphi_{HIV-TB} = \Diamond_l\Diamond_r((\text{p:Patient})_{120yr}\wedge \\
&\quad \Diamond_l\Diamond_r((\text{e1:Elisa})_{1d} \wedge \text{PerformTest(p, e1)}_{1d}\wedge \\
&\quad\quad \Diamond_r((\text{e2:Elisa})_{1d} \wedge \text{PerformTest(p,e2)}_{1d}\wedge \\
&\quad\quad\quad \Diamond_r((\text{w:WesternBlot})_{1d} \wedge \text{PerformTest(p,w)}_{1d}\wedge \\
&\quad\quad\quad\quad \Diamond_r((\text{r:Registration})_{4d} \wedge \text{AdministrativeAction(p,r)}_{4d}\wedge \\
&\quad\quad\quad\quad\quad \Diamond_r(\top_{90d}\wedge \\
&\quad\quad\quad\quad\quad\quad \Diamond_l\Diamond_r((\text{c:CD4Check})_{1d} \wedge \text{PerformTest(p,c)}_{1d}\wedge \\
&\quad\quad\quad\quad\quad\quad\quad \Diamond_l\Diamond_r(\top_{30d}\wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad \Diamond_l\Diamond_r(((\text{p:HasAIDS})_{1h} \wedge (\text{p:}\exists\text{Visitedby.Doctor})_{1h}\wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad \Diamond_l\Diamond_r((\text{k:Kaletra})_{30d} \wedge (\text{t:Tenofovir})_{30d} \wedge (\text{l:Lamivadin})_{30d}\wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{TakeMedicine(p,k)}_{30d} \wedge \text{TakeMedicine(p,t)}_{30d}\wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{TakeMedicine(p,l)}_{30d}))))))))))\wedge \\
&\quad \Diamond_l\Diamond_r((\text{p : Patient})_{6m}\wedge \\
&\quad\quad \Diamond_l\Diamond_r((\text{r}_1\text{:Rifampin})_{60d} \wedge \text{TakeMedicine(p,r}_1)_{60d} \wedge (\text{is}_1\text{:Isoniazid})_{60d} \\
&\quad\quad\quad \wedge \text{TakeMedicine(p,is}_1)_{60d} \wedge (\text{py:Pyrazinamide})_{60d} \wedge \text{TakeMedicine(p,py)}_{60d} \\
&\quad\quad\quad \wedge (\text{et:Ethambutol})_{60d} \wedge \text{TakeMedicine(p,et)}_{60d} \wedge (\text{b6}_1\text{:VitaminB6})_{60d} \\
&\quad\quad\quad \wedge \text{TakeMedicine(p,b6}_1)_{60d}\wedge \\
&\quad\quad\quad \Diamond_r\Diamond_l((\text{s}_1\text{:SuptumSmearTest})_{3d} \wedge \text{HasSputumSmearTest(p,s}_1)_{3d})\wedge \\
&\quad\quad\quad ((\Diamond_r\Diamond_l(\text{s}_1 : \exists\text{HasTestResult.}\neg\text{Positive})_{1h} \wedge \Diamond_r(\varphi_{con1} \wedge \Diamond_r\varphi_{rest}))\vee \\
&\quad\quad\quad\quad (\Diamond_r\Diamond_l(\text{s}_1 : \exists\text{HasTestResult.Positive})_{1h}\wedge \\
&\quad\quad\quad\quad\quad \Diamond_r(\text{p:}\exists\text{IsReferred.TBDirectionService})_{1h}))))
\end{aligned}
$$

$$
\begin{aligned}
&\varphi'_{HIV-TB} = \\
&{}^{1}\Diamond_r^-({}^{2}\Diamond_l^-\Diamond_r^*(((\neg\text{Kaletra} \sqcap \neg\text{Norvir} \sqcap \neg\text{Aptivus}) \sqcup \text{ProteaseInhibitors} = \top)_{120yr} \wedge \\
&\quad (\forall\text{TakeMedicine}.\neg\text{Rifampin} \sqcup \forall\text{TakeMedicine}.\neg\text{ProteaseInhibitors} = \top)_{120yr}) \wedge \\
&\quad {}^{4}\Diamond_l^-\Diamond_r^*((\text{p:Patient})_{120yr} \wedge \\
&\quad\quad {}^{8}\Diamond_l^-\Diamond_r^*((\text{e1:Elisa})_{1d} \wedge \text{PerformTest(p, e1)}_{1d} \wedge \\
&\quad\quad\quad \Diamond_r^*((\text{e2:Elisa})_{1d} \wedge \text{PerformTest(p,e2)}_{1d} \wedge \\
&\quad\quad\quad\quad \Diamond_r^*((\text{w:WesternBlot})_{1d} \wedge \text{PerformTest(p,w)}_{1d} \wedge \\
&\quad\quad\quad\quad\quad \Diamond_r^*((\text{r:Registration})_{4d} \wedge \text{AdministrativeAction(p,r)}_{4d} \wedge \\
&\quad\quad\quad\quad\quad\quad \Diamond_r^*(\top_{90d} \wedge^{16} \Diamond_l^-\Diamond_r^*((\text{c:CD4Check})_{1d} \wedge \text{PerformTest(p,c)}_{1d} \wedge \\
&\quad\quad\quad\quad\quad\quad\quad {}^{32}\Diamond_l^-\Diamond_r^*(\top_{30d} \wedge^{64} \Diamond_l^-\Diamond_r^*(((\text{p:HasAIDS})_{1h} \wedge (\text{p:}\exists\text{Visitedby.Doctor})_{1h} \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad {}^{128}\Diamond_l^-\Diamond_r^*((\text{k:Kaletra})_{30d} \wedge (\text{t:Tenofovir})_{30d} \wedge (\text{l:Lamivadin})_{30d} \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{TakeMedicine(p,k)}_{30d} \wedge \text{TakeMedicine(p,t)}_{30d} \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{TakeMedicine(p,l)}_{30d}))))))))))) \wedge \\
&\quad {}^{256}\Diamond_l^-\Diamond_r^*((\text{p} : \text{Patient})_{6m} \wedge \\
&\quad\quad {}^{512}\Diamond_l^-\Diamond_r^*((\text{r}_1\text{:Rifampin})_{60d} \wedge \text{TakeMedicine(p,r}_1)_{60d} \wedge (\text{is}_1\text{:Isoniazid})_{60d} \wedge \\
&\quad\quad \text{TakeMedicine(p,is}_1)_{60d} \wedge (\text{py:Pyrazinamide})_{60d} \wedge \text{TakeMedicine(p,py)}_{60d} \wedge \\
&\quad\quad (\text{et:Ethambutol})_{60d} \wedge \text{TakeMedicine(p,et)}_{60d} \wedge \\
&\quad\quad (\text{b6}_1\text{:VitaminB6})_{60d} \wedge \text{TakeMedicine(p,b6}_1)_{60d} \wedge \\
&\quad\quad {}^{1024}\Diamond_r^-\Diamond_l^*((\text{s}_1\text{:SuptumSmearTest})_{3d} \wedge \text{HasSputumSmearTest(p,s}_1)_{3d}) \wedge \\
&\quad\quad\quad ((^{2048}\Diamond_r^-\Diamond_l^*(\text{s}_1 : \exists\text{HasTestResult.}\neg\text{Positive})_{1h} \wedge \Diamond_r^*(\varphi_{con1} \wedge \Diamond_r^*\varphi_{rest})) \vee \\
&\quad\quad\quad\quad (^{4096}\Diamond_r^-\Diamond_l^*(\text{s}_1 : \exists\text{HasTestResult.Positive})_{1h} \wedge \\
&\quad\quad\quad\quad\quad \Diamond_r^*(\text{p: }\exists\text{IsReferred.TBDirectionService})_{1h}))))) \qquad (6.17)
\end{aligned}
$$

In this case study $\kappa = 1133d3h$. As can be seen in the Figures C.13, C.14, C.15, C.16, C.17, and C.18, three clashes were detected. Also, for the ease of understanding, we have provided the abstract of the tableau in Figure 6.10. The clashes are as follows:

- $(\mathrm{k} : \text{ProteaseInhibitors})_{120yr}$ (Node 81) and $(\mathrm{k} : \neg\text{ProteaseInhibitors})_{120yr}$ (Node 83) in Figure C.18) while both formula are true on the same interval, i.e., $[e_1, e_3]$.

  The branch, which contains nodes 57 and 82, is always closed because of having Fact1 and Fact2 (see Section 6.2.1) and the fact that Kaletra is a protease inhibitor medicine. Note that this closedness occurs whether the patient has TB or not.

- $(\mathrm{k} : \text{Kaletra})_{30d}$ (Node 57) in Figure C.16 and $(\mathrm{k} : \neg\text{Kaletra})_{120yr}$ (Node 82) in Figure C.18, while $[e_7, e_{10}] \cap [e_1, e_3] = [e_7, e_{10}]$.

  The branch, which contains nodes 57 and 82, is always closed because of having Fact1 and Fact2 (see Section 6.2.1) and the fact that the patient takes Kaletra for the treatment of AIDS disease. Note that this closedness occurs whether patient has TB or not.

- $(\mathrm{k} : \text{ProteaseInhibitors})_{120yr}$ (Node 81) and $(\mathrm{k} : \neg\text{ProteaseInhibitors})_{120yr}$ (Node 83) in Figure C.18, while both formula are true on the same interval, i.e., $[e_1, e_3]$.

  The branch, which contains nodes 81 and 83, is closed because the patient takes *Rifampin* for the treatment of TB while Fact2 forces the patient to not take *Rifampin* because of taking a protease medicine (e.g., Kaletra). This inconsistency never occurs when we only consider a patient who has TB or AIDS, but not both.

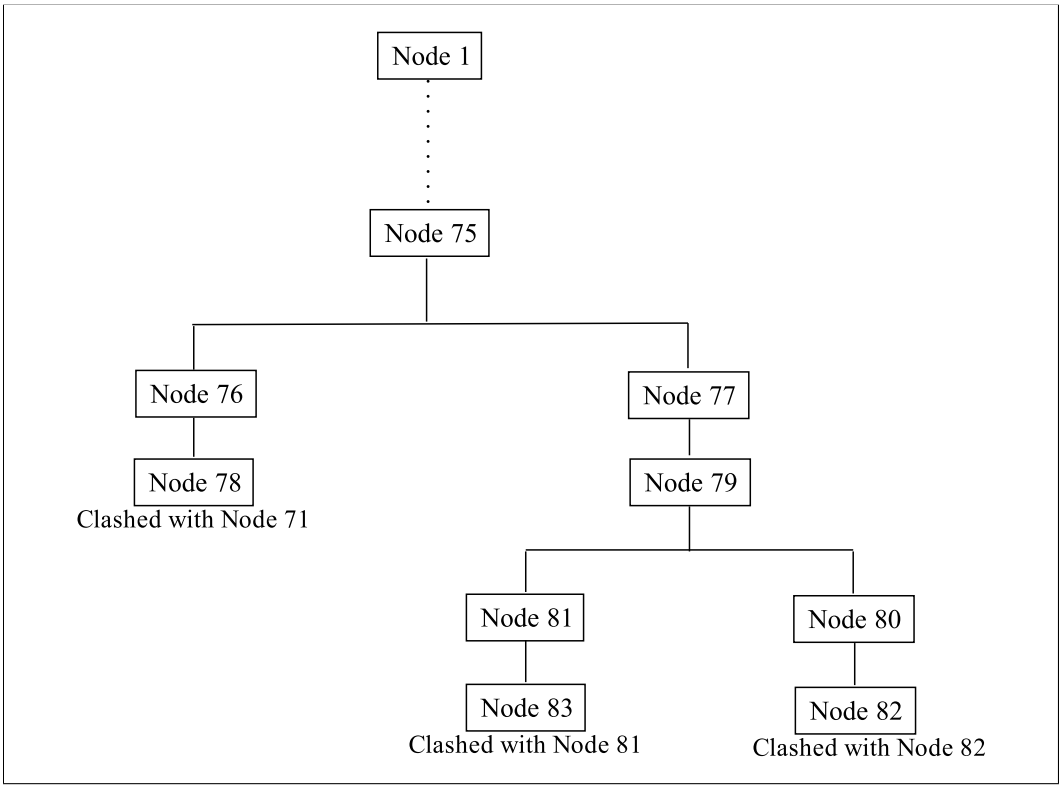

Figure 6.10: Abstract of the tableau for φ'_{HIV-TB} (Modeled in MITDL)

6.5 A comparison of Guideline modelling languages

In this section, we compare some guideline modeling languages: Asbru, EON, GLIF, PRODIGY and PRO*forma* with MITDL. These languages (except MITDL) have been already compared in [133, 134].

6.5.1 Brief description of other languages

- EON [135, 136]

  EON, developed at Stanford University, is a component-based architecture for building protocol-based decision-support systems [137]. This architecture contains a patient data information model, a medical concept model and a task-based guideline model [133]. The medical concept model in the architecture is used to encode the domain ontologies [136, 138]. It is worthwhile mentioning that, due to the use of a task-based approach for defining guidelines, different decision-support services, defined by EON, can be implemented using alternative techniques [139].

- GLIF [140]

  GLIF was developed by the collaboration of Stanford Medical Informatics, Harvard University, McGill University and Columbia University. It was particularly designed to model and to execute clinical practice guidelines. GLIF uses UML diagrams to model guidelines in terms of structured steps which represent clinical actions and decisions. Also, using this language, one can model domain ontologies [136, 138].

- PRO*forma* [135, 136]

PRO*forma* was developed at the Advanced Computation Laboratory of Cancer Research, UK [134]. It is a general purpose process modeling language using a task-based formalism which models clinical processes as a collection of plans, decisions, enquires and actions [141]. While the task model in PRO*forma* is deliberately simple in order to be easy to learn, it is expressive enough to model many guidelines [139]. It is worthwhile to note that PRO*forma* combines logic programming and object-oriented modeling [133], and provides expressive constructs for describing different aspects of a guideline.

- Asbru [135, 136, 142]

  Asbru is a text-based, machine-readable [143] and general purpose process modeling language. It models complex clinical guidelines as a network of tasks; each task may consist of some steps which have a specific function or a goal [135]. On the other hand, Asbru has a formal semantics [144] defined in the style of state charts [145]. This enables one to use a theorem prover or a model checker to verify a guideline, modeled in Asbru, which consequently has the potential to improve the quality of guidelines.

- PRODIGY

  The PRODIGY project was developed in the University of Newcastle upon Tyne [133]. The aim of this project was to produce the simplest, most readily comprehensible model necessary to represent guidelines for the management of the primary care chronic diseases (e.g., asthma, hypertension) [134, 146].

Now, we compare these languages with MITDL.

6.5.2 Comparison Criteria

Since the aims of designing these languages were different, we discuss some criteria that we believe are reasonable choice for comparing the languages (see Table 6.5). These criteria are as follows:

- Executability

  EON, PRO*forma*, GLIF, PRODIGY and Asbru are designed to model and to execute a CPG, but the goal of designing MITDL is to model a guideline and check its consistency before that the CPG is used for any patient; so MITDL is not an executable language.

- Formal Semantics

  EON, GLIF and PRODIGY have no formal semantics [133] while the semantics of Asbru and PRO*forma* are defined based on a formalism called Structural Operational Semantics (SOS) [147, 148]. MITDL is also has a formal semantics defined in Section 5.2 (on Page 77).

- Tool support

  So far, no tools has been developed for MITDL, but there are some tools for other languages. Some of these tools are as follows:

  - EON
    - Protégé-2000
  - GLIF
    - Protégé
  - PRO*forma*
    - Arezzo®, Performer, Tallis [141]
  - Asbru

    - Asbru Interpreter, AsbruView
  - PRODIGY
    - Protégé

- Graphical/Text-based

  EON, PRODIGY and GLIF are graphical languages while MITDL, Asbru, Pro*forma* [149] are text based languages. However, Asbru and PRO*forma* have some tools which help an expert to design a CPG in a graphical environment.

- Ability to define temporal constraints

  Asbru and PRODIGY are able to specify the start time, the end time and the duration of a medical activity (action) whereas EON and GLIF are able to specify only the start time and the duration of an activity. PRO*forma* specifies an interval by the start time and the end time. In MITDL neither start time nor end time of a formula can be specified, but it is possible to specify the duration of an axiom in the MITDL formulas which do not contain $\Diamond_r^+, \Diamond_r^-, \Diamond_l^+, \Diamond_l^-$ in their annotated version.

- Ability to define the domain knowledge (e.g., Medical knowledge)

  While Asbru cannot model medical knowledge, a hierarchy of medical concepts can be created in PRODIGY, EON and GLIF. MITDL is also able to model any medical ontology that can be expressed by the $\mathcal{ALC}$ language.

- Linkage to existing medical ontologies

  PRO*forma* and Asbru provide no intrinsic support to use existing medical ontologies, but some platforms may provide this ability (e.g., PRO*forma*: HeCaSe2 [150] supports the UMLS ontology, and Asbru: Uruz supports

the ICD-9-CM, CPT-4, LOINC-3 ontologies [151]). PRODIGY, GLIF and EON can be bound to existing medical ontologies through the defined medical concepts in the hierarchies. Any $\mathcal{ALC}$ (fragment of) medical ontologies can be automatically transformed to MITDL.

- Formal analysis of CPGs

  Since EON, GLIF and PRODIGY have no formal semantics, it is not possible to formally verify the guidelines modeled in these languages. Consider the following criteria.

  - Checking consistency of a CPG

    Most languages provide some (in)formal mechanisms (e.g., testing) to make sure that a model is unambiguous and syntactically correct [147]. For example, Duftschmid and Miksch have designed an approach to check the consistency of a guideline in Asbru. They look for different anomalies, e.g., unsatisfiable conditions, redundant parameter-value pairs within conditions and ambiguous state transitions. In MITDL, we can easily use the tableau algorithm to check the consistency of a guideline.

  - Satisfaction of properties

    * Model checking

      Asbru can be automatically translated [144, 147] into the internal representation used by the SMV model checker [136, 142], but only a restricted set of properties can be verified [147]. This set contains the properties that can be formulated in ACTL. Note that it is also possible to verify ECTL properties using this method [144].

    * Theorem proving

Asbru can be also automatically translated [144, 147] into the internal representation used by the KIV theorem prover [135, 152]. In this case, the properties are expressed in a variant of Interval Temporal Logic [148].

- Consistency checking of joint CPGs for a pateint with multiple diseases

  Except for MITDL, no other languages can check the consistency of joint CPGs.

- Complexity of analysis

  Satisfiability checking of a simple formula in MITDL has PSPACE complexity while the satisfiability checking of a generic formula has 2EXPTIME. In Asbru, every single component of a plan is checked for anomalies within its own scope in polynomial time in the number of plans. Also, detecting anomalies between two or more components of a single plan needs polynomial time in the number of plans. Finally, the complexity of checking whole plan hierarchy for anomalies, which may originate from dependencies between two or more plans of the hierarchy, is EXPTIME [153].

			Asbru	GLIF	PRO*forma*	EON	PRODIGY	MITDL
1	Executability		Yes	Yes	Yes	Yes	Yes	No
2	Formal Semantic		Yes	No	Yes	No	No	Yes
3	Tool Support		Asbru Interpreter, AsbruView	Protégé	Arezzo, Performer, Tallis	Protégé-2000	Protégé	No tools so far
4	Graphical(G)/Text-based(T)		T	G	Text	G	G	T
5	Temporal Constraint	StartTime	+	+	+	+	+	
		EndTime	+		+		+	
		Duration	+	+		+	+	+/- [1]
6	Defining domain knowledge		No intrinsic way	Hierarchies of concepts	No intrinsic possibility	Hierarchies of concepts	Hierarchies of concepts	Any $\mathcal{ALC}$ ontology
7	Linkage to Existing Ontologies		No intrinsic way but through platform	Binding to existing ontologies through the defined medical concepts	No intrinsic way but through platform	Binding to existing ontologies through the defined medical concepts	Binding to existing ontologies through the defined medical concepts	Existing Ontologies can be automatically transformed to MITDL
8	Checking consistency of a CPG		Yes	Yes	Yes	Yes	Yes	Yes
9	Model checking		SMV	N/A	N/K	N/A	N/A	Future Works!
10	Theorem proving		KIV	N/A	N/K	N/A	N/A	Future Works!
11	Consistency of concurrent CPGs		No ability	No ability	No ability	No ability	No ability	Able to check
12	Complexity of formal reasoning		ExpTime	N/K	N/K	N/K	N/K	2NExpTime

N/A : Not Applicable

N/K : Not Known

[1] The length of an interval for axioms and some formulas can be specified.

Table 6.5: Comparison of EON, GLIF, PRODIGY, PRO*forma* and MITDL

6.6 Conclusion

In this chapter, we presented some case studies from the domain of medicine. We showed how IMPNL and MITDL are used to model some clinical practice guidelines. Also, we used the tableau-based algorithm for checking the satisfiability of IMPNL (resp. MITDL) formulas to check the consistency of the case studies. The first two guidelines were consistent, and the tableau-based algorithm detected a hidden inconsistency in the third guideline. Note that the domain of medicine contains a huge amount of knowledge. Designers of clinical practice guidelines may not be able to maintain all of the knowledge in their mind during the process of designing the guidelines; so they may not be able to detect such inconsistencies. Therefore, these logics provide a reliable way to check the consistency of clinical practice guidelines.

Finally, we compared MITDL with five CPG modeling languages: Asbru, GLIF, PRO*forma*, EON and PRODIGY. As seen in Table 6.5, one important advantage of using MITDL is to have the ability to check the consistency of concurrent CPGs while other languages are not able to provide such a feature. Note that, in reality, there are many patients who suffer from different diseases at the same time, and concurrent CPGs should be used for them to heal their diseases.

Since the process of designing and modeling a guideline is an iterative process, and we have not developed any tools for these logics yet, the construction of a tableau for the guideline formula is very time consuming and error prone.

Chapter 7

Conclusion and Future work

In this thesis, we introduced a metric interval based temporal description logic (MITDL). We first proposed a new metric interval-based temporal logic, called IMPNL, inspired by MPNL$_l$. Then, we developed a sound and complete tableau-based algorithm for checking the satisfiability of an IMPNL formula. Next, we combined a restricted version of IMPNL with the description logic $\mathcal{ALC}$ in order to design MITDL. We developed two sound and complete tableau-based algorithms (one for simple formulas and one for generic formulas) for checking the satisfiability of MITDL formulas. We also proved that checking the satisfiability of a MITDL formula with these algorithms always terminates. We proved that the complexity of the algorithms are, respectively, PSpace and 2ExpTime. Thus, the satisfiability checking service in MITDL is decidable.

In this research, we (partially) addressed the following issues:

- No tableau-based algorithm exists for MPNL$_l$.
  MPNL$_l$ was introduced by Dr.Bresolin in 2010. To the best of our knowledge, no tableau-based algorithm has been designed for checking the

satisfiability of MPNL$_l$ formulas. In this thesis, we designed a metric interval-based temporal logic, called IMPNL, inspired by MPNL$_l$. We also developed a sound and complete tableau-based algorithm for checking the satisfiability of an IMPNL formula. Moreover, we proved that the construction of a tableau for a formula always terminates. We showed that the complexity of IMPNL is PSPACE. Therefore, satisfiability checking in IMPNL is decidable. We believe that the proposed algorithm for IMPNL can be modified in order to check the satisfiability of MPNL$_l$ formulas, because IMPNL is a fragment of MPNL$_l$. However, we do not have homogeneity in MPNL$_l$ which possibly presents new problems.

- No metric interval-based temporal description language exists.

  In order to model the processes existing in some domains, there should be a logic which is able to model both the dynamic and the static aspects of the processes. Moreover, in some of these domains, the duration of some activities is generally restricted to a specific amount of time. Therefore, the logic must be a metric logic in the sense that a user can specify the duration of an activity. As we already mentioned in Section 5.4 (on Page 5.4), the existing interval-based temporal description logics non-metric or they do not have decidable satisfiability checking algorithm.

  Note that IMPNL and MITDL have some restrictions which we mentioned in Section 4.1.2 (on Page 54). In order to address these issues we could increase the expressivity of the logic, e.g., adding full negation to the temporal part. Unfortunately, increasing the expressivity of the logic may make the satisfiability checking reasoning undecidable.

7.1 Future Work

Some possible future work includes the following.

- Optimizing the tableau-based algorithm for satisfiability checking of a generic MITDL formula; e.g, using Normalization and Encoding Techniques [42].
- Developing tools.
  So far, no tools has been developed for modeling a process with MITDL. This tool should provide the following facilities:
  - An environment for writing MITDL formulas.
  - Automated tableau-style theorem prover for MITDL (a MITDL reasoner).
  - The ability to import any $\mathcal{ALC}$ ontology.
    For instance, the domain of medicine includes a huge amount of domain information, contained in different ontologies, e.g., SNOMED CT, Galen, Vaccine [154]. We should automatically transform them to the forms which can be expressed in MITDL.
- Developing an approach for model checking in IMPNL.
- Developing an approach for model checking in MITDL.
- Extending the description logic part of MITDL with the following items:
  - (Qualifying) Number restrictions ($\mathcal{ALCN}$, $\mathcal{ALCQ}$)
    In various domains, there are some situations where we need to model quantitative restrictions. Some of these restrictions cannot be modeled in MITDL. For example, "The normal resting adult human heart rate ranges from 60-100 beats per minute" [155].

– Nominals

Consider the example: the patients who are hospitalized in Canada must have a valid health card during their treatments in the hospital. In this example, Canada can be modeled as a nominal.

– Concrete domains ($\mathcal{ALC}(\mathcal{D})$)

Sometimes we need to model some feature, e.g., age, height and weight of a patient. Then we are able to model some requirements using these features. For instance, a child patient is a patient whose age is less than 15 years.

Bibliography

[1] Snomed ct, http://www.nlm.nih.gov/research/umls/snomed/snomed_main.html, (accessed November 17,2014).

[2] Snomed ct, http://en.wikipedia.org/wiki/snomed_ct, (accessed August 17, 2014).

[3] Ontology (information science), https://www.princeton.edu/ achaney /tmve/wiki100k/docs/ontology_(information_science).html, (accessed November 21, 2014).

[4] Decidability (logic), http://en.wikipedia.org/wiki/decidability_(logic), (accessed November 18, 2014).

[5] Description logic, http://en.wikipedia.org/wiki/description_logic, (accessed June 27, 2012).

[6] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope further. In *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*. Citeseer, 2008.

[7] Franz Baader. Description logics. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine

Rousset, and Renate Schmidt, editors, *Reasoning Web. Semantic Technologies for Information Systems*, volume 5689 of *Lecture Notes in Computer Science*, pages 1–39. Springer Berlin / Heidelberg, 2009.

[8] Franz Baader and Werner Nutt. Basic description logics. In *The description logic handbook*, pages 43–95. Cambridge University Press, 2003.

[9] Alessandro Artale and Enrico Franconi. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence*, 30(1):171–210, 2000.

[10] Alessandro Artale and Carsten Lutz. A correspondence between temporal description logics. *Journal of Applied Non-Classical Logics*, 14(1-2):209–233, 2004.

[11] Davide Bresolin, Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Metric propositional neighborhood logics. Technical report, European Conference on Artificial Intelligence (ECAI), 2010.

[12] Davide Bresolin, Della Monica, D., Goranko, V., Montanari, A., and Guido Sciavicco. Metric propositional neighbourhood logics: Expressiveness, decidability, and undecidability. In H. Coelho, R. Studer, and Wooldridge M., editors, *Proc. of the 19th European Conference on Artificial Intelligence (ECAI)*, pages 695–700. IOS PRESS, 2010.

[13] Till Mossakowski. Logic for computer scientists, ontologies: Description logics. http://www.informatik.uni-bremen.de/agbkb/lehre/ws09-10/Logik/vl19.pdf, Winter 2009/2010 (accessed May 26, 2013).

[14] Franz Baader and Carsten Lutz. Description logic. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *The Handbook of Modal Logic*, pages 757–820. Elsevier, 2006.

[15] Markus Krötzsch, František Simančík, and Ian Horrocks. A description logic primer. *Arxiv preprint arXiv:1201.4089*, 2012.

[16] Ullrich Hustadt, Renate A Schmidt, and Lilia Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1(251-276):3, 2004.

[17] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, 2012.

[18] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of logic and computation*, 9(3):385–410, 1999.

[19] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. LTCS-Report LTCS-05-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005. (http://lat.inf.tu-dresden.de/research/reports.html).

[20] Carsten Lutz. Description logics with concrete domains - a survey. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.6936 &rep=rep1&type=pdf, 2003 (accessed June 8, 2012).

[21] Alessandro Artale and Enrico Franconi. Temporal description logics. *Foundations of Artificial Intelligence*, 1:375–388, 2005.

[22] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*, chapter 3, pages 135–180. Elsevier, 2008.

[23] Franz Baader. What's new in description logics. *Informatik-Spektrum*, pages 1–9, 2011.

[24] Ming Zuo. *High performance absorption algorithms for terminological reasoning in description logics*. PhD thesis, Concordia University, 2006.

[25] Franz Baader. Description logic tutorial (the 2005 logic summer school of the research school of information sciences and engineering at the australian national university). http://lat.inf.tu-dresden.de/ baader/Talks/, 2005 (accessed May 31, 2012).

[26] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In Steffan Staab and Rudi Studer, editors, *Handbook on ontologies*, pages 21–43. Springer-Verlag, second edition, 2009.

[27] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[28] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. *Automated Reasoning*, pages 292–297, 2006.

[29] Rob Shearer, Boris Motik, and Ian Horrocks. Hermit: A highly-efficient owl reasoner. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008)*, pages 26–27, 2008.

[30] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. Concurrent classification of el ontologies. *The Semantic Web–ISWC*, pages 305–320, 2011.

[31] Julian Alfredo Mendez. A classification algorithm for $\mathcal{ELHI}\{_{R_+}$. Master's thesis, Technische Universität Dresden, 2011.

[32] Mina Aslani and Volker Haarslev. Tbox classification in parallel: Design and first evaluation. In *23rd International Workshop on Description Logics (DL2010)*, page 336, 2010.

[33] Mina Aslani and Volker Haarslev. Towards parallel classification of tboxes. In *Proceedings of the 2008 International Workshop on Description Logics (DL-2008)*, 2008.

[34] Mina Aslani and Volker Haarslev. Concurrent classification of owl ontologies – an empirical evaluation. In *Proceedings of the 2012 International Workshop on Description Logics (DL-2012)*, pages 400–410, June 7-10 2012.

[35] Birte Glimm, Ian Horrocks, Boris Motik, and Giorgos Stoilos. Optimising ontology classification. In *The Semantic Web–ISWC 2010*, pages 225–240. Springer, 2010.

[36] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, New York, NY, USA, 2nd edition, 2007.

[37] Quoc Huy Vu. Subsumption in the description logic in $\mathcal{ELHI}\{_{R_+}$ wrt general tboxes. Master's thesis, TU Dresden, Germany, 2008.

[38] Yevgeny Kazakov. Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies. In *Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI 2009)*, pages 2040–2045, July 11-17 2009.

[39] Arne Meier. Generalized complexity of alc subsumption. *Arxiv preprint arXiv:1205.0722*, 2012.

[40] Evgeny Zolin. Complexity of reasoning in description logics, http://www.cs.man.ac.uk/ ezolin/dl/, (accessed october 21, 2014).

[41] Franz Baader and Ulrike Sattler. Tableau algorithms for description logics. In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 1–18. Springer, 2000.

[42] Ian Horrocks. *Optimising tableaux decision procedures for description logics*. PhD thesis, Manchester, 1997.

[43] Alan JA Robinson and Andrei Voronkov. *Handbook of automated reasoning*, volume 1. Elsevier, 2001.

[44] Jocelyne Faddoul and Volker Haarslev. Algebraic tableau reasoning for the description logic. *Journal of Applied Logic*, 8(4):334–355, 2010.

[45] Jocelyne Faddoul. *Reasoning Algebraically with Description Logics*. PhD thesis, Concordia University, 2011.

[46] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001. 10.1023/A:1013882326814.

[47] Yu Ding. *Tableau-based reasoning for description logics with inverse roles and number restrictions*. PhD thesis, Concordia University, 2008.

[48] Uli Sattler. Modal logic and description logics - week 4. COMP61132: Modal Logic and Description Logics, Manchester University, http://studentnet.cs.manchester.ac.uk/pgt/2010/COMP61132/, 2010-2011 (accessed May 7, 2014).

[49] Jie Bao, Doina Caragea, and Vasant G. Honavar. A distributed tableau algorithm for package-based description logics. In *the 2nd International Workshop On Context Representation And Reasoning (CRR 2006), co-located with ECAI 2006*, 2006.

[50] Jie Bao, Dave Braines, and David Mott. A distributed tableau algorithm for the $\mathcal{ALC}$ description logic. https://www.usukita.org/sites/default/files/2011-05-15_DisTab.pdf, (accessed June 22, 2012).

[51] Jocelyne Faddoul and Volker Haarslev. Optimizing algebraic tableau reasoning for shoq: First experimental results. In *23rd International Workshop on Description Logics (DL2010)*, page 161. Citeseer, 2010.

[52] Petr Kremen. Tableau algorithm for $\mathcal{ALC}$. https://cw.felk.cvut.cz/wiki/_media/courses/a4m33rzn/alc-tableau-algorithm.pdf (accessed August 22, 2013).

[53] Chan Le Le Duc, Myriam Lamolle, and Olivier Curé. A tableaux-based algorithm for description logics with transitive closure of roles in concept and role inclusion axioms. http://www.iut.univ-paris8.fr/files/webfm/recherche/linc/RR201012A.pdf, (accessed May 30, 2012).

[54] Thorsten Liebig and Felix Müller. Parallelizing tableaux-based description logic reasoning. In *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems*, pages 1135–1144. Springer-Verlag, 2007.

[55] Carsten Lutz and Maja Milicic. A tableau algorithm for dls with concrete domains and gcis. In *Description Logics*, 2005.

[56] Carsten Lutz and M. Miličič. A tableau algorithm for description logics with concrete domains and general tboxes. *Journal of Automated Reasoning*, 38(1):227–259, 2007.

[57] Boris Motik, Rob Shearer, and Ian Horrocks. A hypertableau calculus for shiq. In *Proc. of the 2007 Description Logic Workshop (DL 2007)*, volume 250. Citeseer, 2007.

[58] Boris Motik, Rob Shearer, and Ian Horrocks. Optimized reasoning in description logics using hypertableaux. *Automated Deduction–CADE-21*, pages 67–83, 2007.

[59] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research*, 36(1):165–228, 2009.

[60] Felix Müller, Michael Hanselmann, Thorsten Liebig, and Olaf Noppens. A tableaux-based mobile dl reasoner - an experience report. In *Proceedings of the 2006 International Workshop on Description Logics (DL 2006)*, Lake District, UK, 2006.

[61] Satya S Sahoo and Krishnaprasad Thirunarayan. Tableau algorithm

for concept satisfiability in description logic $\mathcal{ALCH}$. *Kno.e.sis Center, Technical report.*, 2009.

[62] Sebastian Rudolph. Foundations of description logics. In *Reasoning Web. Semantic Technologies for the Web of Data*, pages 76–136. Springer, 2011.

[63] Temporal logic, http://en.wikipedia.org/wiki/temporal_logic, (accessed April 9, 2013).

[64] Pierfrancesco Bellini, Riccardo Mattolini, and Paolo Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys (CSUR)*, 32(1):12–42, 2000.

[65] Model checking, http://en.wikipedia.org/wiki/model_checking, (accessed August 14, 2014).

[66] Spin model checker, http://en.wikipedia.org/wiki/spin_model_checker, (accessed August 14, 2014).

[67] Automated theorem proving, http://en.wikipedia.org/wiki/automated_theorem_proving, (accessed August 14, 2014).

[68] The kiv system, http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/kiv/, (accessed August 14, 2014).

[69] Lluis Vila. A survey on temporal reasoning in artificial intelligence. *Ai Communications*, 7(1):4–28, 1994.

[70] Alessandros Artale. Lecture iii: Linear temporal logic. Formal methods, Free University of Bolzano, http://web.iitd.ac.in/ sumeet/slide3.pdf, 2010 (accessed November 29, 2013).

[71] Victor Gutierrez-basulto. Branching temporal description logics: Reasoning about ctlalc and ctlel concepts. Master's thesis, Technische Universität Dresden, January 2009.

[72] Madhavan Mukund. Linear-time temporal logic and büchi automata. *Tutorial talk, Winter School on Logic and Computer Science, Indian Statistical Institute, Calcutta*, 1997.

[73] Yih-Kuen Tsay. Büchi automata and model checking. FLOLAC 2009, National Taiwan University, http://flolac.iis.sinica.edu.tw/flolac09/lib/exe/linear_temporal_logic_4on1.pdf, 2009.

[74] Faron Moller and Alexander Rabinovich. On the expressive power of ctl. In *Logic in Computer Science, 1999. Proceedings. 14th Symposium on*, pages 360–368. IEEE, 1999.

[75] Moonzoo Kim. Temporal logic - ltl, ctl, and ctl*. http://pswlab.kaist.ac.kr/courses/cs402-07, 2007 (accessed March 30, 2013).

[76] Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Interval temporal logics: a journey. *Bulletin of the EATCS*, 3(105):73–99, 2011.

[77] Davide Bresolin and Angelo Montanari. A tableau-based decision procedure for a branching-time interval temporal logic. In *Proc. of the 4th Int. Workshop on Methods for Modalities*, pages 38–53, 2005.

[78] Rajeev Alur and Thomas A Henzinger. Linear temporal logic.

Model Checking, EPFL, http://mtc.epfl.ch/courses/ModelChecking-2007/Notes/13.pdf, 2007 (accessed November 29, 2013).

[79] Djallel Bouneffouf. Temporal logic and its applications in natural language processing. http://hal.archives-ouvertes.fr/docs/00/82/04/39/PDF/Temporallogic.pdf, 2010 (accessed November 27, 2013).

[80] Jean-Michel Couvreur. On-the-fly verification of linear temporal logic. *FM'99—Formal Methods*, pages 253–271, 1999.

[81] E. Allen Emerson. Temporal and modal logic. In *Handbook of theoretical computer science*, pages 995–1072. Elsevier, 1995.

[82] Michael David Fisher, Dov M Gabbay, and Lluis Vila. *Handbook of temporal reasoning in artificial intelligence*, volume 1. Access Online via Elsevier, 2005.

[83] Valentin Goranko, Angelo Montanari, and Guido Sciavicco. A road map of interval temporal logics and duration calculi. *Journal of Applied Non-Classical Logics*, 14(1-2):9–54, 2004.

[84] Ulle Endriss and Dov M Gabbay. Halfway between points and intervals: A temporal logic based on ordered trees. In *ESSLLI Workshop on Interval Temporal Logics and Duration Calculi*, pages 100–109, 2003.

[85] Partially ordered set, http://en.wikipedia.org/wiki/partially_ordered_set, (accessed October 20, 2014).

[86] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[87] Angelo Montanari. A guided tour through interval temporal logics lecture 2: Interval structures, relations, and logics. http://users.dimi.uniud.it/ angelo.montanari/aila2012lezione2.pdf, 2012 (accessed April 10, 2013).

[88] Davide Bresolin. *Proof methods for Interval Temporal Logics*. PhD thesis, Dipartimento di Matematica e Informatica, Università degli Studi di Udine, 2007. Forum Editrice, PhD Thesis Series CS 2007.

[89] Yde Venema. A modal logic for chopping intervals. *Journal of Logic and Computation*, 1(4):453–476, 1991.

[90] Davide Bresolin, Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Decidable and undecidable fragments of halpern and shoham's interval temporal logic: towards a complete classification. In Iliano Cervesato and Andrei Veith, Helmut anf Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 5330, pages 590–604. Springer, Springer Berlin / Heidelberg, 2008.

[91] Yde Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.

[92] Benjamin Charles Moszkowski. *Reasoning about digital circuits.* PhD thesis, Stanford University, 1983.

[93] Pierfrancesco Bellini, Giacomo Bucci, and Paolo Nesi. *Interval Temporal Logic for Real-Time Systems: Specification, Execution and Verification Processes*. PhD thesis, Univ. of Florence, Italy, 2001.

[94] Howard Bowman and Simon Thompson. A tableau method for interval temporal logic with projection. *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 108–123, 1998.

[95] Andrei Voronkov. Linear temporal logic LTL. Logic and Modeling, University of Manchester, www.voronkov.com/lics_doc.cgi?what=chapter&n=14, 2013 (accessed November 29, 2013).

[96] Davide Bresolin, Angelo Montanari, Pietro Sala, and Guido Sciavicco. What's decidable about halpern and shoham's interval logic? the maximal fragment abbl. In *26th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 387–396. IEEE, 2011.

[97] Jerzy Marcinkowski and Jakub Michaliszyn. The last paper on the halpern-shoham interval temporal logic. *arXiv preprint arXiv:1010.4529*, 2010.

[98] Joseph Y Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM (JACM)*, 38(4):935–962, 1991.

[99] Davide Bresolin, Dario Della Monica, Angelo Montanari, Pietro Sala, and Guido Sciavicco. Interval temporal logics over strongly discrete linear orders: the complete picture. *arXiv preprint arXiv:1210.2479*, 2012.

[100] Angelo Montanari. A guided tour through interval temporal logics lecture 4: Interval logics: undecidability. http://users.dimi.uniud.it/ angelo.montanari/aila2012lezione4.pdf, 2012 (accessed April 10, 2013).

[101] Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Propositional interval neighborhood temporal logics. *Journal of Universal Computer Science*, 9:1137–1167, 2003.

[102] Davide Bresolin, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Propositional interval neighborhood logics: Expressiveness, decidability, and undecidable extensions. *Annals of Pure and Applied Logic*, 161(3):289–304, 2009.

[103] Guido Sciavicco. Temporal reasoning in propositional neighborhood logic. In *Proc. of the 2nd Int. Conference on Language and Technology*, pages 390–395, 2005.

[104] Angelo Montanari and Guido Sciavicco. A decidable logic for time intervals: Propositional neighborhood logic. In *Proc. of the AAAI-2002 Workshop on Spatial and Temporal Reasoning*, pages 27–34, 2002.

[105] Valentin Goranko, Angelo Montanari, Pietro Sala, and Guido Sciavicco. A general tableau method for propositional interval temporal logics: Theory and implementation. *Journal of Applied Logic*, 4(3):305–330, 2006.

[106] Davide Bresolin, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Right propositional neighborhood logic over natural numbers with integer constraints for interval lengths. In *Software Engineering and Formal Methods, 2009 Seventh IEEE International Conference on*, pages 240–249. IEEE, 2009.

[107] Davide Bresolin, Dario Della Monica, Valentin Goranko, Angelo Montanari, and Guido Sciavicco. Metric propositional neighbourhood logics on natural numbers. *Software and Systems Modeling*, 12(2):245–264, 2013.

[108] Alessandro Artale and Enrico Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research*, 9:463–506, 1998.

[109] Morteza Yousef Sanati, Wendy MacCaull, and Thomas SE Maibaum. Analyzing clinical practice guidelines using a decidable metric interval-based temporal logic. In *FM 2014: Formal Methods*, pages 611–626. Springer, 2014.

[110] Adolfo Gustavo Serra Seca Neto and Marcelo Finger. Effective prover for minimal inconsistency logic. In Max Bramer, editor, *Artificial Intelligence in Theory and Practice*, volume 217 of *IFIP International Federation for Information Processing*, pages 465–474. Springer US, 2006.

[111] Holger Sturm and Frank Wolter. A tableau calculus for temporal description logic: the expanding domain case. *Journal of Logic and Computation*, 12(5):809–838, 2002.

[112] Carsten Lutz and Ulrike Sattler. Esslli 2002 course on description logics. http://www.informatik.uni-bremen.de/člu/esslli.html, 2002 (accessed June 19, 2014).

[113] Jia Tao, Giora Slutzki, and Vasant Honavar. Pspace tableau algorithms for acyclic modalized $\mathcal{ALC}$. *Journal of Automated Reasoning*, 49(4):551–582, 2012.

[114] Jie Zhang. Description logics and time. https://cs.uwaterloo.ca/ david/cs848/presentations-jiezhang-slides.pdf, 2006 (accessed July 24, 2012).

[115] Sang-Kyun Kim, Mi-Young Song, Chul Kim, Sang-Jun Yea, Hyun Chul Jang, and Kyu-Chul Lee. Temporal ontology language for representing and reasoning interval-based temporal knowledge. In *The Semantic Web*, pages 31–45. Springer, 2008.

[116] Alessandro Artale and Enrico Franconi. A computational account for a description logic of time and action. In Jon Doyle, Eric Sandewall, and Pietro Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 3–14. Morgan Kaufmann, 1994.

[117] Natalya Keberle. Temporal classes and owl. In *Proceedings of OWLED*, volume 9, 2009.

[118] Claudio Bettini. A family of temporal terminological logics. *Advances in Artificial Intelligence*, pages 120–131, 1993.

[119] Claudio Bettini. Time-dependent concepts: representation and reasoning using temporal description logics. *Data & Knowledge Engineering*, 22(1):1–38, 1997.

[120] Albrecht Schmiedel. A temporal terminological logic. In *Proceedings of the eighth National conference on Artificial intelligence (AAAI'90)*, volume 1, pages 640–645. AAAI Press, 1990.

[121] Wei Liu, Wenjie Xu, Dong Wang, Zongtian Liu, and Xujie Zhang. A temporal description logic for reasoning about action in event. *Information Technology Journal*, 11:1211–1218, 2012.

[122] Ian Horrocks and Peter Patel-Schneider. Reducing owl entailment to

description logic satisfiability. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):345–357, 2004.

[123] Gerald Mandell, Raphael Dolin, John Bennett, Gerald L Mandell, JE Bennett, et al. *Mandell, Douglas, and Bennett's principles and practice of infectious diseases*. Churchill Livingstone Philadelphia, 2005.

[124] The U.S. Department of Health & Human Services. Hiv symptoms, http://womenshealth.gov/hiv-aids/what-is-hiv-aids/hiv-symptoms.html, July 2011 (accessed December 4, 2013).

[125] The U.S. Department of Health & Human Services. Stages of hiv, http://www.aids.gov/hiv-aids-basics/just-diagnosed-with-hiv-aids/hiv-in-your-body/stages-of-hiv/, June 2009 (accessed December 4, 2013).

[126] Centers for Disease Control and Prevention (CDC). Appendix a mmwr recommendation and reports - aids-defining conditions. *Morbidity and Mortality Weekly Report*, 57, 2008.

[127] Kaletra faq, http://www.kaletra.com/information/faq.aspx, 2013 (accessed December 5, 2013).

[128] Protease inhibitor (pharmacology), http://en.wikipedia.org/wiki/protease_inhibitor_(pharmacology), Feb 2014 (accessed February 07, 2014).

[129] Mahshid Nasehi and Laila Mirhaghani. *A country guideline for fighting Tuberculosis (In persian)*. Andishmand, 2010 (1389).

[130] Centers for Disease Control and Prevention (CDC). Tuberculosis (tb), 2012 (accessed December 08, 2013).

[131] Centers for Disease Control and Prevention (CDC). Updated guidelines for the use of rifabutin or rifampin for the treatment and prevention of tuberculosis among hiv-infected patients taking protease inhibitors or nonnucleoside reverse transcriptase inhibitors. *Morbidity and Mortality Weekly Report*, 49, 2000.

[132] Centers for Disease Control and Prevention (CDC). Prevention and treatment of tuberculosis among patients infected with human immunodeficiency virus: Principles of therapy and revised recommendations. *Morbidity and Mortality Weekly Report*, 47, 1998.

[133] Mor Peleg, Samson Tu, Jonathan Bury, Paolo Ciccarese, John Fox, Robert A Greenes, Richard Hall, Peter D Johnson, Neill Jones, Anand Kumar, Silvia Miksch, Silvana Quaglini, Anreas Seyfang, Edward H Shortliffe, and Mario Stefanelli. Comparing models of decision and action for guideline-based decision support: a case-study approach. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.6260 &rep=rep1&type=pdf, 2002 (accessed July 16, 2014).

[134] Mor Peleg, Samson Tu, Jonathan Bury, Paolo Ciccarese, John Fox, Robert A Greenes, Richard Hall, Peter D Johnson, Neill Jones, Anand Kumar, Silvia Miksch, Silvana Quaglini, Anreas Seyfang, Edward H Shortliffe, and Mario Stefanelli. Comparing computer-interpretable guideline models: a case-study approach. *Journal of the American Medical Informatics Association*, 10(1):52–68, 2003.

[135] Arjen Hommersom, Perry Groot, Peter JF Lucas, Michael Balser, and Jonathan Schmitt. Verification of medical guidelines using background

knowledge in task networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):832–846, 2007.

[136] Stefan Christov, Bin Chen, George S Avrunin, Lori A Clarke, Leon J Osterweil, David Brown, Lucinda Cassells, and Wilson Mertens. Formally defining medical processes. *Methods of Information in Medicine*, 47(5):392–398, 2008.

[137] Mark A Musen, Samson W Tu, Amar K Das, and Yuval Shahar. Eon: A component-based approach to automation of protocol-directed therapy. *Journal of the American Medical Informatics Association*, 3(6):367–388, 1996.

[138] Open Clinical Knowledge Management for Medical Care. Eon. http://www.openclinical.org/gmm_eon.html, March 2004 (accessed February 13, 2014).

[139] Mor Peleg. Computer-interpretable clinical guidelines: A methodological review. *Journal of biomedical informatics*, 46(4):744–763, 2013.

[140] Open Clinical Knowledge Management for Medical Care. Glif. http://www.openclinical.org/gmm_glif.html, March 2006 (accessed February 13, 2014).

[141] Open Clinical Knowledge Management for Medical Care. Proforma. http://www.openclinical.org/gmm_proforma.html, March 2006 (accessed February 13, 2014).

[142] Jonathan Schmitt, Alwin Hoffmann, Michael Balser, Wolfgang Reif, and Marcos Mar. Interactive verification of medical guidelines. In Jayadev

Misra, Tobias Nipkow, and Emil Sekerinski, editors, *FM 2006: Formal Methods*, pages 32–47. Springer Berlin / Heidelberg, 2006.

[143] Yuval Shahar, Silvia Miksch, and Peter Johnson. The asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artificial intelligence in medicine*, 14(1-2):29–51, 1998.

[144] Simon Bäumler, Michael Balser, Andriy Dunets, Wolfgang Reif, and Jonathan Schmitt. Verification of medical guidelines by model checking–a case study. In *Model Checking Software*, pages 219–233. Springer, 2006.

[145] M. Balser, C. Duelli, W. Reif, and J. Schmitt. Formal semantics of asbru–v2. 12. Technical report, Institut für Informatik, 2006.

[146] Open Clinical Knowledge Management for Medical Care. Prodigy. http://www.openclinical.org/gmm_prodigy.html, March 2005 (accessed February 13, 2014).

[147] Arjen Hommersom, Perry Groot, Michael Balser, and Peter Lucas. Formal methods for verification of clinical practice guidelines. *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, 139:63–80, 2008.

[148] Laura Giordano, Paolo Terenziani, Alessio Bottrighi, Stefania Montani, and Loredana Donzella. Model checking for clinical guidelines: an agent-based approach. In *AMIA Annual Symposium Proceedings*, volume 2006, page 289. American Medical Informatics Association, 2006.

[149] David R Sutton and John Fox. The syntax and semantics of the proforma guideline modeling language. *Journal of the American Medical Informatics Association*, 10(5):433–443, 2003.

[150] David Isern, David Sánchez, and Antonio Moreno. Hecase2: a multi-agent ontology-driven guideline enactment engine. In *Multi-Agent Systems and Applications V*, pages 322–324. Springer, 2007.

[151] Yuval Shahar, Ohad Young, Erez Shalom, Alon Mayaffit, Robert Moskovitch, Alon Hessing, and Maya Galperin. Degel: A hybrid, multiple-ontology framework for specification and retrieval of clinical guidelines. In *Artificial Intelligence in Medicine*, pages 122–131. Springer, 2003.

[152] Michael Balser, Wolfgang Reif, Gerhard Schellhorn, Kurt Stenzel, and Andreas Thums. Formal system development with kiv. In Tom Maibaum, editor, *Fundamental approaches to software engineering*, pages 363–366. Springer Berlin / Heidelberg, 2000.

[153] Georg Duftschmid and Silvia Miksch. Knowledge-based verification of clinical guidelines by detection of anomalies. *Artificial intelligence in medicine*, 22(1):23–41, 2001.

[154] Bioportal, https://bioportal.bioontology.org/ontologies, February 2014 (accessed February 7, 2014).

[155] Heart rate, http://en.wikipedia.org/wiki/heart_rate, November 2014 (accessed November 9, 2014).

Appendix A

Proof of Soundness Theorem (IMPNL)

Theorem (Soundness). If $\psi \in$ IMPNL and a tableau $\mathcal{T}$ for the annotated version of $\Diamond_r \psi$ is closed, then ψ is not satisfiable.

Proof. Let $\mathbb{C}$ be the interval structure in $\mathbf{n}$. $P(m)$ is the statement: if the following conditions hold:

1. $\mathbf{n}$ is a node;
2. the height of $\mathbf{n}$ is m;
3. every branch through $\mathbf{n}$ is closed;

then the set $S(\mathbf{n})$ of all labeled formulas in the nodes between $\mathbf{n}$ and the root is not satisfiable over $\mathbb{C}$. We will prove $P(m)$ is true for all $m \geq 0$ using strong induction. Note that based on Theorem 2 (on Page 65), if a formula is not satisfiable over $\mathbb{C}$, it is not satisfiable at all.

(Base case) If $m = 0$, then $\mathbf{n}$ is a leaf, and the unique branch B containing $\mathbf{n}$ is closed. Then, we have one of the following cases. Take any model M

$=\langle \mathbb{D}, \mathbb{I}(\mathbb{D}), V\rangle$ where $\mathbb{D}$ is an extension of $\mathbb{C}$.

1. $S(\mathbf{n})$ contains both the labeled formulas $(p_s,[c_{k_1}, c_{l_1}])$ and $(\neg p_r,[c_{k_2}, c_{l_2}])$ where $([c_{k_1}, c_{l_1}] \cap [c_{k_2}, c_{l_2}] \neq \emptyset)$.
   In this case, clearly M,$[c_{k_1}, c_{l_1}] \models p_s$ if and only if M,$[c_{k_1}, c_{l_1}] \nvDash \neg p_s$ but there is a subinterval of $[c_{k_1}, c_{l_1}]$ such that $M, [c_m, c_n] \vDash \neg p_w$ $(w = |c_n - c_m|)$; therefore M,$[c_{k_1}, c_{l_1}] \nvDash p_s$. With the same reasoning, it is easy to show that M,$[c_{k_2}, c_{l_2}] \nvDash \neg p_r$.

2. $S(\mathbf{n})$ contains the labeled formula $(p_s,[c_k, c_{k_0}])$ and $(c_{k_0}, c_k, s) \in L_{\mathbb{C}}$ and $|c_k - c_{k_0}| \neq s$.
   In this case, M,$[c_k, c_{k_0}] \models p_s$ if and only if $|c_k - c_{k_0}| = s$ (iff $(c_{k_0}, c_k, s) \in L_{\mathbb{C}}$) and $p_s \in V([c_k, c_{k_0}])$.

Based on the cases considered, we are not able to construct a model for the labeled formulas in set S($\mathbf{n}$). Hence, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

(Induction Case) Assume $P(m)$ holds for all m, $0 \leq m \leq t$. We want to prove $P(t+1)$ holds. Suppose the height of $\mathbf{n}$ is $t+1$, and C=$\{c_0,...,c_n\}$. There are two cases to consider; (1) when $\mathbf{n}$ is the direct successor which results after applying the $\mathcal{R}_\wedge$ rule on node $\mathbf{g}$ s.t. $\mathbf{g} = (\varphi_0 \wedge \varphi_1,[c_i, c_j], \mathbb{C})$, and (2) when an expansion rule is applied to $\mathbf{n}$ s.t. $\mathbf{n} = ((\psi,[c_i, c_j]), \mathbb{C})$ or an expansion rule is applied to some labeled formula $(\psi,[c_i, c_j]) \in S(\mathbf{n}) - \{\Phi(\mathbf{n})\}$ (i.e. the existing formula in $\mathbf{n}$) to extend the branch at $\mathbf{n}$. Now, we consider these cases in detail.

1. Suppose $\mathbf{n}'$ be the direct successor of $\mathbf{n}$ due to the application of the $\mathcal{R}_\wedge$ rule on $\mathbf{g}$. Because the height of $\mathbf{n}$ is $t+1$, the height of $\mathbf{n}'$ is t. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}'$ is closed. By the induction assumption, $S(\mathbf{n}')$ is not satisfiable over

$\mathbb{C}$. Since $S(\mathbf{n}') = \{(\varphi_0, [c_i, c_j]), (\varphi_1, [c_i, c_j])\} \cup S(\mathbf{g})$, three cases should be considered:

- If (φ_1,$[c_i, c_j]$) is unsatisfiable then ($\varphi_0 \wedge \varphi_1$,$[c_i, c_j]$) is unsatisfiable too. It follows that $S(\mathbf{g})$ is not satisfiable over $\mathbb{C}$. Since $\mathbf{n}$ is the direct successor of $\mathbf{g}$, $S(\mathbf{g}) \subset S(\mathbf{n})$. Hence, $S(\mathbf{n})$ is not satisfiable.
- If $(\varphi_0, [c_i, c_j])$ is unsatisfiable, then it immediately follows that $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.
- Clearly, if $S(\mathbf{g})$ is not satisfiable then $S(\mathbf{n})$ is not satisfiable.

2. Consider the possible cases for the expansion rule applied at $\mathbf{n}$:

- Let $\psi = \varphi_0 \wedge \varphi_1$. There exists two nodes $\mathbf{n}_0 \in B$ and $\mathbf{n}_1 \in B$ such that $\mathbf{n}_0$ = ((φ_0,$[c_i, c_j]$), $\mathbb{C}$), $\mathbf{n}_1$ = ((φ_1,$[c_i, c_j]$), $\mathbb{C}$), and, suppose wlog, $\mathbf{n}_0$ is the successor of $\mathbf{n}$, and $\mathbf{n}_1$ is the successor of $\mathbf{n}_0$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_1$ is closed. Also, the height of $\mathbf{n}_1$ is less than the height of $\mathbf{n}$ (the height of $\mathbf{n}_1$ is less than or equal to t); thus, by the induction assumption, S($\mathbf{n}_1$) is not satisfiable over $\mathbb{C}$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must, in particular, satisfy ($\varphi_0 \wedge \varphi_1$, $[c_i, c_j]$), and thus (φ_0, $[c_i, c_j]$) and (φ_1, $[c_i, c_j]$), it follows that $S(\mathbf{n})$, $S(\mathbf{n}_0)$, and $S(\mathbf{n}_1)$ are equi-satisfiable over $\mathbb{C}$. Therefore, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;
- Let $\psi = \varphi_0 \vee \varphi_1$. There are two successors $\mathbf{n}_0$ and $\mathbf{n}_1$ of $\mathbf{n}$ such that $\mathbf{n}_0$ = ((φ_0,$[c_i, c_j]$), $\mathbb{C}$), $\mathbf{n}_1$ = ((φ_1,$[c_i, c_j]$), $\mathbb{C}$), and the height of $\mathbf{n}_0$ and the height of $\mathbf{n}_1$ is less than $\mathbf{n}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ and every branch containing $\mathbf{n}_1$ is closed. By the induction assumption,

$S(\mathbf{n}_0)$ and $S(\mathbf{n}_1)$ are not satisfiable over $\mathbb{C}$. Since $S(\mathbf{n}_0) = S(\mathbf{n}) \cup \{(\varphi_0, [c_i, c_j])\}$ and $S(\mathbf{n}_0)$ is not satisfiable, it follows that $S(\mathbf{n})$ is unsatisfiable, or φ_0 is unsatisfiable. If $S(\mathbf{n})$ is unsatisfiable, we have proved $S(\mathbf{n})$ is not satisfiable; suppose $S(\mathbf{n})$ is satisfiable but φ_0 is unsatisfiable. Now, we should consider the other case, i.e., $S(\mathbf{n}_1)$=$S(\mathbf{n}) \cup \{(\varphi_1, [c_i, c_j])\}$. We have a similar reasoning here. Assume that in this case, $\{(\varphi_1, [c_i, c_j])\}$ is unsatisfiable; therefore based on the cases which we mentioned, φ_0 and φ_1 are unsatisfiable. Recall $\{(\varphi_0, [c_i, c_j]) \vee (\varphi_1, [c_i, c_j])\} \in S(\mathbf{n})$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must also satisfy (φ_0,$[c_i, c_j]$) or (φ_1,$[c_i, c_j]$), $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- Let $\psi = \Diamond_r^* \varphi$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}^-(\mathbb{D}), V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that M,$[c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathrm{M}, [c_i, c_j] \models \Diamond_r^* \varphi$, and hence, $\mathrm{M}, [c_j, d] \models \varphi$ where $c_j < d$; thus (φ,$[c_j, d]$) is satisfiable. Node $\mathbf{n}$ has $f = |FF(\varphi)|$ direct successors named $\mathbf{n}_1$, ... ,$\mathbf{n}_f$. Since every branch through $\mathbf{n}$ is closed, then every branch through $\mathbf{n}_1$ ($\mathbf{n}_2$, ... ,$\mathbf{n}_f$) is closed.

  Now, consider the following three cases. Note that $1 \leq m \leq f$.

  - $d \in C$ and $(c_j, d, |d - c_j|) \in L_{\mathbb{C}}$; then there is an direct successor of $\mathbf{n}$, named $\mathbf{n}_m$, s.t. $\mathbf{n}_m = ((\varphi, [c_j, c_m]), \mathbb{C}_m)$ and $c_m = d$ and $\mathbb{C}_m = \mathbb{C}$. Since the height of $\mathbf{n}_m$ is less than the height of $\mathbf{n}$ and every branch through $\mathbf{n}_m$ is closed, by the induction assumption, $S(\mathbf{n}_m) = S(\mathbf{n}) \cup \{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}$, which is a contradiction, because by the assumptions, $S(\mathbf{n})$ and $\{(\varphi,[c_j, c_m])\}$ are satisfiable. Hence $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$, or $\{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}$. If $S(\mathbf{n})$ is

not satisfiable, we have proved the claim. If $\{(\varphi,[c_j, c_m])\}$ is not satisfiable, it follows that $\{(\Diamond_r^*\varphi,[c_i, c_j])\}$ is not satisfiable. Since $\{(\Diamond_r^*\varphi,[c_i, c_j])\} \subset S(\mathbf{n})$, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

- $d \in C$ and $(c_j, d, |d-c_j|) \notin L_{\mathbb{C}}$; then there is an direct successor of $\mathbf{n}$, named $\mathbf{n}_m$, s.t. $\mathbf{n}_m = ((\varphi,\ [c_j, c_m]),\ \mathbb{C}_m)$ and $c_m = d$ and and $C_m = C$ and $L_{\mathbb{C}_m} = L_{\mathbb{C}} \cup (c_j, d, |d - c_j|)$. Since the height of $\mathbf{n}_m$ is less than the height of $\mathbf{n}$ and every branch through $\mathbf{n}_m$ is closed, by the induction assumption, $S(\mathbf{n}_m) = S(\mathbf{n}) \cup \{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}_m$, which is a contradiction; because by the assumptions $S(\mathbf{n})$ and $\{(\varphi,[c_j, c_m])\}$ are satisfiable. Hence $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$ or $\{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}_m$. If $S(\mathbf{n})$ is not satisfiable, we have proved the claim. If $\{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}_m$, it follows that $\{(\Diamond_r^*\varphi,[c_i, c_j])\}$ is not satisfiable over $\mathbb{C}$. Since $\{(\Diamond_r^*\varphi,[c_i, c_j])\} \subset S(\mathbf{n})$, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.
- $d \notin C$; then there is an direct successor of $\mathbf{n}$, named $\mathbf{n}_m$, s.t. $\mathbf{n}_m = ((\varphi,\ [c_j, c_m]),\ \mathbb{C}_m)$ and $c_m = d$ and $C_m = C \cup \{d\}$ and $L_{\mathbb{C}_m} = L_{\mathbb{C}} \cup (c_j, d, |d - c_j|)$. As in the previous case, we can show that $S(\mathbf{n})$ is not satisfiable.

- Let $\psi = \Diamond_r^+\varphi$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), V\rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathrm{M},[c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathrm{M}, [c_i, c_j] \models \Diamond_r^+\varphi$, and hence, $\mathrm{M}, [c_j, d] \models \varphi$ where $c_j < d$. Node $\mathbf{n}$ has $f = |FF(\varphi)| + 1$ direct successors named $\mathbf{n}_0$, ... ,$\mathbf{n}_f$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ ($\mathbf{n}_1$, ... ,$\mathbf{n}_f$) is closed. There are three cases similar to the cases discussed for $\psi = \Diamond_r^*\varphi$. We do not repeat them here.

- Let $\psi = \Diamond_r^- \varphi$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}(\mathbb{D})^-, V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathrm{M}, [c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathrm{M}, [c_i, c_j] \models \Diamond_r^- \varphi$, and hence, $\mathrm{M}, [c_j, d] \models \varphi$ where $c_j < d$. Node $\mathbf{n}$ has one direct successor named $\mathbf{n}_0$. We know $\mathbf{n}_0 = ((\varphi,\ [c_j, d]),\ \mathbb{C}_0)$ and $C_0 = C \cup \{c_j, d\}$ and $L_{\mathbb{C}_0} = L_{\mathbb{C}} \cup (c_j, d, |d - c_j|)$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ is closed. Also, the height of $\mathbf{n}_0$ is less than the height of $\mathbf{n}$; thus, by the induction assumption, $S(\mathbf{n}_0) = S(\mathbf{n}) \cup \{(\varphi, [c_j, d])\}$ is not satisfiable over $\mathbb{C}_0$. As above, we can show $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

- The cases $\Diamond_l^* \varphi$, $\Diamond_l^+ \varphi$, $\Diamond_l^- \varphi$ are analogous to $\Diamond_r^* \varphi$, $\Diamond_r^+ \varphi$, $\Diamond_r^- \varphi$, respectively. Just change (c_j, d) to (d, c_i), $(c_j, d, |d - c_j|)$ to $(d, c_i, |c_i - d|)$, (c_j, c_m) to (c_m, c_i) and $c_j < d$ to $d < c_i$ in the appropriate the proof.

□

Appendix B

Proof of Soundness Theorem (MITDL)

Theorem (Soundness). If $\psi \in$ MITDL and a tableau $\mathcal{T}$ for the annotated version of $\Diamond_r \psi$ is closed, then ψ is not satisfiable.

Proof. Let $\mathbb{C}$ be the interval structure in $\mathbf{n}$. $P(m)$ is the statement: if the following conditions hold:

1. $\mathbf{n}$ is a node;
2. the height of $\mathbf{n}$ is m;
3. every branch through $\mathbf{n}$ is closed;

then set $S(\mathbf{n})$ of all labeled formulas in the nodes between $\mathbf{n}$ and the root is not satisfiable over $\mathbb{C}$. We will prove $P(m)$ is true for all $m \geq 0$ using strong induction. Note that based on Theorem 2 of the Theorem 2 (on Page 63), if a formula is not satisfiable over $\mathbb{C}$, it is not satisfiable at all.

(Base case) If $m = 0$, then $\mathbf{n}$ is a leaf, and the unique branch B containing $\mathbf{n}$ is closed. Take any model M $=\langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathbb{S} \rangle$ where $\mathbb{D}$ is an extension of $\mathbb{C}$.

Then, we have one of the following cases.

1. $S(\mathbf{n})$ contains the labeled formula $(p_s,[c_k, c_{k_0}])$ and $(c_{k_0}, c_k, s) \in L_{\mathbb{C}}$ and $|c_k - c_{k_0}| \neq s$.
   In this case, M,$[c_k, c_{k_0}] \models p_s$ if and only if $|c_k - c_{k_0}| = s$ (iff $(c_{k_0}, c_k, s) \in L_{\mathbb{C}}$) and $p_s \in V([c_k, c_{k_0}])$.

2. $S(\mathbf{n})$ contains both the labeled formulas $((a : C)_s,[c_{k_1}, c_{l_1}])$ and $((a : \neg C)_r,[c_{k_2}, c_{l_2}])$ where $([c_{k_1}, c_{l_1}] \cap [c_{k_2}, c_{l_2}] \neq \emptyset)$.
   In this case, clearly, M,$[c_{k_1}, c_{l_1}] \models (a : C)_s$ if and only if M,$[c_{k_1}, c_{l_1}] \not\models (a : \neg C)_r$ but there is a subinterval of $[c_{k_1}, c_{l_1}]$ such that $M, [c_m, c_n] \vDash (a : \neg C)_w$ $(w = |c_n - c_m|)$; therefore M,$[c_{k_1}, c_{l_1}] \not\models (a : C)_s$. With the same reasoning, it is easy to show that M,$[c_{k_2}, c_{l_2}] \not\models (a : \neg C)_r$.

Based on the cases, we are not able to construct a model for the labeled formulas in set S($\mathbf{n}$). Hence, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

(Induction Case) Assume $P(m)$ holds for all m, $0 \leq m \leq t$. We want to prove $P(t+1)$ holds. Suppose the height of $\mathbf{n}$ is $t+1$ and C=$\{c_0,...,c_n\}$. There are two cases to consider; (1) when $\mathbf{n}$ is the direct successor which results after applying the $\mathcal{R}_\theta$ $(\theta \in \{\wedge, \sqcap, \exists\})$ rule on node $\mathbf{g}$ s.t. $\mathbf{g} = (\psi,[c_i, c_j], \mathbb{C})$, and (2) when an expansion rule is applied to $\mathbf{n}$ s.t. $\mathbf{n} = ((\psi,[c_i, c_j]), \mathbb{C})$ or an expansion rule is applied to some labeled formula $(\psi,[c_i, c_j]) \in S(\mathbf{n}) - \{\Phi(\mathbf{n})\}$ (i.e. the existing formula in $\mathbf{n}$) to extend the branch at $\mathbf{n}$. Now, we consider these cases in detail.

1. Suppose $\mathbf{n}'$ be the direct successor of $\mathbf{n}$ due to the application of the $\mathcal{R}_\theta$ $(\theta \in \{\wedge, \sqcap, \exists\})$ rule on $\mathbf{g}$. Because the height of $\mathbf{n}$ is $t+1$, the height of $\mathbf{n}'$ is t. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}'$ is closed. By the induction assumption, $S(\mathbf{n}')$ is not satisfiable over $\mathbb{C}$. Consider the following cases.

- If $\mathcal{R}_{\wedge}$ is applied on $\mathbf{g}$ ($\psi = \varphi_0 \wedge \varphi_1$), since $S(\mathbf{n}^{'}) = \{(\varphi_0, [c_i, c_j]), (\varphi_1, [c_i, c_j])\} \cup S(\mathbf{g})$, three cases should be considered:
  - If (φ_1,$[c_i, c_j]$) is unsatisfiable, then ($\varphi_0 \wedge \varphi_1$,$[c_i, c_j]$) is unsatisfiable too. It follows that $S(\mathbf{g})$ is not satisfiable over $\mathbb{C}$. Since $\mathbf{n}$ is the direct successor of $\mathbf{g}$, $S(\mathbf{g}) \subset S(\mathbf{n})$. Hence, $S(\mathbf{n})$ is not satisfiable.
  - If $(\varphi_0, [c_i, c_j])$ is unsatisfiable, then based on the induction assumption, it follows that $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.
  - Clearly, if $S(\mathbf{g})$ is not satisfiable then $S(\mathbf{n})$ is not satisfiable.
- If $\mathcal{R}_{\sqcap}$ is applied on $\mathbf{g}$ ($\psi = (a : D \sqcap E)_k$), since $S(\mathbf{n}^{'}) = \{((a : D)_k, [c_i, c_j]), ((a : E)_k, [c_i, c_j])\} \cup S(\mathbf{g})$, three cases should be considered: Let $\mathbf{n} = ((a : D)_k, [c_i, c_j])$.
  - If ($(a : E)_k$,$[c_i, c_j]$) is unsatisfiable then ($(a : D \sqcap E)_k$,$[c_i, c_j]$) is unsatisfiable too. It follows that $S(\mathbf{g})$ is not satisfiable over $\mathbb{C}$. Since $\mathbf{n}$ is the direct successor of $\mathbf{g}$, $S(\mathbf{g}) \subset S(\mathbf{n})$. Hence, $S(\mathbf{n})$ is not satisfiable.
  - If $((a : D)_k, [c_i, c_j])$ is unsatisfiable, then based on the induction assumption, it follows that $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.
  - Clearly, if $S(\mathbf{g})$ is not satisfiable then $S(\mathbf{n})$ is not satisfiable.
- If $\mathcal{R}_{\exists}$ is applied on $\mathbf{g}$ ($\psi = (a : \exists R.D)_k$), and wlog adds two nodes $\mathbf{n} = ((R(a, b)_k$,$[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$ and $\mathbf{n}^{'} = (((b : D)_k$,$[c_i, c_j])$, $\mathbb{C}_{\mathcal{B}})$ to the branch, since $S(\mathbf{n}^{'}) = \{R(a, b)_k, [c_i, c_j]), ((b : D)_k, [c_i, c_j])\} \cup S(\mathbf{g})$, and it is not possible for ($R(a, b)_k$,$[c_i, c_j]$) or $((b : D)_k, [c_i, c_j])$ be unsatisfiable (because b is a new individual name), it follows that $S(\mathbf{g})$ is unsatisfiable. Because $S(\mathbf{g}) \subset S(\mathbf{n})$, it follows that $S(\mathbf{n})$ is unsatisfiable.

2. Consider the possible cases for the expansion rule applied at $\mathbf{n}$:

   - Let $\psi = \varphi_0 \wedge \varphi_1$. There exists two nodes $\mathbf{n}_0 \in B$ and $\mathbf{n}_1 \in B$ such that $\mathbf{n}_0 = ((\varphi_0,[c_i, c_j]),\ \mathbb{C})$, $\mathbf{n}_1 = ((\varphi_1,[c_i, c_j]),\ \mathbb{C})$, and, suppose wlog, $\mathbf{n}_0$ is the successor of $\mathbf{n}$ and $\mathbf{n}_1$ is the successor of $\mathbf{n}_0$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_1$ is closed. Also, the height of $\mathbf{n}_1$ is less than the height of $\mathbf{n}$ (the height of $\mathbf{n}_1$ is less than or equal to t); thus, by the induction assumption, $\mathrm{S}(\mathbf{n}_1)$ is not satisfiable over $\mathbb{C}$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must, in particular, satisfy $(\varphi_0 \wedge \varphi_1,\ [c_i, c_j])$, and thus $(\varphi_0,\ [c_i, c_j])$ and $(\varphi_1,\ [c_i, c_j])$, it follows that $S(\mathbf{n})$, $S(\mathbf{n}_0)$, and $S(\mathbf{n}_1)$ are equi-satisfiable over $\mathbb{C}$. Therefore, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

   - Let $\psi = \varphi_0 \vee \varphi_1$. There are two successors $\mathbf{n}_0$ and $\mathbf{n}_1$ of $\mathbf{n}$ such that $\mathbf{n}_0 = ((\varphi_0,[c_i, c_j]),\ \mathbb{C})$, $\mathbf{n}_1 = ((\varphi_1,[c_i, c_j]),\ \mathbb{C})$, and the height of $\mathbf{n}_0$ and the height of $\mathbf{n}_1$ is less than $\mathbf{n}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ and every branch containing $\mathbf{n}_1$ is closed. By the induction assumption, $S(\mathbf{n}_0)$ and $S(\mathbf{n}_1)$ are not satisfiable over $\mathbb{C}$. Since $S(\mathbf{n}_0) = S(\mathbf{n}) \cup \{(\varphi_0, [c_i, c_j])\}$ and $S(\mathbf{n}_0)$ is not satisfiable, it follows that $S(\mathbf{n})$ is unsatisfiable or φ_0 is unsatisfiable. If $S(\mathbf{n})$ is unsatisfiable, we have proved $S(\mathbf{n})$ is not satisfiable; suppose $S(\mathbf{n})$ is satisfiable but φ_0 is unsatisfiable. Now, we should consider the other case, i.e., $S(\mathbf{n}_1)$=$S(\mathbf{n}) \cup \{(\varphi_1, [c_i, c_j])\}$. We have a similar reasoning here. Assume that in this case, $\{(\varphi_1, [c_i, c_j])\}$ is unsatisfiable; therefore based on the cases which we mentioned, φ_0 and φ_1 are unsatisfiable. Recall $\{(\varphi_0, [c_i, c_j]) \vee (\varphi_1, [c_i, c_j])\} \in S(\mathbf{n})$. Since every model over

$\mathbb{C}$ satisfying $S(\mathbf{n})$ must also satisfy (φ_0,$[c_i, c_j]$) or (φ_1,$[c_i, c_j]$), $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- Let $\psi = \Diamond_r^*\varphi$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}^-(\mathbb{D}), V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that M,$[c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathrm{M}, [c_i, c_j] \models \Diamond_r^*\varphi$, and hence, $\mathrm{M}, [c_j, d] \models \varphi$ where $c_j < d$; thus (φ,$[c_j, d]$) is satisfiable. Node $\mathbf{n}$ has $f = |FF(\varphi)|$ direct successors named $\mathbf{n}_1$, ..., $\mathbf{n}_f$. Since every branch through $\mathbf{n}$ is closed, then every branch through $\mathbf{n}_1$ ($\mathbf{n}_2$, ... ,$\mathbf{n}_f$) is closed.

  Now, consider the following three cases. Note that $1 \leq m \leq f$.

  - $d \in C$ and $(c_j, d, |d - c_j|) \in L_{\mathbb{C}}$; then there is an direct successor of $\mathbf{n}$, named $\mathbf{n}_m$, s.t. $\mathbf{n}_m = ((\varphi, [c_j, c_m]), \mathbb{C}_m)$ and $c_m = d$ and $\mathbb{C}_m = \mathbb{C}$. Since the height of $\mathbf{n}_m$ is less than the height of $\mathbf{n}$, and every branch through $\mathbf{n}_m$ is closed, by the induction assumption, $S(\mathbf{n}_m) = S(\mathbf{n}) \cup \{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}$, which is a contradiction; because by the assumptions, $S(\mathbf{n})$ and $\{(\varphi,[c_j, c_m])\}$ are satisfiable. Hence $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$, or $\{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}$. If $S(\mathbf{n})$ is not satisfiable, we have proved the claim. If $\{(\varphi,[c_j, c_m])\}$ is not satisfiable, it follows that $\{(\Diamond_r^*\varphi,[c_i, c_j])\}$ is not satisfiable. Since $\{(\Diamond_r^*\varphi,[c_i, c_j])\} \subset S(\mathbf{n})$, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.
  - $d \in C$ and $(c_j, d, |d - c_j|) \notin L_{\mathbb{C}}$; then there is an direct successor of $\mathbf{n}$, named $\mathbf{n}_m$, s.t. $\mathbf{n}_m = ((\varphi, [c_j, c_m]), \mathbb{C}_m)$ and $c_m = d$ and and $C_m = C$ and $L_{\mathbb{C}_m} = L_{\mathbb{C}} \cup (c_j, d, |d - c_j|)$. Since the height of $\mathbf{n}_m$ is less than the height of $\mathbf{n}$, and every branch through $\mathbf{n}_m$ is closed, by the induction assumption, $S(\mathbf{n}_m) = S(\mathbf{n}) \cup$

$\{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}_m$, which is a contradiction; because by the assumptions $S(\mathbf{n})$ and $\{(\varphi,[c_j, c_m])\}$ are satisfiable. Hence $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$ or $\{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}_m$. If $S(\mathbf{n})$ is not satisfiable, we have proved the claim. If $\{(\varphi,[c_j, c_m])\}$ is not satisfiable over $\mathbb{C}_m$, it follows that $\{(\Diamond_r^*\varphi,[c_i, c_j])\}$ is not satisfiable over $\mathbb{C}$. Since $\{(\Diamond_r^*\varphi,[c_i, c_j])\} \subset S(\mathbf{n})$, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

- $d \notin C$; then there is an direct successor of $\mathbf{n}$, named $\mathbf{n}_m$, s.t. $\mathbf{n}_m = ((\varphi,\ [c_j, c_m]),\ \mathbb{C}_m)$ and $c_m = d$ and $C_m = C \cup \{d\}$ and $L_{\mathbb{C}_m} = L_{\mathbb{C}} \cup (c_j, d, |d - c_j|)$. As in the previous case, we can show that $S(\mathbf{n})$ is not satisfiable.

- Let $\psi = \Diamond_r^+\varphi$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), V\rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathrm{M},[c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathrm{M}, [c_i, c_j] \models \Diamond_r^+\varphi$, and hence, $\mathrm{M}, [c_j, d] \models \varphi$ where $c_j < d$. Node $\mathbf{n}$ has $f = |FF(\varphi)| + 1$ direct successors named $\mathbf{n}_0$, ... ,$\mathbf{n}_f$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ ($\mathbf{n}_1$, ... ,$\mathbf{n}_f$) is closed. There are three cases similar to the cases mentioned in $\psi = \Diamond_r^*\varphi$; we do not repeat them here.

- Let $\psi = \Diamond_r^-\varphi$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathrm{M} = \langle \mathbb{D}, \mathbb{I}^-(\mathbb{D}), V\rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathrm{M},[c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathrm{M}, [c_i, c_j] \models \Diamond_r^-\varphi$, and hence, $\mathrm{M}, [c_j, d] \models \varphi$ where $c_j < d$. Node $\mathbf{n}$ has one direct successor named $\mathbf{n}_0$. We know $\mathbf{n}_0 = ((\varphi,\ [c_j, d]),\ \mathbb{C}_0)$ and $C_0 = C \cup \{c_j, d\}$ and $L_{\mathbb{C}_0} = L_{\mathbb{C}} \cup (c_j, d, |d - c_j|)$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ is closed. Also, the height of $\mathbf{n}_0$ is less than the height of $\mathbf{n}$; thus,

by the induction assumption, $S(\mathbf{n}_0) = S(\mathbf{n}) \cup \{(\varphi,[c_j, d])\}$ is not satisfiable over $\mathbb{C}_0$. As above, we can show $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

- The cases $\Diamond_l^*\varphi$, $\Diamond_l^+\varphi$, $\Diamond_l^-\varphi$ are analogous to $\Diamond_r^*\varphi$, $\Diamond_r^+\varphi$, $\Diamond_r^-\varphi$, respectively. Just change (c_j, d) to (d, c_i), $(c_j, d, |d - c_j|)$ to $(d, c_i, |c_i - d|)$, (c_j, c_m) to (c_m, c_i) and $c_j < d$ to $d < c_i$.

- Let $\psi = (a : D \sqcap E)_k$. There exists two nodes $\mathbf{n}_0 \in B$ and $\mathbf{n}_1 \in B$ such that $\mathbf{n}_0 = (((a : D)_k,[c_i, c_j]),\mathbb{C})$, $\mathbf{n}_1 = (((a : E)_k,[c_i, c_j]),\mathbb{C})$, and, suppose wlog, $\mathbf{n}_0$ is the successor of $\mathbf{n}$, and $\mathbf{n}_1$ is the successor of $\mathbf{n}_0$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_1$ is closed. Also, the height of $\mathbf{n}_1$ is less than the height of $\mathbf{n}$ (the height of $\mathbf{n}_1$ is less than or equal to t); thus, by the induction assumption, $\mathrm{S}(\mathbf{n}_1)$ is not satisfiable over $\mathbb{C}$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must, in particular, satisfy $((a : D \sqcap E)_k, [c_i, c_j])$, and thus $((a : D)_k, [c_i, c_j])$ and $((a : E)_k, [c_i, c_j])$, it follows that $S(\mathbf{n})$, $S(\mathbf{n}_0)$, and $S(\mathbf{n}_1)$ are equi-satisfiable over $\mathbb{C}$. Therefore, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- Let $\psi = (a : D \sqcup E)_k$. There are two successors $\mathbf{n}_0$ and $\mathbf{n}_1$ of $\mathbf{n}$ such that $\mathbf{n}_0 = (((a : D)_k,[c_i, c_j]),\mathbb{C})$, $\mathbf{n}_1 = (((a : E)_k,[c_i, c_j]),\mathbb{C})$ and the height of $\mathbf{n}_0$, and the height of $\mathbf{n}_1$ is less than $\mathbf{n}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$, and every branch containing $\mathbf{n}_1$ is closed. By the induction assumption, $S(\mathbf{n}_0)$ and $S(\mathbf{n}_1)$ are not satisfiable over $\mathbb{C}$. Let $S(\mathbf{n}_0) = S(\mathbf{n}) \cup \{((a : D)_k, [c_i, c_j])\}$, and let $S(\mathbf{n}_1) = S(\mathbf{n}) \cup \{((a : E)_k, [c_i, c_j])\}$. Since $S(\mathbf{n}_0)$ is not satisfiable, it follows that $S(\mathbf{n})$ is unsatisfiable, or $((a : D)_k, [c_i, c_j])$ is unsatisfiable. If $S(\mathbf{n})$ is unsatisfiable, we are

done. Suppose $S(\mathbf{n})$ is satisfiable but $((a : D)_k, [c_i, c_j])$ is unsatisfiable. Now, we consider $S(\mathbf{n}_1)$. We have a similar reasoning here. Assume that in this case, $\{((a : E)_k, [c_i, c_j])\}$ is unsatisfiable; therefore based on the cases which we mentioned, $((a : D)_k, [c_i, c_j])$ and $((a : E)_k, [c_i, c_j])$ are unsatisfiable. Recall $\{((a : D)_k, [c_i, c_j]) \sqcup ((a : E)_k, [c_i, c_j])\} \in S(\mathbf{n})$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must also satisfy $((a : D)_k,[c_i, c_j])$ or $((a : E)_k,[c_i, c_j])$, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- Let $\psi = (a : \exists R.D)_k$. There exists two nodes $\mathbf{n}_0 \in B$ and $\mathbf{n}_1 \in B$ such that $\mathbf{n}_0 = (((b : D)_k,[c_i, c_j]),\mathbb{C})$, $\mathbf{n}_1 = ((R(a, b)_k,[c_i, c_j]),\mathbb{C})$, and, suppose wlog, $\mathbf{n}_0$ is the successor of $\mathbf{n}$, and $\mathbf{n}_1$ is the successor of $\mathbf{n}_0$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$, and every branch containing $\mathbf{n}_1$ is closed. Also, the height of $\mathbf{n}_0$ and the height of $\mathbf{n}_1$ is less than $\mathbf{n}$; thus, by the induction assumption, neither S($\mathbf{n}_0$) nor S($\mathbf{n}_1$) is satisfiable over $\mathbb{C}$. b is a new individual name when the rule is applied, both $(R(a, b)_k, [c_i, c_j])$ and $((b : D)_k, [c_i, c_j])$ are satisfiable. $S(\mathbf{n_1}) = \{(R(a, b)_k, [c_i, c_j]), ((b : D)_k, [c_i, c_j])\} \cup S(\mathbf{n})$ is not satisfiable, thus S($\mathbf{n}$) is not satisfiable;

- Let $\psi = (a : \forall R.D)_k$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model M $= \langle \mathbb{D}, \mathbb{I}^-(\mathbb{D}), V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that M,$[c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. Since $\{(a : \forall R.D)_k,[c_i, c_j]),(R(a, b)_{k'},[c_{i_0}, c_{j_0}])\} \in S(\mathbf{n})$ where $[c_{i_0}, c_{j_0}] \subseteq [c_i, c_j]$, follows that, M, $[c_i, c_j] \models (a : \forall R.D)_k$ and M, $[c_{i_0}, c_{j_0}] \models R(a, b)_{k'}$. It is easy to show that M, $[c_{i_1}, c_{j_1}] \models (b : D)_{k''}$ where $k'' = |c_{j_1} - c_{i_1}|$ and $[c_{i_1}, c_{j_1}] = [c_i, c_j] \cap [c_{i_0}, c_{j_0}]$; thus $((b : D)_{k''},[c_{i_1}, c_{j_1}])$ is satisfiable over $\mathbb{C}_{B_1}$. Node $\mathbf{n}$ has one direct successor named $\mathbf{n}_0$. Since every

branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ is closed. Also, the height of $\mathbf{n}_0$ is less than the height of $\mathbf{n}$; thus by the induction assumption, $S(\mathbf{n}^{'}) = \{((a : D)_{k''}, [c_{i_1}, c_{j_1}])\} \cup S(\mathbf{n})$ is unsatisfiable over $\mathbb{C}_{B_1}$. Since $((a : D)_{k''}, [c_{i_1}, c_{j_1}])$ is satisfiable, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}_B$;

- Let $\psi = (D = \top)_k$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model M $=\langle \mathbb{D}, \mathbb{I}^{-}(\mathbb{D}), V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that M,$[c_i, c_j] \models \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. Since $\{((D = \top)_k$,$[c_i, c_j])$,$((a : E)_{k'}$,$[c_{i_0}, c_{j_0}])\} \in S(\mathbf{n})$ where $[c_i, c_j] \cap [c_{i_0}, c_{j_0}] = [c_{i_1}, c_{j_1}] \neq \emptyset$, follows that, M, $[c_i, c_j] \models (D = \top)_k$ and M, $[c_{i_0}, c_{j_0}] \models (a : E)_{k'}$ $(k^{'} = |c_{j_0} - c_{i_0}|)$. It is easy to show that M, $[c_{i_1}, c_{j_1}] \models (a : D)_{k''}$ $(k^{''} = |c_{j_1} - c_{i_1}|)$; thus $((a : D)_{k''}$,$[c_{i_1}, c_{j_1}])$ is satisfiable over $\mathbb{C}_{B_1}$. Node $\mathbf{n}$ has one direct successor named $\mathbf{n}_0$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ is closed. Also, the height of $\mathbf{n}_0$ is less than the height of $\mathbf{n}$; thus, by the induction assumption, $S(\mathbf{n_0})$=$\{((a : D)_{k''}$,$[c_{i_1}, c_{j_1}])\} \cup S(\mathbf{n})$ is not satisfiable over $\mathbb{C}_{B_1}$. Since $((a : D)_{k''}, [c_{i_1}, c_{j_1}])$ is satisfiable, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$.

□

Appendix C

Tableaus in Detail

C.1 Tableau for ϑ_{HIV}

$\boxed{1}$[root] $((\Diamond_r^*(Elisa_{1d}^{\#1} \wedge \Diamond_r^*(Elisa_{1d}^{\#2} \wedge \dots)),[c_0,c_1]),\ \{\{c_0 < c_1\},\{(c_0,c_1,2h)\}\})$

$\boxed{1\text{-}\mathcal{R}_r^*}$

$\boxed{2}((Elisa_{1d}^{\#1} \wedge \Diamond_r^*(Elisa_{1d}^{\#2} \wedge \dots),[c_1,c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,1d)\}\})$

$\boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{3}((Elisa_{1d}^{\#1},[c_1,c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,1d)\}\})$

$\boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{4}((\Diamond_r^*(Elisa_{1d}^{\#2} \wedge \Diamond_r^*(WesternBlot_{1d} \wedge \dots)),[c_1,c_2]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,1d)\}\})$

$\boxed{4\text{-}\mathcal{R}_r^*}$

$\boxed{5}((Elisa_{1d}^{\#2} \wedge \Diamond_r^*(WesternBlot_{1d} \wedge \dots),[c_2,c_3]),$
$\{\{c_0 < c_1 < c_2 < c_3\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d)\}\})$

$\boxed{5\text{-}\mathcal{R}_\wedge}$

$\boxed{6}((Elisa_{1d}^{\#2},[c_2,c_3]),$
$\{\{c_0 < c_1 < c_2 < c_3\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d)\}\})$

$\boxed{5\text{-}\mathcal{R}_\wedge}$

$\boxed{7}((\Diamond_r^*(WesternBlot_{1d} \wedge \dots),[c_2,c_3]),$
$\{\{c_0 < c_1 < c_2 < c_3\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d)\}\})$

$\boxed{7\text{-}\mathcal{R}_r^*}$

$\boxed{8}((WesternBlot_{1d} \wedge \Diamond_r^*(Registration_{4d} \wedge \dots),[c_3,c_4]),$
$\{\{c_0 < c_1 < c_2 < c_3 < c_4\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d)\}\})$

$\boxed{8\text{-}\mathcal{R}_\wedge}$

$\boxed{9}((WesternBlot_{1d},[c_3,c_4]),$
$\{\{c_0 < c_1 < c_2 < c_3 < c_4\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d)\}\})$

$\boxed{8\text{-}\mathcal{R}_\wedge}$

$\boxed{10}((\Diamond_r^*(Registration_{4d} \wedge \dots),[c_3,c_4]),$
$\{\{c_0 < c_1 < c_2 < c_3 < c_4\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d)\}\})$

$\boxed{10\text{-}\mathcal{R}_r^*}$

$\boxed{11}((Registration_{4d} \wedge \Diamond_r^*(\top_{90d} \wedge CD4Check_{1d} \wedge \dots),[c_4,c_5]),$
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d),\ (c_4,c_5,4d)\}\})$

$\boxed{11\text{-}\mathcal{R}_\wedge}$

$\boxed{12}((Registration_{4d},[c_4,c_5]),$
$\{\{c_0 < c_1 < c_2 < c_3 < c_4 < c_5\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d),\ (c_4,c_5,4d)\}\})$

$\boxed{11\text{-}\mathcal{R}_\wedge}$

$\boxed{13}((\Diamond_r^*(\top_{90d} \wedge \Diamond_l^- \Diamond_r^*(CD4Check_{1d} \wedge \dots)),[c_4,c_5]),$
$\{\{c_0 < c_1 < c_2 < c_3 < c_4 < c_5\},\{(c_0,c_1,2h),\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d),\ (c_4,c_5,4d)\}\})$

$\boxed{13\text{-}\mathcal{R}_r^*}$

$\boxed{14}((\top_{90d} \wedge \Diamond_l^- \Diamond_r^*(CD4Check_{1d} \wedge \dots),[c_5,c_6]),$
$\{\{\cdots < c_2 < c_3 < c_4 < c_5 < c_6\},\{\dots,\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d),\ (c_4,c_5,4d),\ (c_5,c_6,90d)\}\})$

$\boxed{14\text{-}\mathcal{R}_\wedge}$

$\boxed{15}((\top_{90d},[c_5,c_6]),$
$\{\{\cdots < c_2 < c_3 < c_4 < c_5 < c_6\},\{\dots,\ (c_1,c_2,1d),(c_2,c_3,1d),\ (c_3,c_4,1d),\ (c_4,c_5,4d),\ (c_5,c_6,90d)\}\})$

Figure C.1: Tableau for ϑ_{HIV} (Modeled in IMPNL)

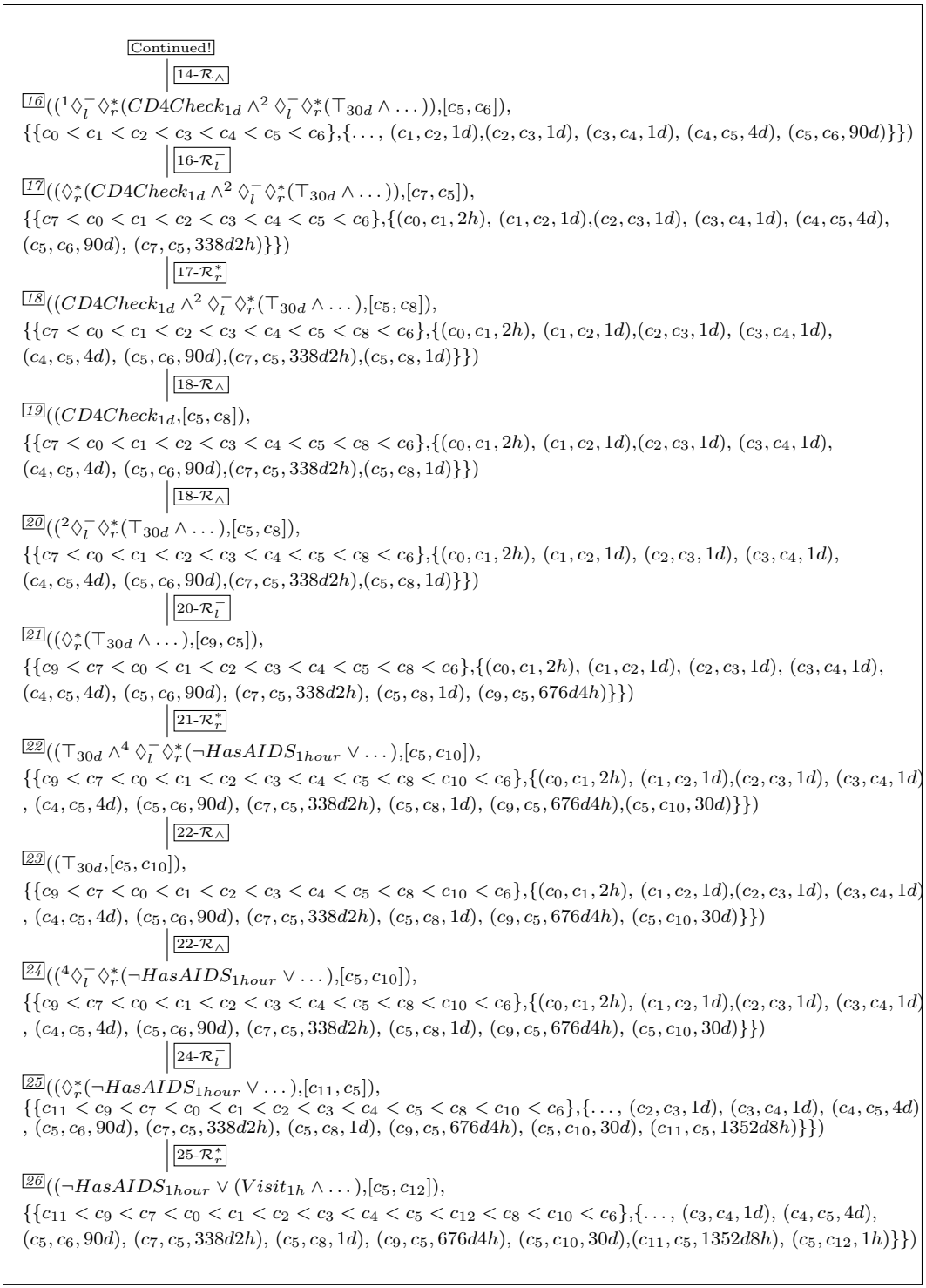

Figure C.2: Tableau for ϑ_{HIV} – Modeled in IMPNL (continued)

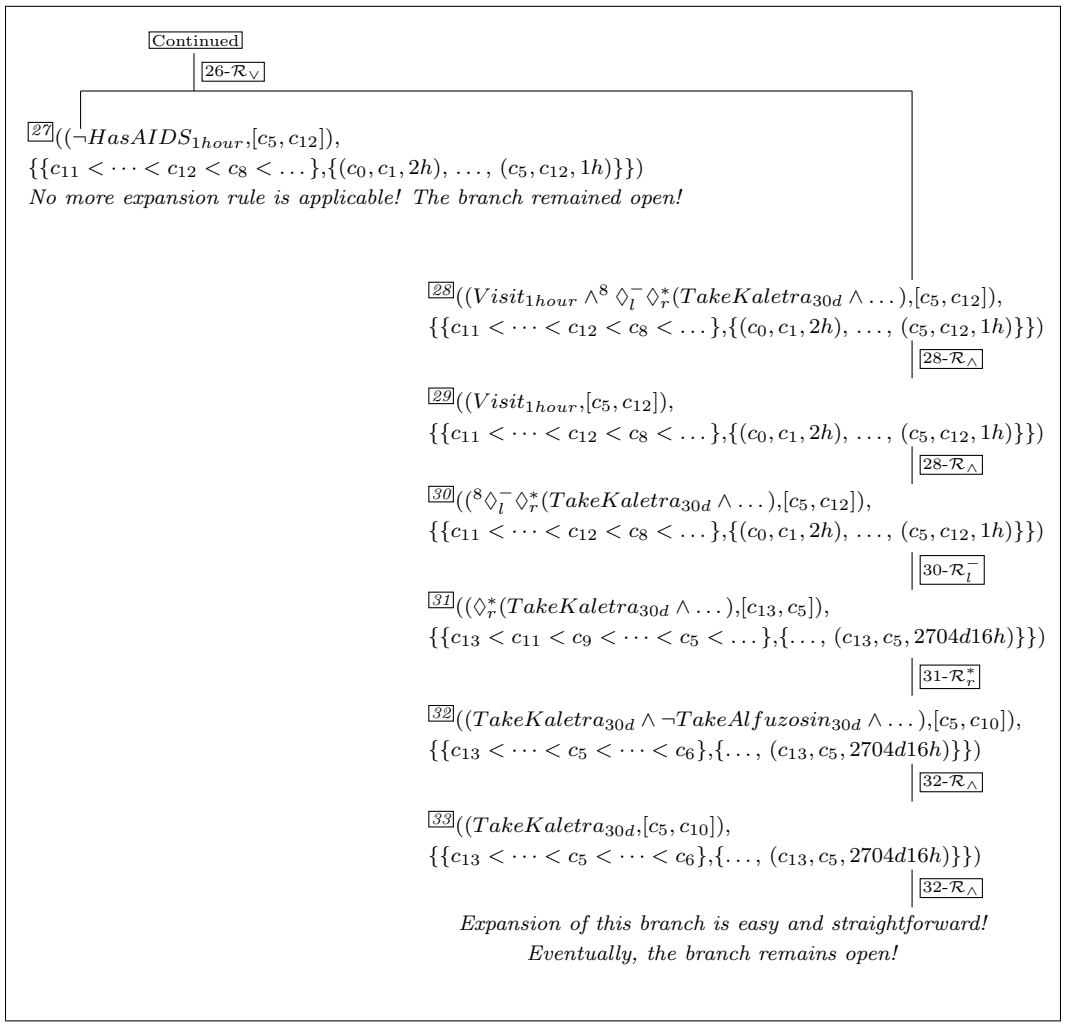

Figure C.3: Tableau for ϑ_{HIV} – Modeled in IMPNL (continued)

C.2 Tableau for ξ_{HIV}

$\boxed{1}[root]$ $((\xi_{HIV},[c_0,c_1])$, $\{\{c_0 < c_1\},\{(c_0,c_1,2h)\}\})$

$\boxed{1\text{-}\mathcal{R}_r^*}$

$\boxed{2}((((\neg Kaletra \sqcap \neg Norvir \sqcap \neg Aptivus) \sqcup ProteaseInhibitors = \top)_{120yr} \wedge \dots),[c_1,c_2])$,
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr)\}\})$

$\boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{3}((((\neg Kaletra \sqcap \neg Norvir \sqcap \neg Aptivus) \sqcup ProteaseInhibitors = \top)_{120yr},[c_1,c_2])$,
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr)\}\})$

$\boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{4}(((\forall TakeMedicine.\neg Rifampin \sqcup \forall TakeMedicine.\neg ProteaseInhibitors = \top)_{120yr} \wedge \dots,[c_1,c_2])$,
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr)\}\})$

$\boxed{4\text{-}\mathcal{R}_\wedge}$

$\boxed{5}(((\forall TakeMedicine.\neg Rifampin \sqcup \forall TakeMedicine.\neg ProteaseInhibitors = \top)_{120yr},[c_1,c_2])$,
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr)\}\})$

$\boxed{4\text{-}\mathcal{R}_\wedge}$

$\boxed{6}(((p : Patient)_{120yr} \wedge^1 \Diamond_l^- \Diamond_r^*((e1 : Elisa)_{1d} \wedge \dots),[c_1,c_2])$,
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr)\}\})$

$\boxed{6\text{-}\mathcal{R}_\wedge}$

$\boxed{7}(((p : Patient)_{120yr},[c_1,c_2])$,
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr)\}\})$

$\boxed{6\text{-}\mathcal{R}_\wedge}$

$\boxed{8}((({}^1\Diamond_l^- \Diamond_r^*((e1 : Elisa)_{1d} \wedge \dots),[c_1,c_2])$,
$\{\{c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr)\}\})$

$\boxed{8\text{-}\mathcal{R}_l^-}$

$\boxed{9}((\Diamond_r^*((e1 : Elisa)_{1d} \wedge \dots),[c_{-1},c_1])$,
$\{\{c_{-1} < c_0 < c_1 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr),(c_{-1},c_1,\kappa)\}\})$

$\boxed{9\text{-}\mathcal{R}_r^*}$

$\boxed{10}(((e1 : Elisa)_{1d} \wedge PerformTest(p,e1)_{1d} \wedge \dots,[c_1,c_3])$,
$\{\{c_{-1} < c_0 < c_1 < c_3 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr),(c_{-1},c_1,\kappa),\ (c_1,c_3,1d)\}\})$

$\boxed{10\text{-}\mathcal{R}_\wedge}$

$\boxed{11}(((e1 : Elisa)_{1d},[c_1,c_3])$,
$\{\{c_{-1} < c_0 < c_1 < c_3 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr),(c_{-1},c_1,\kappa),\ (c_1,c_3,1d)\}\})$

$\boxed{10\text{-}\mathcal{R}_\wedge}$

$\boxed{12}((PerformTest(p,e1)_{1d} \wedge \dots,[c_1,c_3])$,
$\{\{c_{-1} < c_0 < c_1 < c_3 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr),(c_{-1},c_1,\kappa),\ (c_1,c_3,1d)\}\})$

$\boxed{12\text{-}\mathcal{R}_\wedge}$

$\boxed{13}((PerformTest(p,e1)_{1d},[c_1,c_3])$,
$\{\{c_{-1} < c_0 < c_1 < c_3 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr),(c_{-1},c_1,\kappa),\ (c_1,c_3,1d)\}\})$

$\boxed{12\text{-}\mathcal{R}_\wedge}$

$\boxed{14}((\Diamond_r^*((e2 : Elisa)_{1d} \wedge \dots),[c_1,c_3])$,
$\{\{c_{-1} < c_0 < c_1 < c_3 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr),(c_{-1},c_1,\kappa),\ (c_1,c_3,1d)\}\})$

$\boxed{14\text{-}\mathcal{R}_r^*}$

$\boxed{15}(((e2 : Elisa)_{1d} \wedge \dots,[c_3,c_4])$,
$\{\{c_{-1} < c_0 < c_1 < c_3 < c_4 < c_2\},\{(c_0,c_1,2h),\ (c_1,c_2,120yr),(c_{-1},c_1,\kappa),\ (c_1,c_3,1d),(c_3,c_4,1d)\}\})$

Figure C.4: Tableau for ξ_{HIV} (Modeled in MITDL)

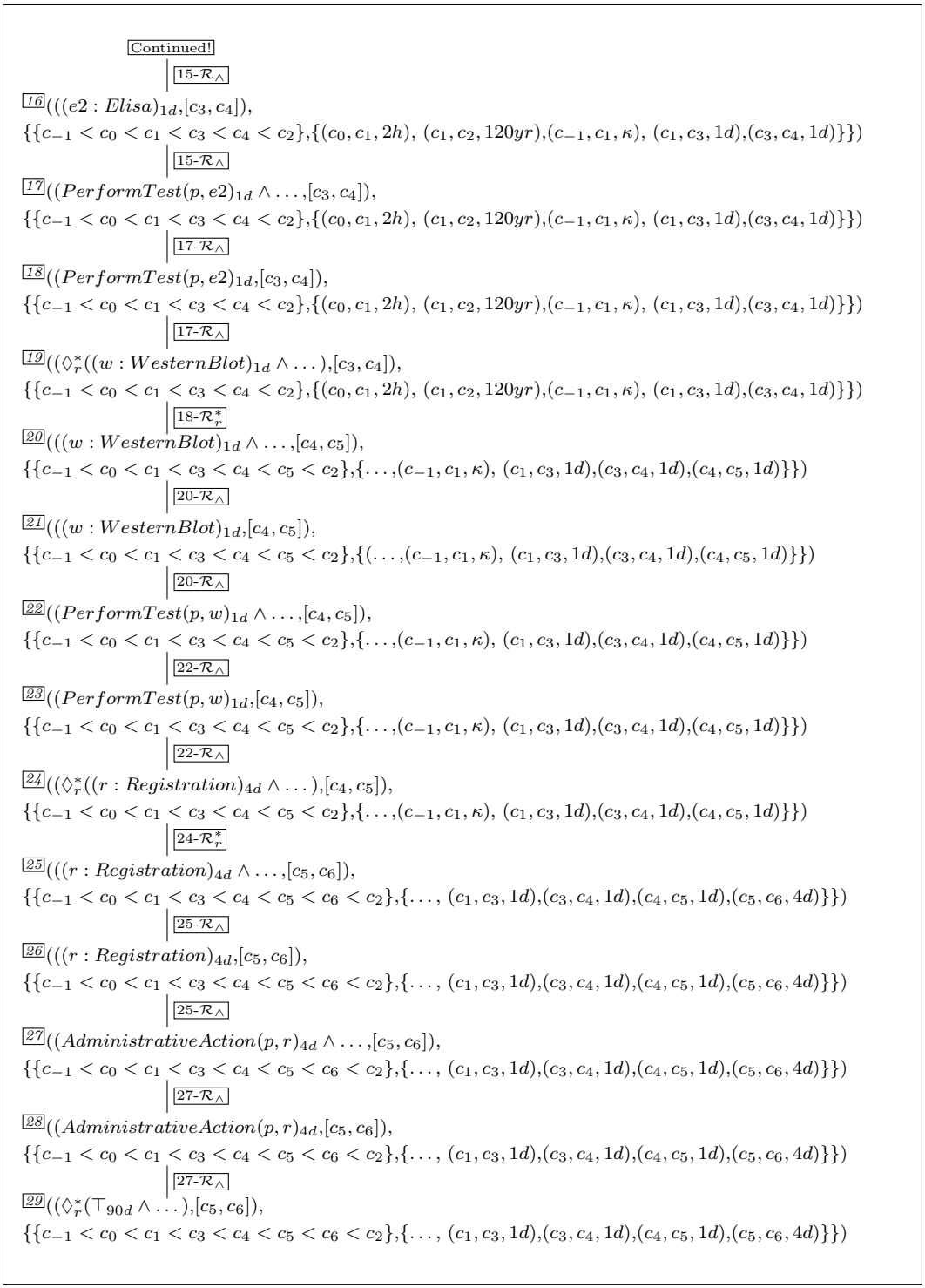

Figure C.5: Tableau for ξ_{HIV} – Modeled in MITDL (continued)

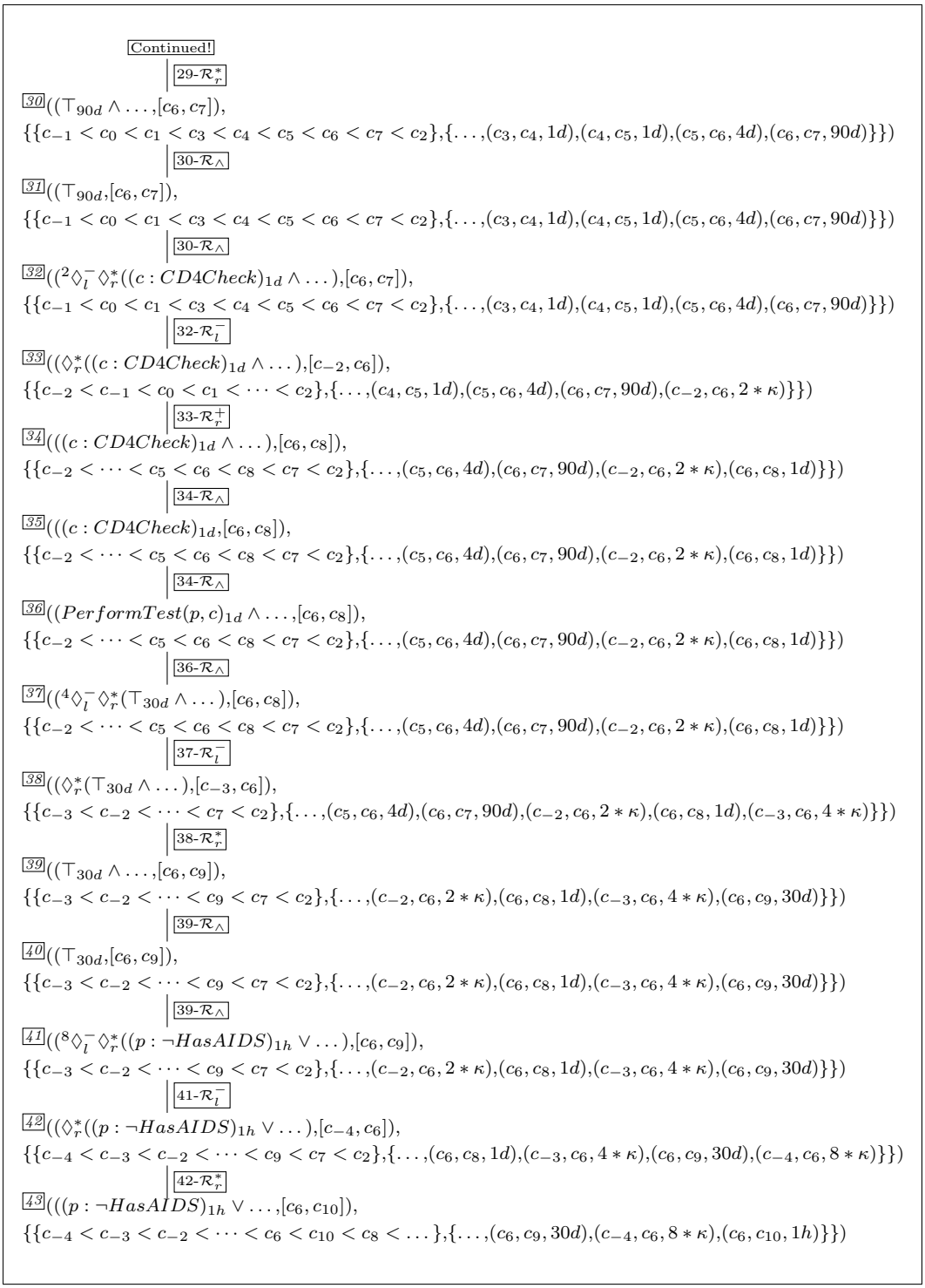

Figure C.6: Tableau for ξ_{HIV} – Modeled in MITDL (continued)

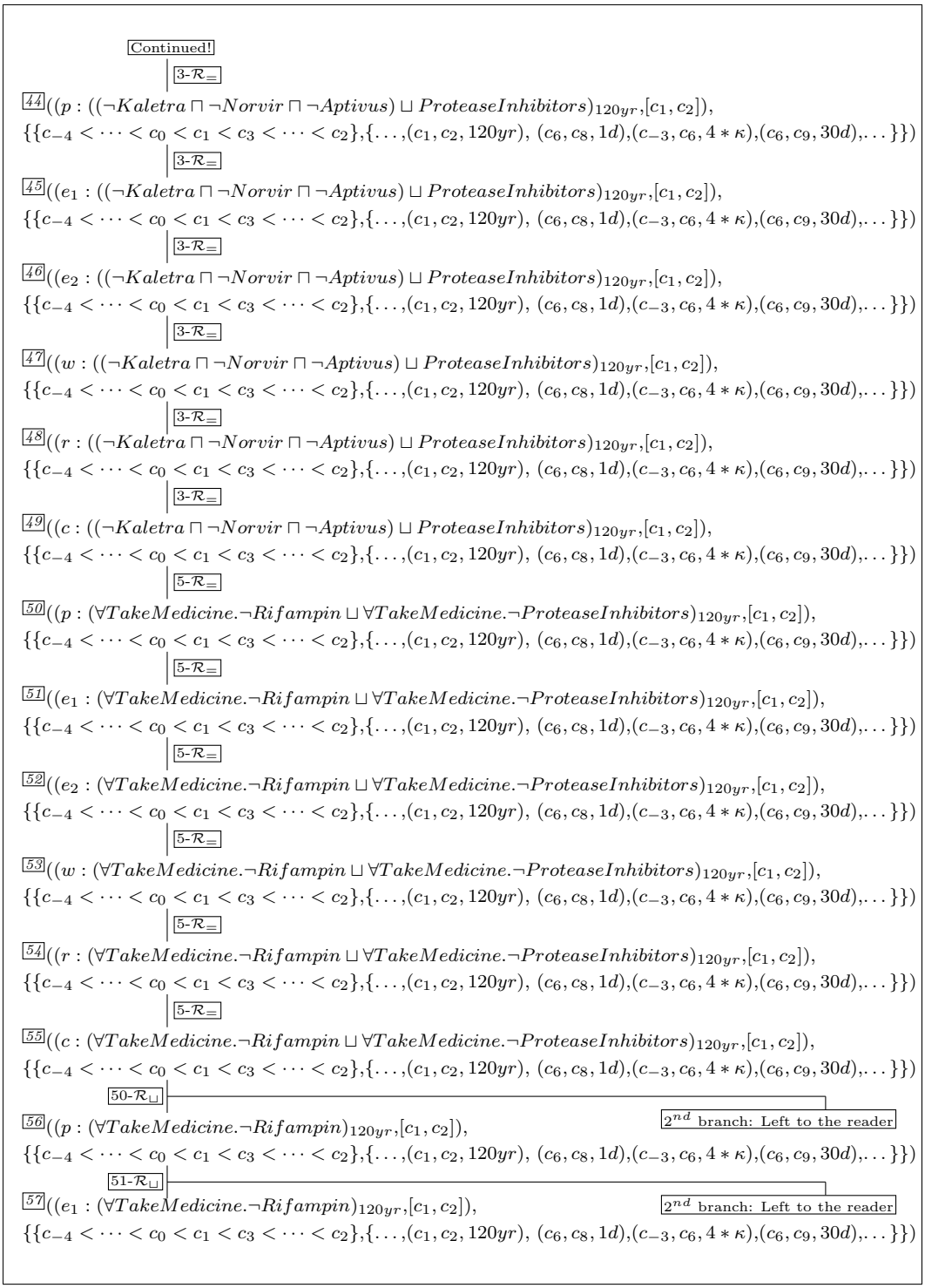

Figure C.7: Tableau for ξ_{HIV} – Modeled in MITDL (continued)

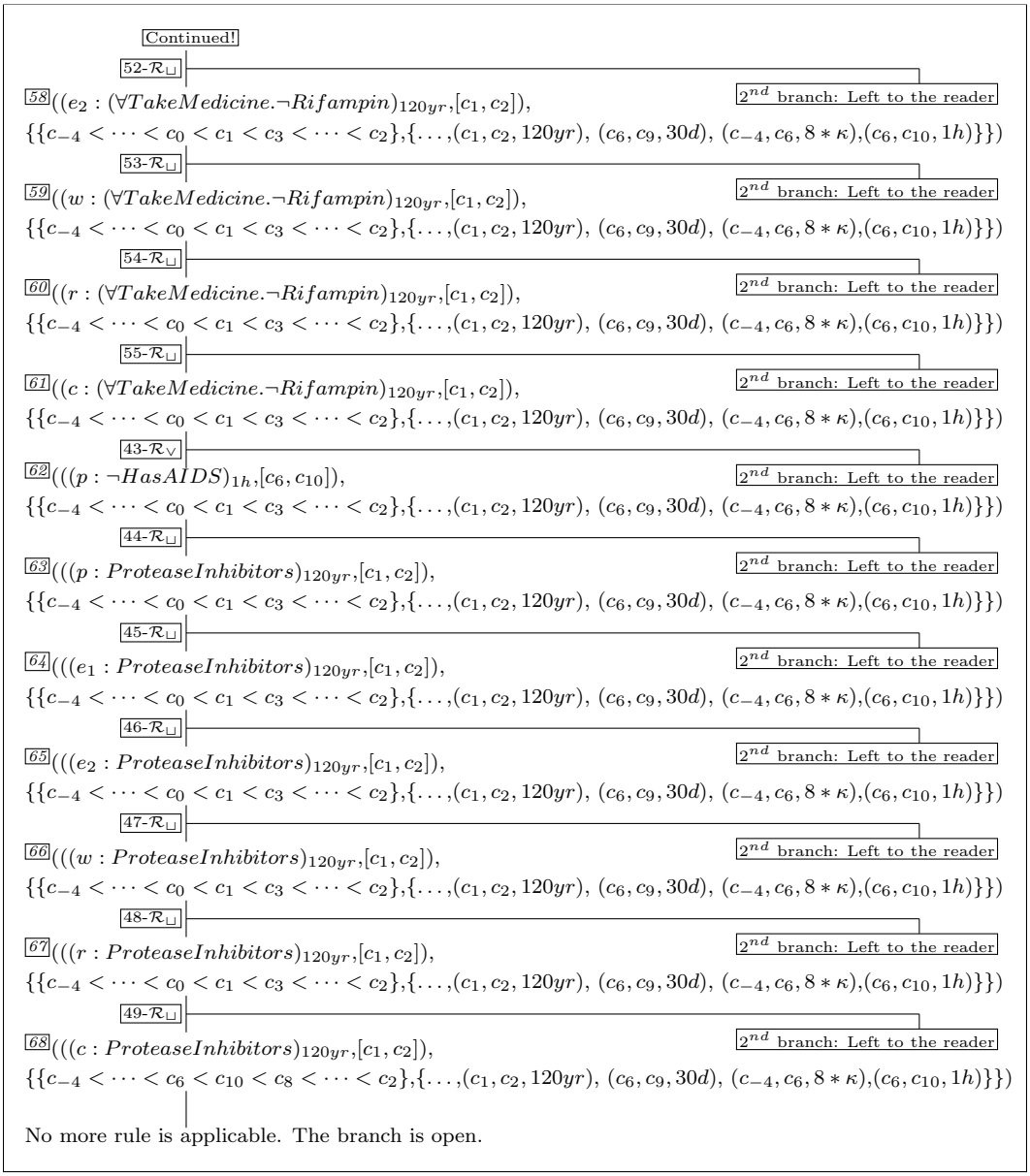

Figure C.8: Tableau for ξ_{HIV} – Modeled in MITDL (continued)

C.3 Tableau for ψ'_{HIV-TB}

$\boxed{1}[root]\ ((^1\Diamond_r^-(^2\Diamond_l^-\Diamond_r^*(^{HIV}Elisa_{1d}^{\#1}\wedge\dots)\wedge^{64}\Diamond_l^-\Diamond_r^*(^{TB}TakeRifampin_{120d}\wedge\dots)))),[d_0,d_1]),$
$\{\{d_0<d_1\},\{(d_0,d_1,2h)\}\}$)

$\boxed{1\text{-}\mathcal{R}_r^-}$

$\boxed{2}((^2\Diamond_l^-\Diamond_r^*(^{HIV}Elisa_{1d}^{\#1}\wedge\dots)\wedge^{64}\Diamond_l^-\Diamond_r^*(^{TB}TakeRifampin_{120d}\wedge\dots)),[d_1,d_{11}]),$
$\{\{d_0<d_1<d_{11}\},\{(d_0,d_1,2h),\ (d_1,d_{11},3y32d3h)\}\}$)

$\boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{3}((^2\Diamond_l^-\Diamond_r^*(^{HIV}Elisa_{1d}^{\#1}\wedge\dots)),[d_1,d_{11}]),$
$\{\{d_0<d_1<d_{11}\},\{(d_0,d_1,2h),\ (d_1,d_{11},3y32d3h)\}\}$)

$\boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{4}((^{64}\Diamond_l^-\Diamond_r^*(^{TB}TakeRifampin_{120d}\wedge\dots),[d_1,d_{11}]),$
$\{\{d_0<d_1<d_{11}\},\{(d_0,d_1,2h),\ (d_1,d_{11},3y32d3h)\}\}$)

$\boxed{3\text{-}\mathcal{R}_l^-}$

$\boxed{5}((\Diamond_r^*(^{HIV}Elisa_{1d}^{\#1}\wedge\dots),[d_{-1},d_1]),$
$\{\{d_{-1}<d_0<d_1<d_{11}\},\{(d_0,d_1,2h),\ (d_1,d_{11},3y32d3h),\ (d_{-1},d_1,6y64d6h)\}\}$)

$\boxed{5\text{-}\mathcal{R}_r^*}$

$\boxed{6}((^{HIV}Elisa_{1d}^{\#1}\wedge\Diamond_r^*(^{HIV}Elisa_{1d}^{\#2}\wedge\dots),[d_1,d_2]),$
$\{\{d_{-1}<d_0<d_1<d_2<d_{11}\},\{(d_0,d_1,2h),\ (d_1,d_{11},3y32d3h),\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d)\}\}$)

$\boxed{6\text{-}\mathcal{R}_\wedge}$

$\boxed{7}((^{HIV}Elisa_{1d}^{\#1},[d_1,d_2]),$
$\{\{d_{-1}<d_0<d_1<d_2<d_{11}\},\{(d_0,d_1,2h),\ (d_1,d_{11},3y32d3h),\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d)\}\}$)

$\boxed{6\text{-}\mathcal{R}_\wedge}$

$\boxed{8}((\Diamond_r^*(^{HIV}Elisa_{1d}^{\#2}\wedge\Diamond_r^*(^{HIV}WesternBlot_{1d}\wedge\dots)),[d_1,d_2]),$
$\{\{d_{-1}<d_0<d_1<d_2<d_{11}\},\{(d_0,d_1,2h),\ (d_1,d_{11},3y32d3h),\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d)\}\}$)

$\boxed{8\text{-}\mathcal{R}_r^*}$

$\boxed{9}((^{HIV}Elisa_{1d}^{\#2}\wedge\Diamond_r^*(^{HIV}WesternBlot_{1d}\wedge\dots),[d_2,d_3]),$
$\{\{d_{-1}<d_0<d_1<d_2<d_3<d_{11}\},\{\dots,\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d),(d_2,d_3,1d)\}\}$)

$\boxed{9\text{-}\mathcal{R}_\wedge}$

$\boxed{10}((^{HIV}Elisa_{1d}^{\#2},[d_2,d_3]),$
$\{\{d_{-1}<d_0<d_1<d_2<d_3<d_{11}\},\{\dots,\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d),(d_2,d_3,1d)\}\}$)

$\boxed{9\text{-}\mathcal{R}_\wedge}$

$\boxed{11}((\Diamond_r^*(^{HIV}WesternBlot_{1d}\wedge\Diamond_r^*(^{HIV}Registration_{4d}\wedge\dots)),[d_2,d_3]),$
$\{\{d_{-1}<d_0<d_1<d_2<d_3<d_{11}\},\{\dots,\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d),(d_2,d_3,1d)\}\}$)

$\boxed{11\text{-}\mathcal{R}_r^*}$

$\boxed{12}((^{HIV}WesternBlot_{1d}\wedge\Diamond_r^*(^{HIV}Registration_{4d}\wedge\dots),[d_3,d_4]),$
$\{\{d_{-1}<d_0<\dots<d_3<d_4<d_{11}\},\{\dots,\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d),(d_2,d_3,1d),(d_3,d_4,1d)\}\}$)

$\boxed{12\text{-}\mathcal{R}_\wedge}$

$\boxed{13}((^{HIV}WesternBlot_{1d},[d_3,d_4]),$
$\{\{\dots<d_2<d_3<d_4<d_{11}\},\{\dots,\ (d_{-1},d_1,218d1h),(d_1,d_2,1d),(d_2,d_3,1d),(d_3,d_4,1d)\}\}$)

$\boxed{12\text{-}\mathcal{R}_\wedge}$

$\boxed{14}((\Diamond_r^*(^{HIV}Registration_{4d}\wedge\Diamond_r^*(\top_{90d}\wedge\dots)),[d_3,d_4]),$
$\{\{\dots<d_2<d_3<d_4<d_{11}\},\{\dots,\ (d_{-1},d_1,6y64d6h),(d_1,d_2,1d),(d_2,d_3,1d),(d_3,d_4,1d)\}\}$)

$\boxed{14\text{-}\mathcal{R}_r^*}$

$\boxed{15}((^{HIV}Registration_{4d}\wedge\Diamond_r^*(\top_{90d}\wedge\dots),[d_4,d_5]),$
$\{\{\dots<d_3<d_4<d_5<d_{11}\},\{\dots,(d_1,d_2,1d),(d_2,d_3,1d),(d_3,d_4,1d),(d_4,d_5,4d)\}\}$)

Figure C.9: Tableau for ψ'_{HIV-TB} (Modeled in IMPNL)

$\boxed{16}((^{HIV}Registration_{4d},[d_4,d_5]),$
$\{\{\cdots < d_3 < d_4 < d_5 < d_{11}\},\{\ldots,\ (d_{-1},d_1,\ 6y64d6h),(d_1,d_2,\ 1d),(d_2,d_3,\ 1d),(d_3,d_4,\ 1d),(d_4,d_5,\ 4d)\}\})$
$\quad\big|\ \boxed{15\text{-}\mathcal{R}_\wedge}$
$\boxed{16}((\langle\rangle_r^*(\top_{90d} \wedge^4 \langle\rangle_l^- \langle\rangle_r^*(^{HIV}CD4Check_{1d} \wedge \ldots)),[d_4,d_5]),$
$\{\{\cdots < d_3 < d_4 < d_5 < d_{11}\},\{\ldots,\ (d_1,d_2,\ 1d),\ (d_2,d_3,\ 1d),\ (d_3,d_4,\ 1d),\ (d_4,d_5,\ 4d)\}\})$
$\quad\big|\ \boxed{16\text{-}\mathcal{R}_r^*}$
$\boxed{17}((\top_{90d} \wedge^4 \langle\rangle_l^- \langle\rangle_r^*(^{HIV}CD4Check_{1d} \wedge \ldots),[d_5,d_9]),$
$\{\{d_{-1} < \cdots < d_4 < d_5 < d_9 < d_{11}\},\{(d_0,d_1,2h),\ \ldots,\ (d_4,d_5,\ 4d),\ (d_5,d_9,\ 90d)\}\})$
$\quad\big|\ \boxed{17\text{-}\mathcal{R}_\wedge}$
$\boxed{18}((\top_{90d},[d_5,d_9]),$
$\{\{d_{-1} < \cdots < d_4 < d_5 < d_9 < d_{11}\},\{(d_0,d_1,2h),\ \ldots,\ (d_4,d_5,\ 4d),\ (d_5,d_9,\ 90d)\}\})$
$\quad\big|\ \boxed{17\text{-}\mathcal{R}_\wedge}$
$\boxed{19}((^4\langle\rangle_l^- \langle\rangle_r^*(^{HIV}CD4Check_{1d} \wedge \ldots),[d_5,d_9]),$
$\{\{d_{-1} < \cdots < d_4 < d_5 < d_9 < d_{11}\},\{(d_0,d_1,2h),\ \ldots,\ (d_4,d_5,\ 4d),\ (d_5,d_9,\ 90d)\}\})$
$\quad\big|\ \boxed{19\text{-}\mathcal{R}_l^-}$
$\boxed{20}((\langle\rangle_r^*(^{HIV}CD4Check_{1d} \wedge \ldots),[d_{-2},d_5]),$
$\{\{d_{-2} < d_{-1} < d_1 < \cdots < d_{11}\},\{(d_0,d_1,2h),\ \ldots,\ (d_4,d_5,\ 4d),\ (d_5,d_9,\ 90d),\ (d_{-2},d_5,\ 12y128d12h)\}\})$
$\quad\big|\ \boxed{20\text{-}\mathcal{R}_r^*}$
$\boxed{21}((^{HIV}CD4Check_{1d} \wedge^8 \langle\rangle_l^- \langle\rangle_r^*(\top_{30d} \wedge \ldots),[d_5,d_7]),$
$\{\{d_{-2} < d_{-1} < d_1 < \cdots < d_7 < d_9 < d_{11}\},\{\ldots,\ (d_5,d_9,\ 90d),\ (d_{-2},d_1,\ 12y128d12h),\ (d_5,d_7,\ 1d)\}\})$
$\quad\big|\ \boxed{21\text{-}\mathcal{R}_\wedge}$
$\boxed{22}((^{HIV}CD4Check_{1d},[d_5,d_7]),$
$\{\{d_{-2} < d_{-1} < d_1 < \cdots < d_7 < d_9 < d_{11}\},\{\ldots,\ (d_5,d_9,\ 90d),\ (d_{-2},d_1,\ 12y128d12h),\ (d_5,d_7,\ 1d)\}\})$
$\quad\big|\ \boxed{21\text{-}\mathcal{R}_\wedge}$
$\boxed{23}((^8\langle\rangle_l^- \langle\rangle_r^*(\top_{30d} \wedge^{16} \langle\rangle_l^- \langle\rangle_r^*(^{HIV}Visit_{1hour} \wedge \ldots)),[d_5,d_7]),$
$\{\{d_{-2} < d_{-1} < d_1 < \cdots < d_7 < d_9 < d_{11}\},\{\ldots,\ (d_5,d_9,\ 90d),\ (d_{-2},d_1,\ 12y128d12h),\ (d_5,d_7,\ 1d)\}\})$
$\quad\big|\ \boxed{23\text{-}\mathcal{R}_l^-}$
$\boxed{24}((\langle\rangle_r^*(\top_{30d} \wedge^{16} \langle\rangle_l^- \langle\rangle_r^*(^{HIV}Visit_{1hour} \wedge \ldots)),[d_{-3},d_5]),$
$\{\{d_{-3} < d_{-2} < d_{-1} < \cdots < d_7 < d_9 < d_{11}\},\{\ldots,\ (d_5,d_7,\ 1d),\ (d_{-3},d_5,\ 24yd256d)\}\})$
$\quad\big|\ \boxed{24\text{-}\mathcal{R}_r^*}$
$\boxed{25}((\top_{30d} \wedge^{16} \langle\rangle_l^- \langle\rangle_r^*(^{HIV}Visit_{1hour} \wedge \ldots),[d_5,d_8]),$
$\{\{d_{-3} < d_{-2} < \cdots < d_5 < d_7 < d_8 < d_9 < d_{11}\},\{\ldots,\ (d_{-3},d_5,\ 24yd256d),\ (d_5,d_8,\ 30d)\}\})$
$\quad\big|\ \boxed{25\text{-}\mathcal{R}_\wedge}$
$\boxed{26}((\top_{30d},[d_5,d_8]),$
$\{\{d_{-3} < d_{-2} < \cdots < d_5 < d_7 < d_8 < d_9 < d_{11}\},\{\ldots,\ (d_{-3},d_5,\ 24yd256d),\ (d_5,d_8,\ 30d)\}\})$
$\quad\big|\ \boxed{25\text{-}\mathcal{R}_\wedge}$
$\boxed{27}((^{16}\langle\rangle_l^- \langle\rangle_r^*(^{HIV}Visit_{1hour} \wedge \ldots),[d_5,d_8]),$
$\{\{d_{-3} < d_{-2} < \cdots < d_5 < d_7 < d_8 < d_9 < d_{11}\},\{(d_0,d_1,2h),\ \ldots,\ (d_5,d_7,\ 1d),\ (d_5,d_8,\ 30d)\}\})$
$\quad\big|\ \boxed{27\text{-}\mathcal{R}_l^-}$
$\boxed{28}((\langle\rangle_r^*(^{HIV}Visit_{1hour} \wedge \ldots),[d_{-4},d_5]),$
$\{\{d_{-4} < d_{-3} < d_{-2} < \cdots < d_4 < d_5 < \ldots\},\{\ldots,\ (d_5,d_7,\ 1d),\ (d_5,d_8,\ 30d),\ (d_{-4},d_5,\ 49y156d)\}\})$
$\quad\big|\ \boxed{27\text{-}\mathcal{R}_r^*}$
$\boxed{29}((^{HIV}Visit_{1hour} \wedge^{32} \langle\rangle_l^- \langle\rangle_r^*(^{HIV}TakeKaletra_{30d} \wedge \ldots),[d_5,d_6]),$
$\{\{d_{-4} < d_{-3} < \cdots < d_5 < d_6 < d_7 < \ldots\},\{(d_0,d_1,2h),\ \ldots,\ (d_{-4},d_5,\ 49y156d),\ (d_5,d_6,\ 1h)\}\})$

Figure C.10: Tableau for ψ'_{HIV-TB} (Modeled in IMPNL)

$\boxed{30}((^{HIV}Visit_{1hour},[d_5,d_6]),$
$\{\{d_{-4}<d_{-3}<\cdots<d_5<d_6<d_7<\ldots\},\{\ldots,(d_{-2},d_5,12y128d12h),(d_5,d_7,1d),(d_5,d_6,1h)\}\})$
$\boxed{29\text{-}\mathcal{R}_\wedge}$
$\boxed{31}((^{32}\Diamond_l^-\Diamond_r^*(^{HIV}TakeKaletra_{30d}\wedge\ldots),[d_5,d_6]),$
$\{\{d_{-4}<d_{-3}<\cdots<d_5<d_6<d_7<\ldots\},\{\ldots,(d_{-2},d_5,12y128d12h),(d_5,d_7,1d),(d_5,d_6,1h)\}\})$
$\boxed{31\text{-}\mathcal{R}_l^-}$
$\boxed{32}((\Diamond_r^*(^{HIV}TakeKaletra_{30d}\wedge\ldots),[d_{-5},d_5]),$
$\{\{d_{-5}<d_{-4}<\cdots<d_4<d_5<d_6<\ldots\},\{(d_0,d_1,2h),\ldots,(d_5,d_6,1h),(d_{-5},d_5,98y312d)\}\})$
$\boxed{32\text{-}\mathcal{R}_r^*}$
$\boxed{33}((^{HIV}TakeKaletra_{30d}\wedge\neg^{HIV}TakeAlfuzosin_{30d}\wedge\ldots,[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{33\text{-}\mathcal{R}_\wedge}$
$\boxed{34}((^{HIV}TakeKaletra_{30d},[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{33\text{-}\mathcal{R}_\wedge}$
$\boxed{35}((\neg^{HIV}TakeAlfuzosin_{30d}\wedge\neg^{HIV}TakeCisapride_{30d}\wedge\ldots,[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{35\text{-}\mathcal{R}_\wedge}$
$\boxed{36}((\neg^{HIV}TakeAlfuzosin_{30d},[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{35\text{-}\mathcal{R}_\wedge}$
$\boxed{37}((\neg^{HIV}TakeCisapride_{30d}\wedge\neg^{HIV}TakeRifampin_{30d}\wedge\ldots,[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{37\text{-}\mathcal{R}_\wedge}$
$\boxed{38}((\neg^{HIV}TakeCisapride_{30d},[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{37\text{-}\mathcal{R}_\wedge}$
$\boxed{39}((\neg^{HIV}TakeRifampin_{30d}\wedge\neg^{HIV}TakePimozide_{30d}\wedge\ldots,[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{39\text{-}\mathcal{R}_\wedge}$
$\boxed{40}((\neg^{HIV}TakeRifampin_{30d},[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{39\text{-}\mathcal{R}_\wedge}$
$\boxed{41}((\neg^{HIV}TakePimozide_{30d}\wedge^{HIV}TakeTenofovir_{30d}\wedge\ldots,[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{41\text{-}\mathcal{R}_\wedge}$
$\boxed{42}((\neg^{HIV}TakePimozide_{30d},[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{41\text{-}\mathcal{R}_\wedge}$
$\boxed{43}((^{HIV}TakeTenofovir_{30d}\wedge^{HIV}TakeLamivadin_{30d},[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$
$\boxed{43\text{-}\mathcal{R}_\wedge}$
$\boxed{44}((^{HIV}TakeTenofovir_{30d},[d_5,d_8]),$
$\{\{\cdots<d_4<d_5<d_6<d_7<d_8<d_9<\ldots\},\{\ldots,(d_5,d_9,90d),(d_{-2},d_5,12y128d12h),(d_5,d_8,30d)\}\})$

Figure C.11: Tableau for ψ'_{HIV-TB} (Modeled in IMPNL)

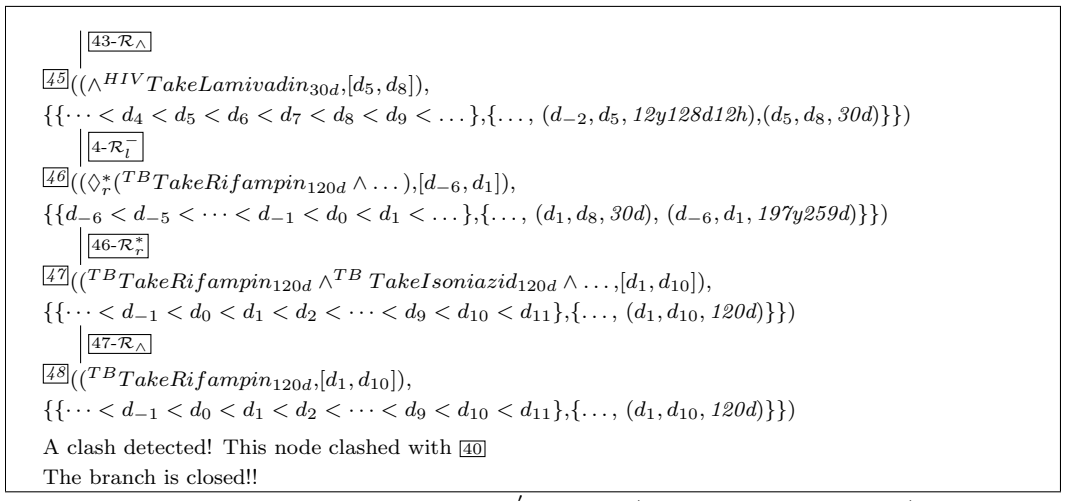

Figure C.12: Tableau for ψ'_{HIV-TB} (Modeled in IMPNL)

C.4 Tableau for φ'_{HIV-TB}

$\boxed{1}[root]\ ((\psi'_{HIV-TB},[e_0,e_1]),\ \{\{e_0 < e_1\},\{(e_0,e_1,2h)\}\})$

$\quad\big|\ \boxed{1\text{-}\mathcal{R}^*_r}$

$\boxed{2}((^2\Diamond_l^-\Diamond_r^*(((\neg Kaletra \sqcap \neg Norvir \sqcap \neg Aptivus) \sqcup ProteaseInhibitors = \top)_{120yr} \wedge \dots) \wedge^4 \Diamond_l^-\Diamond_r^*(\dots)$
$\wedge \dots,[e_1,e_2]),\{\{e_0 < e_1 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa)\}\})$

$\quad\big|\ \boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{3}((^2\Diamond_l^-\Diamond_r^*(((\neg Kaletra \sqcap \neg Norvir \sqcap \neg Aptivus) \sqcup ProteaseInhibitors = \top)_{120yr} \wedge \dots),[e_1,e_2]),$
$\{\{e_0 < e_1 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa)\}\})$

$\quad\big|\ \boxed{2\text{-}\mathcal{R}_\wedge}$

$\boxed{4}((^4\Diamond_l^-\Diamond_r^*((p : Patient)_{120yr} \wedge \dots) \wedge^{256} \Diamond_l^-\Diamond_r^*((p : Patient)_{6m} \wedge \dots),[e_1,e_2]),$
$\{\{e_0 < e_1 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa)\}\})$

$\quad\big|\ \boxed{4\text{-}\mathcal{R}_l^-}$

$\boxed{5}((\Diamond_r^*(((\neg Kaletra \sqcap \neg Norvir \sqcap \neg Aptivus) \sqcup ProteaseInhibitors = \top)_{120yr} \wedge \dots),[e_{-1},e_1]),$
$\{\{e_{-1} < e_0 < e_1 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa)\}\})$

$\quad\big|\ \boxed{5\text{-}\mathcal{R}^*_r}$

$\boxed{6}((((\neg Kaletra \sqcap \neg Norvir \sqcap \neg Aptivus) \sqcup ProteaseInhibitors = \top)_{120yr} \wedge \dots,[e_1,e_3]),$
$\{\{e_{-1} < e_0 < e_1 < e_3 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr)\}\})$

$\quad\big|\ \boxed{6\text{-}\mathcal{R}_\wedge}$

$\boxed{7}((((\neg Kaletra \sqcap \neg Norvir \sqcap \neg Aptivus) \sqcup ProteaseInhibitors = \top)_{120yr},[e_1,e_3]),$
$\{\{e_{-1} < e_0 < e_1 < e_3 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr)\}\})$

$\quad\big|\ \boxed{6\text{-}\mathcal{R}_\wedge}$

$\boxed{8}(((\forall TakeMedicine.\neg Rifampin \sqcup \forall TakeMedicine.\neg ProteaseInhibitors = \top)_{120yr},[e_1,e_3]),$
$\{\{e_{-1} < e_0 < e_1 < e_3 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr)\}\})$

$\quad\big|\ \boxed{4\text{-}\mathcal{R}_\wedge}$

$\boxed{9}((^4\Diamond_l^-\Diamond_r^*((p : Patient)_{120yr} \wedge \dots),[e_1,e_2]),$
$\{\{e_{-1} < e_0 < e_1 < e_3 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr)\}\})$

$\quad\big|\ \boxed{4\text{-}\mathcal{R}_\wedge}$

$\boxed{10}((^{256}\Diamond_l^-\Diamond_r^*((p : Patient)_{6m} \wedge \dots),[e_1,e_2]),$
$\{\{e_{-1} < e_0 < e_1 < e_3 < e_2\},\{(e_0,e_1,2h),\ (e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr)\}\})$

$\quad\big|\ \boxed{9\text{-}\mathcal{R}_l^-}$

$\boxed{11}((\Diamond_r^*((p : Patient)_{120yr} \wedge \dots),[e_{-2},e_1]),$
$\{\{e_{-2} < e_{-1} < e_0 < e_1 < e_3 < e_2\},\{\dots,(e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr),\ (e_{-2},e_1,4*\kappa)\}\})$

$\quad\big|\ \boxed{11\text{-}\mathcal{R}^*_r}$

$\boxed{12}(((p : Patient)_{120yr} \wedge \dots,[e_1,e_3]),$
$\{\{e_{-2} < e_{-1} < e_0 < e_1 < e_3 < e_2\},\{\dots,(e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr),\ (e_{-2},e_1,4*\kappa)\}\})$

$\quad\big|\ \boxed{12\text{-}\mathcal{R}_\wedge}$

$\boxed{13}(((p : Patient)_{120yr},[e_1,e_3]),$
$\{\{e_{-2} < e_{-1} < e_0 < e_1 < e_3 < e_2\},\{\dots,(e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr),\ (e_{-2},e_1,4*\kappa)\}\})$

$\quad\big|\ \boxed{12\text{-}\mathcal{R}_\wedge}$

$\boxed{14}((^8\Diamond_l^-\Diamond_r^*((e1 : Elisa)_{1d} \wedge \dots),[e_1,e_3]),$
$\{\{e_{-2} < e_{-1} < e_0 < e_1 < e_3 < e_2\},\{\dots,(e_1,e_2,\kappa),\ (e_{-1},e_1,2*\kappa),\ (e_1,e_3,120yr),\ (e_{-2},e_1,4*\kappa)\}\})$

$\quad\big|\ \boxed{14\text{-}\mathcal{R}_l^-}$

$\boxed{15}((\Diamond_r^*((e1 : Elisa)_{1d} \wedge \dots),[e_{-3},e_1]),$
$\{\{e_{-3} < e_{-2} < e_{-1} < \dots < e_3 < e_2\},\{\dots,(e_1,e_3,120yr),\ (e_{-2},e_1,4*\kappa),\ (e_{-3},e_1,8*\kappa)\}\})$

Figure C.13: Tableau for ψ'_{HIV-TB} (Modeled in MITDL)

$\boxed{15\text{-}\mathcal{R}_r^*}$

$\boxed{16}$ $(((e1 : Elisa)_{1d} \wedge \ldots,[e_1, e_4]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_3 < e_2\},\{\ldots, (e_1, e_3, 120yr), (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d)\}\})$

$\boxed{16\text{-}\mathcal{R}_\wedge}$

$\boxed{17}$ $(((e1 : Elisa)_{1d},[e_1, e_4]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_3 < e_2\},\{\ldots, (e_1, e_3, 120yr), (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d)\}\})$

$\boxed{16\text{-}\mathcal{R}_\wedge}$

$\boxed{18}$ $((PerformTest(p, e1)_{1d} \wedge \ldots,[e_1, e_4]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_3 < e_2\},\{\ldots, (e_1, e_3, 120yr), (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d)\}\})$

$\boxed{18\text{-}\mathcal{R}_\wedge}$

$\boxed{19}$ $((PerformTest(p, e1)_{1d},[e_1, e_4]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_3 < e_2\},\{\ldots, (e_1, e_3, 120yr), (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d)\}\})$

$\boxed{18\text{-}\mathcal{R}_\wedge}$

$\boxed{20}$ $((\Diamond_r^*((e2 : Elisa)_{1d} \wedge \ldots),[e_1, e_4]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_3 < e_2\},\{\ldots, (e_1, e_3, 120yr), (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d)\}\})$

$\boxed{20\text{-}\mathcal{R}_r^*}$

$\boxed{21}$ $(((e2 : Elisa)_{1d} \wedge \ldots,[e_4, e_5]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_5 < e_3 < e_2\},\{\ldots, (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d)\}\})$

$\boxed{21\text{-}\mathcal{R}_\wedge}$

$\boxed{22}$ $(((e2 : Elisa)_{1d},[e_4, e_5]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_5 < e_3 < e_2\},\{\ldots, (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d)\}\})$

$\boxed{21\text{-}\mathcal{R}_\wedge}$

$\boxed{23}$ $((PerformTest(p, e2)_{1d} \wedge \ldots,[e_4, e_5]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_5 < e_3 < e_2\},\{\ldots, (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d)\}\})$

$\boxed{23\text{-}\mathcal{R}_\wedge}$

$\boxed{24}$ $((PerformTest(p, e2)_{1d},[e_4, e_5]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_5 < e_3 < e_2\},\{\ldots, (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d)\}\})$

$\boxed{23\text{-}\mathcal{R}_\wedge}$

$\boxed{25}$ $((\Diamond_r^*((w : WesternBlot)_{1d} \wedge \ldots),[e_4, e_5]),$
$\{\{e_{-3} < \cdots < e_1 < e_4 < e_5 < e_3 < e_2\},\{\ldots, (e_{-2}, e_1, 4 * \kappa), (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d)\}\})$

$\boxed{25\text{-}\mathcal{R}_\wedge}$

$\boxed{26}$ $(((w : WesternBlot)_{1d} \wedge \ldots,[e_5, e_6]),$
$\{\{e_{-3} < \cdots < e_4 < e_5 < e_6 < e_3 < e_2\},\{\ldots, (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d)\}\})$

$\boxed{26\text{-}\mathcal{R}_\wedge}$

$\boxed{27}$ $(((w : WesternBlot)_{1d},[e_5, e_6]),$
$\{\{e_{-3} < \cdots < e_4 < e_5 < e_6 < e_3 < e_2\},\{\ldots, (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d)\}\})$

$\boxed{26\text{-}\mathcal{R}_\wedge}$

$\boxed{28}$ $((PerformTest(p, w)_{1d} \wedge \ldots,[e_5, e_6]),$
$\{\{e_{-3} < \cdots < e_4 < e_5 < e_6 < e_3 < e_2\},\{\ldots, (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d)\}\})$

$\boxed{28\text{-}\mathcal{R}_\wedge}$

$\boxed{29}$ $((PerformTest(p, w)_{1d},[e_5, e_6]),$
$\{\{e_{-3} < \cdots < e_4 < e_5 < e_6 < e_3 < e_2\},\{\ldots, (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d)\}\})$

$\boxed{28\text{-}\mathcal{R}_\wedge}$

$\boxed{30}$ $((\Diamond_r^*((r : Registration)_{4d} \wedge \ldots),[e_5, e_6]),$
$\{\{e_{-3} < \cdots < e_4 < e_5 < e_6 < e_3 < e_2\},\{\ldots, (e_{-3}, e_1, 8 * \kappa), (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d)\}\})$

Figure C.14: Tableau for φ'_{HIV-TB} (Modeled in MITDL)

$\boxed{30\text{-}\mathcal{R}_r^*}$

$\boxed{31}$ $(((r : Registration)_{4d} \wedge \ldots, [e_6, e_7]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_3 < e_2\}, \{\ldots, (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d)\}\})$

$\boxed{31\text{-}\mathcal{R}_\wedge}$

$\boxed{32}(((r : Registration)_{4d}, [e_6, e_7]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_3 < e_2\}, \{\ldots, (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d)\}\})$

$\boxed{31\text{-}\mathcal{R}_\wedge}$

$\boxed{33}((AdministrativeAction(p, r)_{4d} \wedge \ldots, [e_6, e_7]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_3 < e_2\}, \{\ldots, (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d)\}\})$

$\boxed{33\text{-}\mathcal{R}_\wedge}$

$\boxed{34}((AdministrativeAction(p, r)_{4d}, [e_6, e_7]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_3 < e_2\}, \{\ldots, (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d)\}\})$

$\boxed{33\text{-}\mathcal{R}_\wedge}$

$\boxed{35}((\Diamond_r^*(\top_{90d} \wedge \ldots), [e_6, e_7]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_3 < e_2\}, \{\ldots, (e_1, e_4, 1d), (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d)\}\})$

$\boxed{35\text{-}\mathcal{R}_r^*}$

$\boxed{36}((\top_{90d} \wedge \ldots, [e_7, e_8]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_8 < e_3 < e_2\}, \{\ldots, (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d)\}\})$

$\boxed{36\text{-}\mathcal{R}_\wedge}$

$\boxed{37}(((\top_{90d}, [e_7, e_8]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_8 < e_3 < e_2\}, \{\ldots, (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d)\}\})$

$\boxed{36\text{-}\mathcal{R}_\wedge}$

$\boxed{38}((^{16}\Diamond_l^- \Diamond_r^*((c : CD4Check)_{1d} \wedge \ldots), [e_7, e_8]),$
$\{\{e_{-3} < \cdots < e_5 < e_6 < e_7 < e_8 < e_3 < e_2\}, \{\ldots, (e_4, e_5, 1d), (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d)\}\})$

$\boxed{38\text{-}\mathcal{R}_l^-}$

$\boxed{39}((\Diamond_r^*((c : CD4Check)_{1d} \wedge \ldots), [e_{-4}, e_7]),$
$\{\{e_{-4} < e_{-3} < \cdots < e_6 < e_7 < e_8 < e_3 < e_2\}, \{\ldots, (e_6, e_7, 4d), (e_7, e_8, 90d), (e_{-4}, e_7, 16 * \kappa)\}\})$

$\boxed{39\text{-}\mathcal{R}_r^*}$

$\boxed{40}(((c : CD4Check)_{1d} \wedge \ldots, [e_7, e_9]),$
$\{\{e_{-3} < \cdots < e_6 < e_7 < e_9 < e_8 < e_3 < e_2\}, \{\ldots, (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d), (e_7, e_9, 1d)\}\})$

$\boxed{40\text{-}\mathcal{R}_\wedge}$

$\boxed{41}(((c : CD4Check)_{1d}, [e_7, e_9]),$
$\{\{e_{-3} < \cdots < e_6 < e_7 < e_9 < e_8 < e_3 < e_2\}, \{\ldots, (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d), (e_7, e_9, 1d)\}\})$

$\boxed{40\text{-}\mathcal{R}_\wedge}$

$\boxed{42}((PerformTest(p, c)_{1d} \wedge \ldots, [e_7, e_9]),$
$\{\{e_{-3} < \cdots < e_6 < e_7 < e_9 < e_8 < e_3 < e_2\}, \{\ldots, (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d), (e_7, e_9, 1d)\}\})$

$\boxed{42\text{-}\mathcal{R}_\wedge}$

$\boxed{43}((PerformTest(p, c)_{1d}, [e_7, e_9]),$
$\{\{e_{-3} < \cdots < e_6 < e_7 < e_9 < e_8 < e_3 < e_2\}, \{\ldots, (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d), (e_7, e_9, 1d)\}\})$

$\boxed{42\text{-}\mathcal{R}_\wedge}$

$\boxed{44}((^{32}\Diamond_l^- \Diamond_r^*(\top_{30d} \wedge \ldots), [e_7, e_9]),$
$\{\{e_{-3} < \cdots < e_6 < e_7 < e_9 < e_8 < e_3 < e_2\}, \{\ldots, (e_5, e_6, 1d), (e_6, e_7, 4d), (e_7, e_8, 90d), (e_7, e_9, 1d)\}\})$

$\boxed{44\text{-}\mathcal{R}_l^-}$

$\boxed{45}((\Diamond_r^*(\top_{30d} \wedge \ldots), [e_{-5}, e_7]),$
$\{\{e_{-5} < e_{-3} < \cdots < e_6 < e_7 < e_9 < \ldots\}, \{\ldots, (e_7, e_8, 90d), (e_7, e_9, 1d), (e_{-5}, e_7, 32 * \kappa)\}\})$

Figure C.15: Tableau for φ'_{HIV-TB} (Modeled in MITDL)

$\boxed{45\text{-}\mathcal{R}_r^*}$

$\boxed{46}$ $((\top_{30d} \wedge \ldots, [e_7, e_{10}]),$
$\{\{\cdots < e_6 < e_7 < e_9 < e_{10} < e_8 < \ldots\}, \{\ldots, (e_7, e_8, 90d), (e_7, e_9, 1d), (e_{-5}, e_7, 32 * \kappa), (e_7, e_{10}, 30d)\}\})$

$\boxed{46\text{-}\mathcal{R}_\wedge}$

$\boxed{47}((\top_{30d}, [e_7, e_{10}]),$
$\{\{\cdots < e_6 < e_7 < e_9 < e_{10} < e_8 < \ldots\}, \{\ldots, (e_7, e_8, 90d), (e_7, e_9, 1d), (e_{-5}, e_7, 32 * \kappa), (e_7, e_{10}, 30d)\}\})$

$\boxed{46\text{-}\mathcal{R}_\wedge}$

$\boxed{48}(({}^{64}\Diamond_l^- \Diamond_r^*(((p : HasAIDS)_{1h} \wedge \ldots)), [e_7, e_{10}]),$
$\{\{\cdots < e_6 < e_7 < e_9 < e_{10} < e_8 < \ldots\}, \{\ldots, (e_7, e_8, 90d), (e_7, e_9, 1d), (e_{-5}, e_7, 32 * \kappa), (e_7, e_{10}, 30d)\}\})$

$\boxed{48\text{-}\mathcal{R}_l^-}$

$\boxed{49}((\Diamond_r^*(((p : HasAIDS)_{1h} \wedge \ldots)), [e_{-6}, e_7]),$
$\{\{e_{-6} < e_{-5} < \cdots < e_6 < e_7 < e_9 < e_{10} < \ldots\}, \{\ldots, (e_{-5}, e_7, 32 * \kappa), (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa)\}\})$

$\boxed{49\text{-}\mathcal{R}_r^*}$

$\boxed{50}((((p : HasAIDS)_{1h} \wedge \ldots), [e_7, e_{11}]),$
$\{\{\cdots < e_6 < e_7 < e_{11} < e_9 < e_{10} < \ldots\}, \{\ldots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{50\text{-}\mathcal{R}_\wedge}$

$\boxed{51}(((p : HasAIDS)_{1h}, [e_7, e_{11}]),$
$\{\{\cdots < e_6 < e_7 < e_{11} < e_9 < e_{10} < \ldots\}, \{\ldots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{50\text{-}\mathcal{R}_\wedge}$

$\boxed{52}(((p : \exists Visitedby.Doctor)_{1h} \wedge \ldots, [e_7, e_{11}]),$
$\{\{\cdots < e_6 < e_7 < e_{11} < e_9 < e_{10} < \ldots\}, \{\ldots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{52\text{-}\mathcal{R}_\wedge}$

$\boxed{53}(((p : \exists Visitedby.Doctor)_{1h}, [e_7, e_{11}]),$
$\{\{\cdots < e_6 < e_7 < e_{11} < e_9 < e_{10} < \ldots\}, \{\ldots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{52\text{-}\mathcal{R}_\wedge}$

$\boxed{54}(({}^{128}\Diamond_l^- \Diamond_r^*((k : Kaletra)_{30d} \wedge \ldots), [e_7, e_{11}]),$
$\{\{\cdots < e_6 < e_7 < e_{11} < e_9 < e_{10} < \ldots\}, \{\ldots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{54\text{-}\mathcal{R}_l^-}$

$\boxed{55}((\Diamond_r^*((k : Kaletra)_{30d} \wedge \ldots), [e_{-7}, e_7]),$
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \ldots\}, \{\ldots, (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h), (e_{-7}, e_7, 128 * \kappa)\}\})$

$\boxed{55\text{-}\mathcal{R}_r^*}$

$\boxed{56}(((k : Kaletra)_{30d} \wedge \ldots, [e_7, e_{10}]),$
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \ldots\}, \{\ldots, (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h), (e_{-7}, e_7, 128 * \kappa)\}\})$

$\boxed{56\text{-}\mathcal{R}_\wedge}$

$\boxed{57}(((k : Kaletra)_{30d}, [e_7, e_{10}]),$
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \ldots\}, \{\ldots, (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h), (e_{-7}, e_7, 128 * \kappa)\}\})$

$\boxed{56\text{-}\mathcal{R}_\wedge}$

$\boxed{58}(((t : Tenofovir)_{30d} \wedge \ldots, [e_7, e_{10}]),$
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \ldots\}, \{\ldots, (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h), (e_{-7}, e_7, 128 * \kappa)\}\})$

$\boxed{58\text{-}\mathcal{R}_\wedge}$

$\boxed{59}(((t : Tenofovir)_{30d}, [e_7, e_{10}]),$
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \ldots\}, \{\ldots, (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h), (e_{-7}, e_7, 128 * \kappa)\}\})$

$\boxed{58\text{-}\mathcal{R}_\wedge}$

$\boxed{60}(((l : Lamivadin)_{30d} \wedge \ldots, [e_7, e_{10}]),$
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \ldots\}, \{\ldots, (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h), (e_{-7}, e_7, 128 * \kappa)\}\})$

Figure C.16: Tableau for φ'_{HIV-TB} (Modeled in MITDL)

$\boxed{60\text{-}\mathcal{R}_\wedge}$

$\boxed{61}$ $(((l : Lamivadin)_{30d},[e_7, e_{10}])$,
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \dots\},\{\dots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{60\text{-}\mathcal{R}_\wedge}$

$\boxed{62}((TakeMedicine(p, k)_{30d} \wedge \dots,[e_7, e_{10}])$,
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \dots\},\{\dots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{62\text{-}\mathcal{R}_\wedge}$

$\boxed{63}((TakeMedicine(p, k)_{30d},[e_7, e_{10}])$,
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \dots\},\{\dots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{62\text{-}\mathcal{R}_\wedge}$

$\boxed{64}((TakeMedicine(p, t)_{30d} \wedge \dots,[e_7, e_{10}])$,
$\{\{e_{-7} < e_{-6} < e_{-5} < \cdots < e_6 < e_7 < \dots\},\{\dots, (e_7, e_{10}, 30d), (e_{-6}, e_7, 64 * \kappa), (e_7, e_{11}, 1h)\}\})$

$\boxed{10\text{-}\mathcal{R}_l^-}$

$\boxed{65}((\Diamond_r^*((p : Patient)_{6m} \wedge \dots),[e_{-8}, e_1])$,
$\{\{e_{-8} < e_{-7} < \cdots < e_1 < e_4 < \dots\},\{\dots, (e_7, e_{11}, 1h), (e_{-7}, e_7, 128 * \kappa), (e_{-8}, e_1, 256 * \kappa)\}\})$

$\boxed{65\text{-}\mathcal{R}_r^*}$

$\boxed{66}(((p : Patient)_{6m} \wedge \dots,[e_1, e_{12}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_8 < e_{12} < e_3 < e_2\},\{\dots, (e_{-7}, e_7, 128 * \kappa), (e_{-8}, e_1, 256 * \kappa), (e_1, e_{12}, 6m)\}\})$

$\boxed{66\text{-}\mathcal{R}_\wedge}$

$\boxed{67}(((p : Patient)_{6m},[e_1, e_{12}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_8 < e_{12} < e_3 < e_2\},\{\dots, (e_{-7}, e_7, 128 * \kappa), (e_{-8}, e_1, 256 * \kappa), (e_1, e_{12}, 6m)\}\})$

$\boxed{66\text{-}\mathcal{R}_\wedge}$

$\boxed{68}(({}^{512}\Diamond_l^-\Diamond_r^*((r_1 : Rifampin)_{60d} \wedge \dots),[e_1, e_{12}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_8 < e_{12} < e_3 < e_2\},\{\dots, (e_{-7}, e_7, 128 * \kappa), (e_{-8}, e_1, 256 * \kappa), (e_1, e_{12}, 6m)\}\})$

$\boxed{68\text{-}\mathcal{R}_l^-}$

$\boxed{69}((\Diamond_r^*((r_1 : Rifampin)_{60d} \wedge \dots),[e_{-9}, e_1])$,
$\{\{e_{-9} < e_{-8} < \cdots < e_1 < e_4 < \dots\},\{\dots, (e_{-8}, e_1, 256 * \kappa), (e_1, e_{12}, 6m), (e_{-9}, e_1, 512 * \kappa)\}\})$

$\boxed{69\text{-}\mathcal{R}_r^*}$

$\boxed{70}(((r_1 : Rifampin)_{60d} \wedge \dots,[e_1, e_{13}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_{10} < e_{13} < e_8 < \dots\},\{\dots, (e_{-9}, e_1, 512 * \kappa), (e_1, e_{13}, 60d)\}\})$

$\boxed{70\text{-}\mathcal{R}_\wedge}$

$\boxed{71}((r_1 : Rifampin)_{60d},[e_1, e_{13}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_{10} < e_{13} < e_8 < \dots\},\{\dots, (e_{-9}, e_1, 512 * \kappa), (e_1, e_{13}, 60d)\}\})$

$\boxed{70\text{-}\mathcal{R}_\wedge}$

$\boxed{72}((TakeMedicine(p, r_1)_{60d} \wedge \dots,[e_1, e_{13}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_{10} < e_{13} < e_8 < \dots\},\{\dots, (e_{-9}, e_1, 512 * \kappa), (e_1, e_{13}, 60d)\}\})$

$\boxed{72\text{-}\mathcal{R}_\wedge}$

$\boxed{73}((TakeMedicine(p, r_1)_{60d},[e_1, e_{13}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_{10} < e_{13} < e_8 < \dots\},\{\dots, (e_{-9}, e_1, 512 * \kappa), (e_1, e_{13}, 60d)\}\})$

$\boxed{72\text{-}\mathcal{R}_\wedge}$

$\boxed{74}(((is_1 : Isoniazid)_{60d} \wedge \dots,[e_1, e_{13}])$,
$\{\{\cdots < e_1 < e_4 < \cdots < e_{10} < e_{13} < e_8 < \dots\},\{\dots, (e_{-9}, e_1, 512 * \kappa), (e_1, e_{13}, 60d)\}\})$

Figure C.17: Tableau for φ'_{HIV-TB} (Modeled in MITDL)

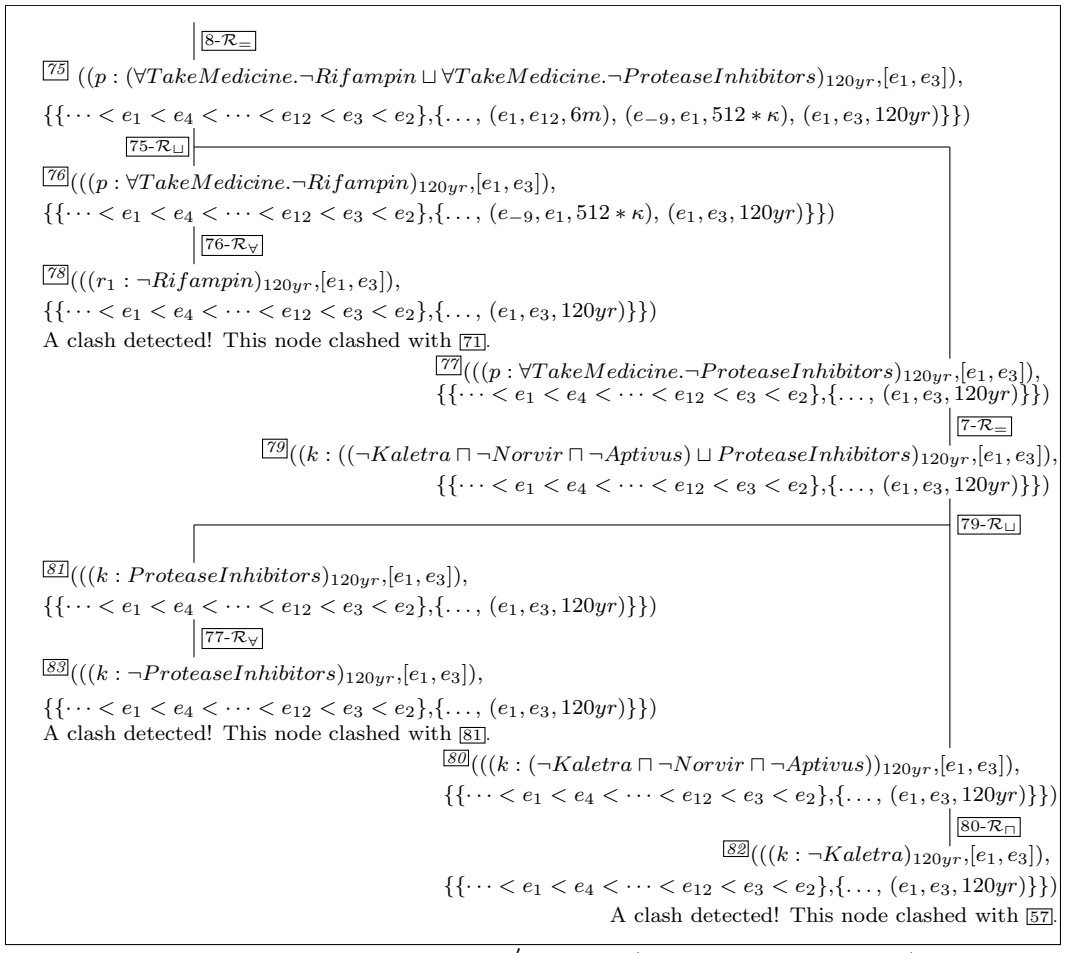

Figure C.18: Tableau for φ'_{HIV-TB} (Modeled in MITDL)