UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA
DIPARTIMENTO DI INFORMATICA E COMUNICAZIONE

Dottorato di Ricerca in Informatica
XXIII Ciclo – 2007-2010
Ph.D. Thesis

Kripke Semantics and Tableau Procedures for Constructive Description Logics

Loris Bozzato

Advisor:
Prof. Mauro Ferrari

Supervisor of the Doctoral program:
Prof. Gaetano Aurelio Lanzarone

Dedicated to my wonderful family.

Abstract

In this work we present the decidable constructive description logic $\mathcal{KALC}$: the logic is based on a Kripke-style semantics for the language of the description logic $\mathcal{ALC}$ and it is directly inspired by the Kripke semantics for first order intuitionistic logic. We study the constructive properties of this logic and its relations with classical semantics. Then, by means of an example, we show how its semantics is suitable for the description of incomplete and dynamic knowledge. We then introduce a tableau calculus for this logic and we prove its completeness with respect to $\mathcal{KALC}$ semantics. Most notably, by proving the completeness and termination results for such calculus, we obtain an effective proof search algorithm for our logic. We also study the relations of $\mathcal{KALC}$ with our previous proposals for constructive description logics, with first order intuitionistic logic and with well-known intuitionistic multi-modal logics. We conclude by presenting an application for a different constructive semantics for $\mathcal{KALC}$ in the context of Semantic Web services composition.

Acknowledgements

I would like to start by thanking my advisor prof. Mauro Ferrari for constantly supporting me during my Ph.D. and for helping me in projecting and reviewing the presentation of the results in this work. But, most of all, I thank him for believing in my capabilities and ideas for this work, which would not have been possible without its friendly support.

I wish to thank prof. Camillo Fiorentini and prof. Guido Fiorino for the essential discussions about the formulation of many of the results introduced in this thesis and for their irreplaceable collaboration since our first ideas on constructive description logics. I am also grateful to prof. Gaetano Aurelio Lanzarone, supervisor of the doctoral program, for giving me the opportunity to work in my research topic since the beginning of my Ph.D. course.

I want to thank all of my Ph.D. colleagues, with whom I shared both the work and fun moments of the intense last three years: among them, particular thanks go to Paola Villa for her invaluable collaboration in our past works; sincere thanks to Paolo Brivio, Stefano Braghin, Michele Chinosi, Pietro Colombo, Andrea Carcano and Lorenzo Bossi. I also thank all of the members and staff of our Department for creating an always welcoming and stimulating working environment.

Finally, I thank my family and friends for always encouraging me throughout these years.

Grazie / Thanks,
Loris

Contents

1

Introduction

1.1 Description logics

Description logics (DLs) [Baader and Nutt(2003)] are a family of logical based knowledge representation formalisms, which are currently becoming particularly known for their applications. Description logics are born from the need of a logical characterization for non-logic and *network based* knowledge representation systems, such as *frame systems* [Minsky(1981)] and *semantic networks* [Brachman(1979)]. From a formal point of view, description logics are expressive though decidable fragments of classical first order logic: in fact, the actual research on description logics is regarded to originate from the study [Brachman and Levesque(1984)] of the central question of the relation between expressivity and complexity of such fragments.

The study of formal properties of description logics has been developed in parallel with the investigation of implementation and optimization for their reasoning problems. Many of the advancements in description logics expressivity have been driven by their need in applications. Indeed, because of their well defined semantics and the efficient ways of reasoning they support, description logics are nowadays at the basis of many relevant applications: most notably, description logics represent the logical base of the ontology representation languages for the *Semantic Web*, like the W3C standard language *OWL (Web Ontology Language)* [van Harmelen and McGuinness(2004), Patel-Schneider *et al.*(2009)].

Intuitively, description logics represent knowledge about a domain by means of two basic representation elements, *concepts* and *roles*. Concepts represent classes of objects (or, more precisely, *individuals*), while roles are used to express binary relationships between objects. For example, the concept `Human` can be used to represent the set of all human beings while the role `hasChild` could represent the relation between parent and son.

So, concept and roles represent the elementary *descriptions* of the language: complex descriptions are obtained by combining concept and roles by means of logical operators. We can distinguish two kinds of such operators: *concept construc-*

tors and *role restrictions*. Concept constructors can be used to represent the usual boolean set operators between concepts, as conjunction ($\sqcap$), disjunction ($\sqcup$) and negation ($\neg$). For example, $C \sqcup D$ describes the concept obtained by the union of two concepts C and D. Roles restrictions build new concepts by means of roles, basically by restricting the types of possible relationships involving elements of such concepts. Some typical role restrictions are *universal restriction* ($\forall$), *existential restriction* ($\exists$) and *number restriction* ($\geqslant$): as an example, the concept $\forall R.C$ represents the set of individuals such that, if any object is related to them by the role R, then this must belong to the concept C.

The semantics of description logics (in the case of *descriptive semantics*) is based on the set theoretical interpretation of classical first order logics and supports the intuitive definitions given above: concepts are interpreted as subsets of a given non empty set (the *domain* of the interpretation) while roles are interpreted as binary relations (set of pairs) over the domain. The semantics of complex concepts is based on the valuation of operators: for example, as noted above, concept constructors are interpreted as the usual set operations over the sets denoted by concepts.

The number and combination of operators defines a specific description logic and determines its expressivity and the complexity of reasoning over its representations. The addition of new constructors to the base operators is, in general, motivated by knowledge representation needs: such additions are directly related to the study on the decidability and complexity of the resulting logics. The minimal description logic languages can be identified with the family of the description logic $\mathcal{AL}$ *(attributive language)*. The most relevant extension of the $\mathcal{AL}$ logic is $\mathcal{ALC}$ *(attributive language with complements)*: this logic is usually considered as the reference basic description logic, since it is the smallest description logic closed under every boolean connective and for which complete inference algorithms are known. The basic description logic $\mathcal{ALC}$ is presented in the following chapters: in fact, this logic is the base for our studies on constructive description logics.

A set of descriptions for a given domain forms a *knowledge base*. A knowledge base is constituted by two components: a *TBox*, that represents the *terminological knowledge* of the domain, and an *ABox*, describing the *assertional knowledge* of the representation. In other words, the TBox contains the definitions (*axioms*) for concepts used in the descriptions of the knowledge base, while the ABox introduces information (*assertions*) about individuals in terms of membership to concepts and roles. In general, TBox axioms are composed by *concept definitions* as $C \equiv D$, describing the necessary and sufficient conditions for the membership to a concept, and *concept inclusions* as $C \sqsubseteq D$, which only define necessary membership conditions. For example, in our TBox we can state:

$$\texttt{Human} \equiv \texttt{Man} \sqcup \texttt{Woman} \qquad \texttt{Father} \sqsubseteq \texttt{Parent}$$

The ABox assertions can be simply divided in *concept assertions*, in the form $a : C$, and *role assertions*, in the form $(a, b) : R$, respectively stating that the individual denoted by a belongs to the concept C and that a and b are related by the role R.

For example, in our ABox we can assert knowledge about the individuals `adam` and `abel` as follows:

$$\texttt{adam:Father} \qquad (\texttt{adam}, \texttt{abel})\texttt{:hasChild}$$

Reasoning tasks for description logics can also be divided in tasks for the TBox and tasks for the ABox. The main reasoning problems for the TBox are *concept satisfiability*, i.e. checking if a concept can be instantiated in a given TBox, and *concept subsumption*, i.e. the problem to determine the inclusion of two complex concepts. On the other hand, the most relevant reasoning services for the ABox include *ABox consistency*, i.e. the test for satisfiability for assertions of an ABox with respect to a given TBox, and *instance checking*, i.e. the task to determine if a given individual belongs to a complex concept or a role.

1.2 Constructive description logics

Apart from the formal study of description logics languages and applications, many approaches to extend the formalism of description logics have been proposed. Part of these approaches aim at extending the expressivity of DLs, while others propose ways to use DLs and their features in relation to other formalisms and logical systems. In particular, since description logics are restrictions of the classical first order logic, their semantics corresponds to a classical reading of descriptions for concepts and individuals. Thus, some of the extensions to DLs substitute their classical semantics with non-classical interpretations, such as interpretations from fuzzy logic and extensions with modal operators [Baader *et al.*(2003b)]. As can be easily understood, these different interpretations of DL descriptions aim at modeling knowledge domains and problems that can hardly be treated in the context of a classical semantics.

In this context, it can also be interesting to study interpretations of description logics based on *constructive logics* [Troelstra(1977)]. In general, a constructive reading of DLs is useful in domains with possibly dynamic and incomplete knowledge. This view is supported by the model theoretical features of constructive semantics which allows the representation of stages of information and truth evidence. Moreover, constructive reinterpretations of description logics are significant in the context of computer science, since they allow to take advantage of the computational properties of their formulas and proofs, see e.g. [Goad(1980), Miglioli and Ornaghi(1981)]. In particular, such properties mainly come from the *Curry-Howard isomorphism* [Howard(1980)] and its relations to the *formulas-as-types* and *proofs-as-programs* paradigms: thanks to these connections, it is possible to draw from the constructive tradition of program synthesis and type systems definition also in the DLs scenario.

The idea of a combination of description logics and constructive logics, that we can call *constructive description logics*, is still quite recent: only few works have proposed constructive semantics for DLs, starting from different motivations. We give a brief account of the most relevant works of this area in the following section.

1.2.1 Current approaches and motivations

In the following, we present the main approaches to constructive description logics and we detail the motivations for their proposal.

de Paiva approach: translations of $\mathcal{ALC}$ in constructive logics

One of the first proposals to the definition of a constructive description logic is presented in [de Paiva(2005)]. The paper motivates the study of constructive description logics as useful to extend proof-theoretical methods (in particular the Curry-Howard isomorphism) to description logics. The proposal defines three constructive reinterpretations of $\mathcal{ALC}$, based on translations from $\mathcal{ALC}$ into known constructive systems. In particular, the proposal exploits the known translations of $\mathcal{ALC}$ in first order logic and in the multimodal logic K_m. By these translations, constructive semantics for $\mathcal{ALC}$ are given as translations into a fragment of first order intuitionistic logic (presented with respect to its standard Kripke semantics) and into two different systems for K_m.

Odintsov and Wansing approach: constructive paraconsistent semantics for $\mathcal{ALC}$

Other early proposals for the definition of constructive DLs are the formalizations introduced in [Odintsov and Wansing(2003), Odintsov and Wansing(2008)] for inconsistency tolerant paraconsistent description logics: the idea is that constructive semantics provide a tool to represent inconsistent data or incomplete information. In [Odintsov and Wansing(2003)] three different constructive paraconsistent semantics for $\mathcal{ALC}$ are presented: the idea is to substitute the classical semantics for the description logic with the one for N4, a four valued paraconsistent logic [Almukdad and Nelson(1984)]. This is obtained at different levels in the three translations: the first of these, $\mathcal{CALC}^C$, is mainly obtained from $\mathcal{ALC}$ by substituting subsumption with intuitionistic implication and classical negation with a constructive negation. In the semantics of $\mathcal{CALC}^C$, every element of the domain is seen as an "individual at a state" and individuals are related by an accessibility relation $\preceq$ whenever they represent the same element at a different state of knowledge: this interpretation, that can be easily compared to the usual Kripke-style semantics for intuitionistic and modal logics, leads to a constructive interpretation of implications and negations and to typical constructive properties as the disjunction property. The other two logics proposed in this paper are called $\mathcal{CALC}^{\mathsf{N4}}$ and $\mathcal{CALC}^{\mathsf{N4d}}$. The semantics of both logics is derived as translations to QN4, the first order variant of the N4 logic. $\mathcal{CALC}^{\mathsf{N4d}}$ adds duality of quantifiers with respect to the constructive negation and allows to prove a direct correspondence with the decidable modal logic **FS** [Fischer Servi(1981), Fischer Servi(1984)]: this implies the decidability of the positive fragment of this logic. Sound and complete tableaux calculi for each of the three logics are presented in [Odintsov and Wansing(2003)]. Those calculi are inspired to the usual calculi for intuitionistic and multimodal K logics and allow to solve the reasoning problems for the two logics. The calculi are

reviewed in the second paper [Odintsov and Wansing(2008)], where the authors provide a tableaux based decision procedure for $\mathcal{CALC}^{C}$, the only logic among the three which is elementary decidable, and they study its complexity.

Mendler and Scheele approach: Kripke semantics with fallible entities

Another work centered on the proposal of a constructive interpretation for DLs and the study of its applications is [Mendler and Scheele(2010)]. As in the case of the previous proposal, the motivation of this work resides in the treatment of partial information and consistency under abstraction. In particular, the authors consider the issue of evolving and dynamic knowledge, in which entities represent abstract individuals with changing properties. In such context, the static vision of classical *Open World Assumption (OWA)* can not support the idea of knowledge refinement: on the other hand, the representation of stages of information provided by constructive semantics can be used to model this idea of *Evolving OWA*. The paper proposes a constructive version of $\mathcal{ALC}$, called $c\mathcal{ALC}$, which is essentially based on a Kripke semantics. Similarly to [Odintsov and Wansing(2003)], entities are seen as states of knowledge or as partial descriptions of individuals: thus, the semantics of $c\mathcal{ALC}$ extends the classical semantics by adding a pre-order $\preceq^{\mathcal{I}}$ on the elements of the domain. Most notably, interpretations of $c\mathcal{ALC}$ also include a set $\perp^{\mathcal{I}}$ of *fallible entities*: these elements represent contradictory abstract individuals and they can express maximal elements with respect to the partial order or undefined role fillers. This semantics definition allows to prove some constructive properties of $c\mathcal{ALC}$ as the omission of the excluded middle or the distributivity of quantifiers over the disjunction. Two calculi for $c\mathcal{ALC}$ are proposed, a Hilbert-style proof system and a tableau calculus, both sound and complete with respect to the presented semantics. Finite model property for the logic and decidability of the calculi are proved via completeness of the tableau procedure. An application of $c\mathcal{ALC}$ in the context of data streams for auditing has been presented in [Mendler and Scheele(2009)].

Hofmann and Kaneiwa approaches

The works presented in [Hofmann(2005), Kaneiwa(2005)] represent two "minor" proposals in which a constructive interpretation of DLs is seen more as a formal tool than as the central theme of their study. In [Kaneiwa(2005)] different interpretations for constructive negation are introduced to model several notions of negative information. This is done by defining two different extensions to the descriptive semantics of $\mathcal{ALC}$ with different interaction between constructive and classical negation: the paper also provides a sound tableaux algorithm for satisfiability in the new semantics. A similar semantics has recently been proposed in [Kamide(2010)] for the definition of a paraconsistent version of $\mathcal{ALC}$. The proposal in [Hofmann(2005)] is centered on the study of proof-theoretical properties rather than constructive semantics. The paper introduces a polynomial-time decision procedure for subsumption derived as a variant of Gentzen sequent calculus. The procedure is proved to

be sound and complete with respect to descriptive and *fixpoint semantics* for DLs and to meet typical properties of sequent calculi as the *cut elimination* property.

$\mathcal{BCDL}$

Given the former approaches, in the current sections we resume our previous proposals in the context of constructive description logics and we motivate the work presented in the following chapters.

One of the most interesting aspects of constructive logics by the point of view of computer science, as already noted, is the possibility of giving a computational interpretation of their proofs. This idea is studied in our proposal for $\mathcal{BCDL}$ [Bozzato *et al.*(2007), Ferrari *et al.*(2010a)], a constructive reinterpretation for $\mathcal{ALC}$ based on an *information terms semantics*. The base structure of this constructive semantics are *information terms*: intuitively, an information term is a structured object that constructively justifies the validity of a formula in a classical model. This semantics can be seen as an implementation of the *BHK (Brower-Heyting-Kolmogorov)* interpretation of logical connectives and as a generalization of *realizability* interpretations. Differently from the previous works, such semantics allows to preserve the classical reading of description logic formulas. A simple proof theoretical characterization for $\mathcal{BCDL}$ by a sound and complete natural deduction calculus is presented in [Ferrari *et al.*(2010a)]. The calculus allows to prove the constructive properties of $\mathcal{BCDL}$ as the *disjunction property* and the *explicit definability property*. Such calculus provide a computational interpretation of its proofs, following the proofs-as-programs paradigm: the proved formula is seen as a goal while its proofs represent programs to solve it. Starting from the properties of this constructive semantics and the natural notion of state encoded by information terms, we have also proposed in [Bozzato *et al.*(2009a)] an initial formalization for an action language based on an information terms semantics for $\mathcal{ALC}$: even in this quite limited formalism, information terms semantics has shown its advantages by giving a way to build states obtained by actions application and to trace reasons for state inconsistency.

$\mathcal{KALC}^{\infty}$

We provided a second constructive interpretation for $\mathcal{ALC}$, that we call $\mathcal{KALC}^{\infty}$ [Bozzato *et al.*(2009b), Villa(2010)], which, however, can be seen only as partially related to $\mathcal{BCDL}$. $\mathcal{KALC}^{\infty}$ provides a Kripke-style semantics for $\mathcal{ALC}$ based on possibly infinite posets which is related to the Kripke semantics for first order intuitionistic logic. Differently from the similar semantics presented in [de Paiva(2005)], which can be seen as a direct translation in DLs of the standard intuitionistic Kripke semantics, we impose a condition on the form of the admitted models: namely, every state must be followed by a classical world, i.e. a state in which concepts are interpreted under the classical semantics for $\mathcal{ALC}$. Such condition seems to be essential to obtain the *Finite Model Property* (and thus decidability) for our logic: this semantical condition can be characterized by an axiom schema that represent the equivalent in the description logics context of the well-known *Kuroda principle* for

first order logic. In [Bozzato *et al.*(2009b)] we also presented a sound and complete tableaux calculus for $\mathcal{KALC}^{\infty}$, inspired to the ones for intuitionistic and Kuroda logic. The calculus features an efficient treatment of duplications on implicative formulas, useful for the development of an efficient decision procedure for $\mathcal{KALC}^{\infty}$ and to compare to tableaux procedures for classical description logics.

1.2.2 Motivations of the thesis

In the definition of constructive description logics suitable for effective use in applications, the essential problem regards guaranteeing their decidability. Indeed, after using constructive semantics to represent partial or incomplete information, to be able to solve and implement constructive versions of the usual description logic reasoning problems needed to perform inferences on such representations, it is necessary to provide at least decidable procedures for such problems.

Following this motivation, in order to obtain a decidable constructive DL, in this work we introduce a different Kripke-style semantics for $\mathcal{ALC}$: the logic based on this semantics, that will be the central formalism analyzed in this thesis, is called $\mathcal{KALC}$. In a nutshell, $\mathcal{KALC}$ presents a constructive semantics for $\mathcal{ALC}$ based on finite Kripke models: over this semantics, we can formulate a tableau calculus with an effective algorithm for proof search. We can show that $\mathcal{KALC}$ meets the disjunction property and, differently from other proposals as [de Paiva(2005)], $\mathcal{KALC}$ consistency agrees with the classical consistency of $\mathcal{ALC}$. Moreover, as we present by means of some examples, its Kripke semantics allows the representation of dynamic and incomplete knowledge. The idea is that a Kripke model can be considered as a set of worlds, representing states of knowledge, partially ordered by their information content. This permits to express partial and incomplete states of knowledge which can increase in time in the context of the Open World Assumption. Reasoning problems in $\mathcal{KALC}$ can be formulated exactly as in their classical counterparts for $\mathcal{ALC}$ and, as shown by such examples, they also assume a constructive meaning. Most notably, the proof search algorithm for $\mathcal{KALC}$ can be directly used to solve these reasoning problems, thus showing their decidability in our logic.

In the following chapters, after some preliminary definitions in Chapter 2 for the basic presentation of the classical version of $\mathcal{ALC}$, we introduce the syntax and semantics of $\mathcal{KALC}$ and we discuss its constructive properties and application in the representation of dynamic knowledge. Afterwards, the tableau calculus $\mathcal{T}_{\mathcal{K}}$ for $\mathcal{KALC}$ is introduced and its soundness with respect to $\mathcal{KALC}$ semantics is proved. The study of completeness and termination results give rise to the subsequent definition of the tableaux procedure associated with the calculus. The complexity properties of such procedure are discussed in the following section. In Chapter 4 we then study the logic properties of $\mathcal{KALC}$, specifically the relations of $\mathcal{KALC}$ with our previous proposals and with other intuitionistic multi-modal logics. Finally, in Chapter 5 we present an application of an information terms semantics for $\mathcal{KALC}$ in the context of Semantic Web services composition.

2

Preliminaries

In this chapter we introduce the notation and main definition for the formalisms which we refer to in the following chapters: the description logic $\mathcal{ALC}$ and the propositional and first order intuitionistic logics **Int** and **QInt**.

2.1 Description logic $\mathcal{ALC}$

We begin by introducing the language of the basic description logic $\mathcal{ALC}$ and its classical semantics [Baader *et al.*(2003a), Schmidt-Schauß and Smolka(1991)]. The formalization for $\mathcal{ALC}$ that we present is the one introduced in [Bozzato *et al.*(2007), Ferrari *et al.*(2010a)] and it forms the base for the constructive extensions that we discuss in the following chapters.

The language $\mathcal{L}$ of $\mathcal{ALC}$ is based on the following denumerable sets: the set $\mathtt{NR}$ of *role names*, the set $\mathtt{NC}$ of *concept names*, the set $\mathtt{NI}$ of *individual names*. A *concept* C is an expression of the kind:

$$C \ ::= \ A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists R.C \mid \forall R.C$$

where $A \in \mathtt{NC}$ and $R \in \mathtt{NR}$.

Let $\mathtt{VAR}$ be a denumerable set of *individual variables*; the *formulas* of $\mathcal{L}$ are defined according to the following grammar:

$$H \ ::= \ \bot \mid (s,t) : R \mid t : C \mid C$$

where $s, t \in \mathtt{NI} \cup \mathtt{VAR}$, $R \in \mathtt{NR}$ and C is a concept. We write $H \in \mathcal{L}$ to mean that H is a formula of $\mathcal{L}$ and we write $\mathcal{L}' \supseteq \mathcal{L}$ to mean that the language $\mathcal{L}'$ is an extension of $\mathcal{L}$, namely $\mathcal{L}'$ includes the same roles, concepts and individual names of $\mathcal{L}$.

An *atomic formula* of $\mathcal{L}$ is a formula of the kind $\bot$, $(s,t) : R$ or $t : A$, where $R \in \mathtt{NR}$ and $A \in \mathtt{NC}$. A *negated formula* is a formula of the kind $t : \neg C$ with C a concept. A *simple formula* is either a negated or an atomic formula. A formula is *closed* if it does not contain variables: in other words, a formula is closed if it is "grounded" with elements $s, t \in \mathtt{NI}$.

A *concept formula* is a formula of the kind $t : C$ or C, with C a concept. A *universal formula* is a formula of the kind C with C a concept: universal formulas intuitively represent axioms that hold on every element of the domain. A *role formula* is a formula of the kind $(t, s) : R$ with $R \in \mathtt{NR}$.

We define the *concept inclusion* relation (*subsumption*) $C \sqsubseteq D$ as $\neg C \sqcup D$. We remark that in the classical setting $C \rightarrow D$ is equivalent to $\neg C \sqcup D$, thus the implication constructor is usually left out from $\mathcal{ALC}$ presentation.

A *knowledge base* $\mathcal{K}$ in $\mathcal{ALC}$ is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consisting of a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$, where:

– A TBox $\mathcal{T}$ is a finite set of terminological axioms in the form $C \sqsubseteq D$ or $C \equiv D$, where C, D are concepts and $C \equiv D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$.

– An ABox $\mathcal{A}$ is a finite set of closed concept and role assertions of the kind $t : C$ or $(t, s) : R$, where $R \in \mathtt{NR}$ and $t, s \in \mathtt{NI}$.

We also distinguish two kinds of TBoxes on the base of the axioms they contain: in particular, it is usual to recognize different forms of TBoxes for complexity and decidability reasons [Baader and Nutt(2003)].

Definition 2.1
We say that a TBox $\mathcal{T}$ is acyclic *iff it holds that:*

1. *$\mathcal{T}$ only contains inclusions $A \sqsubseteq C$, with A an atomic concept.*

2. *Let us say that the atomic concept A* directly uses *A' in $\mathcal{T}$ iff, for some $A \sqsubseteq C \in \mathcal{T}$, A' is a subformula of C and let* uses *be the transitive closure of the "directly uses" relation. Then, no concept occurring in $\mathcal{T}$ uses itself.*

A TBox $\mathcal{T}$ is general *whenever $\mathcal{T}$ is not limited by the previous restrictions.*

We denote with $\mathcal{N}$ a finite subset of individual names $\mathtt{NI}$. We write $\mathcal{L}_{\mathcal{N}}$ to denote the language of the concept and role formulas in which the occurring individual names belong to $\mathcal{N}$.

A *classical model (interpretation)* $\mathcal{M}$ for $\mathcal{L}$ is a pair $(\mathcal{D}^{\mathcal{M}}, \cdot^{\mathcal{M}})$, where $\mathcal{D}^{\mathcal{M}}$ is a non-empty set (the *domain* of $\mathcal{M}$) and $\cdot^{\mathcal{M}}$ is a *valuation* map such that:

– for every $c \in \mathtt{NI}$, $c^{\mathcal{M}} \in \mathcal{D}^{\mathcal{M}}$;

– for every $A \in \mathtt{NC}$, $A^{\mathcal{M}} \subseteq \mathcal{D}^{\mathcal{M}}$;

– for every $R \in \mathtt{NR}$, $R^{\mathcal{M}} \subseteq \mathcal{D}^{\mathcal{M}} \times \mathcal{D}^{\mathcal{M}}$.

A non atomic concept C is interpreted by a subset $C^{\mathcal{M}}$ of $\mathcal{D}^{\mathcal{M}}$ as follows:

– $(\neg A)^{\mathcal{M}} = \mathcal{D}^{\mathcal{M}} \setminus A^{\mathcal{M}}$

– $(A \sqcap B)^{\mathcal{M}} = A^{\mathcal{M}} \cap B^{\mathcal{M}}$

– $(A \sqcup B)^{\mathcal{M}} = A^{\mathcal{M}} \cup B^{\mathcal{M}}$

- $(\exists R.A)^{\mathcal{M}} = \{\, d \in \mathcal{D}^{\mathcal{M}} \mid \exists d' \in \mathcal{D}^{\mathcal{M}} \text{ s.t. } (d, d') \in R^{\mathcal{M}} \text{ and } d' \in A^{\mathcal{M}} \,\}$
- $(\forall R.A)^{\mathcal{M}} = \{\, d \in \mathcal{D}^{\mathcal{M}} \mid \forall d' \in \mathcal{D}^{\mathcal{M}}, (d, d') \in R^{\mathcal{M}} \text{ implies } d' \in A^{\mathcal{M}} \,\}$

An *assignment* on a model $\mathcal{M}$ is a map $\theta : \mathtt{VAR} \to \mathcal{D}^{\mathcal{M}}$. If $t \in \mathtt{NI} \cup \mathtt{VAR}$, $t^{\mathcal{M},\theta}$ is the element of $\mathcal{D}^{\mathcal{M}}$ denoting t in $\mathcal{M}$ w.r.t. θ, namely: $t^{\mathcal{M},\theta} = \theta(t)$ if $t \in \mathtt{VAR}$, $t^{\mathcal{M},\theta} = t^{\mathcal{M}}$ if $t \in \mathtt{NI}$. A formula H is *$\mathcal{ALC}$-valid* in $\mathcal{M}$ w.r.t. θ, and we write $\mathcal{M}, \theta \models H$, if $H \neq \bot$ and one of the following conditions holds:

- $\mathcal{M}, \theta \models t : C$ iff $t^{\mathcal{M},\theta} \in C^{\mathcal{M}}$;
- $\mathcal{M}, \theta \models (s, t) : R$ iff $(s^{\mathcal{M},\theta}, t^{\mathcal{M},\theta}) \in R^{\mathcal{M}}$;
- $\mathcal{M}, \theta \models C$ iff $C^{\mathcal{M}} = \mathcal{D}^{\mathcal{M}}$.

We write $\mathcal{M} \models H$ iff $\mathcal{M}, \theta \models H$ for every assignment θ. Note that, given a concept C of $\mathcal{L}$, $\mathcal{M} \models C$ iff $\mathcal{M} \models x : C$, with x any variable. If Γ is a set of formulas, $\mathcal{M} \models \Gamma$ means that $\mathcal{M} \models H$ for every $H \in \Gamma$. We say that H is an *$\mathcal{ALC}$-logical consequence* of Γ, and we write $\Gamma \models H$, iff, for every $\mathcal{M}$ and every θ, $\mathcal{M}, \theta \models \Gamma$ implies $\mathcal{M}, \theta \models H$.

With respect to a TBox $\mathcal{T}$, we say that a model $\mathcal{M}$ satisfies an axiom if:

$$\mathcal{M} \models C \sqsubseteq D \text{ iff } C^{\mathcal{M}} \subseteq D^{\mathcal{M}} \qquad \mathcal{M} \models C \equiv D \text{ iff } C^{\mathcal{M}} = D^{\mathcal{M}}$$

With respect to an ABox $\mathcal{A}$, we say that a model $\mathcal{M}$ satisfies an assertion if:

$$\mathcal{M} \models a : C \text{ iff } a^{\mathcal{I}} \in C^{\mathcal{M}} \qquad \mathcal{M} \models (a, b) : R \text{ iff } (a^{\mathcal{M}}, b^{\mathcal{M}}) \in R^{\mathcal{M}}$$

A model $\mathcal{M}$ satisfies a TBox $\mathcal{T}$ iff $\mathcal{M} \models H$ for every $H \in \mathcal{T}$. Similarly, $\mathcal{M}$ satisfies an ABox $\mathcal{A}$ iff $\mathcal{M} \models H$ for every $H \in \mathcal{A}$. We also say that $\mathcal{M}$ satisfies a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ iff $\mathcal{M}$ satisfies $\mathcal{T} \cup \mathcal{A}$.

Over this formalization for $\mathcal{ALC}$, we can state some of the typical problems on description logics. Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

- *Concept satisfiability:* a concept C is satisfiable w.r.t. $\mathcal{K}$ iff there exists a model $\mathcal{M}$ of $\mathcal{K}$ such that $C^{\mathcal{M}} \neq \emptyset$;
- *Concept subsumption:* given two concepts C and D, D subsumes C w.r.t. $\mathcal{K}$ iff $\mathcal{K} \models C \sqsubseteq D$;
- *Instance checking:* given an individual name t and a concept C, the individual t is an instance of the concept C w.r.t. $\mathcal{K}$ iff $\mathcal{K} \models t : C$.

2.2 Propositional intuitionistic logic Int

In this section, we introduce the syntax and Kripke semantics for the propositional intuitionistic logic **Int**.

We consider the propositional language $\mathcal{PL}$ based on a denumerable set of *propositional variables* $\mathcal{PV}$, the logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$ and the logical constants $\top$ and $\bot$. The *formulas* of propositional intuitionistic logic are expression of the form:

$$F ::= \bot \mid \top \mid p \mid \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F$$

where $p \in \mathcal{PV}$ is a propositional variable. Writing formulas we assume that $\neg$ binds stronger than $\wedge$ and $\vee$, which in turn are stronger than $\rightarrow$.

In the following we recall the main definitions about *Kripke semantics* for **Int**: we refer to [Chagrov and Zakharyaschev(1997)] for further details. Kripke semantics is based on partial orders: intuitively, the elements of the ordering, also called *worlds*, can be seen as states of knowledge. An interpretation is assigned to each world, i.e., every world justifies a subset of the closed atomic formulas of the language. The knowledge acquired in each world is monotonic, in the sense that if a closed atomic formula is known to be true at a world α, it preserves its truth in all the worlds that follow α in the partial order. Thus, moving from one world to one of its successors, our knowledge on atomic facts can only increase.

Formally, a *partially ordered set (poset)* is a pair $(P, \leq)$, where:

- P is a non-empty set (possibly infinite);
- $\leq \subseteq P \times P$ is a *partial order* relation over P, that is a reflexive, transitive and antisymmetric relation.

If the poset contains a minimum ρ, we call $(P, \leq)$ a poset with *root* ρ.

The interpretation of propositional intuitionistic logic given by *Kripke models* can be defined as follows:

Definition 2.2 (Propositional intuitionistic Kripke model)
A propositional intuitionistic Kripke model is a quadruple

$$\underline{K} = \langle P, \leq, \rho, \Vdash \rangle,$$

where $(P, \leq)$ is a poset with root ρ and $\Vdash$ is a binary relation called forcing relation *between elements of P and formulas of $\mathcal{PL}$, satisfying the following properties:*

- *for every $\alpha, \beta \in P$ and $p \in \mathcal{PV}$, $\alpha \Vdash p$ and $\alpha \leq \beta$ imply $\beta \Vdash p$;*
- *$\alpha \Vdash \top$;*
- *$\alpha \Vdash \bot$ does not hold;*
- *$\alpha \Vdash F_1 \wedge F_2$ iff $\alpha \Vdash F_1$ and $\alpha \Vdash F_2$;*
- *$\alpha \Vdash F_1 \vee F_2$ iff $\alpha \Vdash F_1$ or $\alpha \Vdash F_2$;*
- *$\alpha \Vdash F_1 \rightarrow F_2$ iff for every $\beta \in P$ such that $\alpha \leq \beta$, $\beta \Vdash F_1$ implies that $\beta \Vdash F_2$;*
- *$\alpha \Vdash \neg F$ iff for every $\beta \in P$ such that $\alpha \leq \beta$, $\beta \Vdash F$ does not hold.*

The relation $\alpha \Vdash F$ may be read "F is true at a world α" or, more concisely, "α forces F". We denote with $\alpha \nVdash F$ the fact that $\alpha \Vdash F$ does not hold.

A formula F *is valid in a model* $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$, denoted $\underline{K} \Vdash F$, iff $\alpha \Vdash F$ for every $\alpha \in P$ (or equivalently if $\rho \Vdash A$). We say that a formula F is **Int**-*valid* iff it is valid in every Kripke model for **Int**. Let $\mathrm{Mod}_{\mathbf{Int}}$ be the class of all propositional Kripke models, then we define:

$$\mathbf{Int} = \{ F \mid F \text{ is a formula of } \mathcal{PL} \text{ and, for every } \underline{K} \in \mathrm{Mod}_{\mathbf{Int}}, \underline{K} \Vdash F \}$$

In other words, we identify the logic **Int** as the set of formulas valid in all Kripke models.

A relevant property of Kripke models is the *monotonicity property*: if a given formula F is forced at a world α of a Kripke model $\underline{K}$, then it is forced in all the worlds $\beta \in P$ such that $\alpha \leq \beta$. This means that all of the knowledge is preserved between subsequent worlds and it can only increase monotonically.

Proposition 2.3 (Monotonicity property)
Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a propositional intuitionistic Kripke model, $\alpha \in P$ and F be a formula. If $\alpha \Vdash F$ in $\underline{K}$, then $\beta \Vdash F$ in K for every $\beta \in P$ such that $\alpha \leq \beta$. □

The proposition can be easily proved by induction on the structure of the formula F and the inductive definition of the forcing relation.

A typical constructive property of **Int** semantics is the *Disjunction Property (DP)*, which can be stated as follows:

Proposition 2.4 (Disjunction Property)
If $F_1 \vee F_2 \in \mathbf{Int}$, then either $F_1 \in \mathbf{Int}$ or $F_2 \in \mathbf{Int}$. □

In other words, whenever a disjunctive formula $F_1 \vee F_2$ is valid in **Int**, then we can prove that at least one between F_1 and F_2 is valid in **Int**.

Given the definitions for **Int** semantics, we can now provide a *Hilbert-style calculus* for intuitionistic propositional logic. A detailed description of Hilbert-style calculi can be found in [Kleene(1952)]. The Hilbert-style calculus $\mathcal{H}_{\mathbf{Int}}$ for propositional intuitionistic logic consists of the axioms and rules listed in Figure 2.1. We assume the usual definitions and conventions for Hilbert-style calculi [Kleene(1952)]: in particular, a *proof* of $\mathcal{H}_{\mathbf{Int}}$ is a finite sequence of formulas where every formula is either an axiom or it is obtained by a rule application on previous formulas of the sequence.

It is well known that $\mathcal{H}_{\mathbf{Int}}$ is sound and complete with respect to **Int** semantics: in other words, the set of formulas provable in $\mathcal{H}_{\mathbf{Int}}$ coincides with **Int**, the set of all **Int**-valid formulas [Chagrov and Zakharyaschev(1997)].

2.3 First order intuitionistic logic QInt

We can now extend the definitions for intuitionistic logic and its Kripke semantics to the first order case [Smorynski(1973)]. We call **QInt** the resulting first order intuitionistic logic.

Axioms for conjunction:

$$\begin{aligned} Ax1: &\quad A \sqcap B \to A \\ Ax2: &\quad A \sqcap B \to B \\ Ax3: &\quad A \to (B \to (A \sqcap B)) \end{aligned}$$

Axioms for disjunction:

$$\begin{aligned} Ax4: &\quad A \to A \sqcup B \\ Ax5: &\quad B \to A \sqcup B \\ Ax6: &\quad (A \to C) \to ((B \to C) \to (A \sqcup B \to C)) \end{aligned}$$

Axioms for implication:

$$\begin{aligned} Ax7: &\quad A \to (B \to A) \\ Ax8: &\quad (A \to (B \to C)) \to ((A \to B) \to (A \to C)) \end{aligned}$$

Axiom for intuitionistic contradiction:

$$\begin{aligned} Ax9: &\quad \bot \to A \\ Ax10: &\quad A \sqcap \neg A \to \bot \end{aligned}$$

Rules:

$$\frac{A \qquad A \to B}{B}\ \text{MP}$$

Figure 2.1: Axioms and rules of $\mathcal{H}_{\mathbf{Int}}$

A *signature* $\Sigma = \langle \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ consists of the denumerable sets of *constant symbols* $\mathcal{C}$, *function symbols* $\mathcal{F}$ and *predicate symbols* $\mathcal{R}$. The intersection of these elements must be empty and we consider predicates in $\mathcal{R}$ or function symbols in $\mathcal{F}$ to have arity greater than zero.

The language of first order intuitionistic logic $\mathcal{FL}$ is built starting from a denumerable set of variables $\mathcal{V}$, the logical constants $\top$ and $\bot$, the logical connectives $\wedge$, $\vee$, $\neg$ and $\to$, the quantifiers $\exists$ and $\forall$ and the auxiliary symbols '(' and ')'. A *term* is defined as:

– every variable $x \in \mathcal{V}$ is a term;

– every constant symbol $c \in \mathcal{C}$ is a term;

– if $f \in \mathcal{F}$ is a function symbol with arity n and $t_1, \ldots, t_n$ are terms, then also $f(t_1, \ldots, t_n)$ is a term.

First order formulas are defined according to the following syntax:

$$F ::= \bot \mid \top \mid p(t_1, \ldots, t_n) \mid \\ \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid \exists x.F(x) \mid \forall x.F(x)$$

where $t_1, \ldots, t_n$ are terms, $p \in \mathcal{R}$ is a predicate symbol and $x \in \mathcal{V}$ is a variable. Formulas of the kind $p(t_1, \ldots, t_n)$ with $p \in \mathcal{R}$ and $t_1, \ldots, t_n$ terms are called *atomic formulas*. A *substitution* σ is any function from variables to terms. We denote with $t\sigma$ and $F\sigma$ respectively, the term and the formula obtained by simultaneously replacing every free occurrence of a variable x with $\sigma(x)$. We assume the usual considerations on substitutions in the first order context, see e.g. [Kleene(1952)] for a detailed discussion. A term is *closed* if it has no free variables. Analogously, a formula is *closed* if it does not contain free variables. We say that σ is a *closing substitution for a tuple* $t_1, \ldots, t_n$ of terms if $t_1\sigma, \ldots, t_n\sigma$ is a tuple of closed terms. Analogously, σ is a *closing substitution* for a formula F if $F\sigma$ is a closed formula.

Kripke models for **QInt** have to be extended from the propositional case to treat the interpretation of first order formulas (and in particular the interpretation of terms) in every state of the poset.

Definition 2.5 (First order intuitionistic model)
A first order intuitionistic Kripke model is a quintuple

$$\underline{K} = \langle P, \leq, \rho, \mathcal{D}, \Vdash \rangle$$

where:

- $(P, \leq)$ *is a poset with root* ρ*;*
- $\mathcal{D}$ *is a function associating to every element* $\alpha \in P$ *a non empty set* $\mathcal{D}(\alpha)$ *such that* $\alpha, \beta \in P$ *and* $\alpha \leq \beta$ *implies* $\mathcal{D}(\alpha) \subseteq \mathcal{D}(\beta)$*;*
- $\Vdash$ *is a relation called* forcing relation *between elements in* P *and closed formulas of* $\mathcal{FL}$ *that satisfies the following definition:*
  - *for* $p(t_1, \ldots, t_n)$ *and for every* $\beta \in P$ *such that* $\alpha \leq \beta$ *and* $d_1, \ldots, d_n \in \mathcal{D}(\alpha)$*, if* $\alpha \Vdash p(d_1, \ldots, d_n)$*, then* $\beta \Vdash p(d_1, \ldots, d_n)$*;*
  - $\alpha \Vdash F_1 \wedge F_2$ *iff* $\alpha \Vdash F_1$ *and* $\alpha \Vdash F_2$*;*
  - $\alpha \Vdash F_1 \vee F_2$ *iff* $\alpha \Vdash F_1$ *or* $\alpha \Vdash F_2$*;*
  - $\alpha \Vdash F_1 \rightarrow F_2$ *iff for every* $\beta \in P$ *such that* $\alpha \leq \beta$*,* $\beta \Vdash F_1$ *implies that* $\beta \Vdash F_2$*;*
  - $\alpha \Vdash \neg F$ *iff for every* $\beta \in P$ *such that* $\alpha \leq \beta$*,* $\beta \nVdash F$*;*
  - $\alpha \Vdash \exists x.F(x)$ *iff there exists* $a \in \mathcal{D}(\alpha)$ *such that* $\alpha \Vdash F(a)$*;*
  - $\alpha \Vdash \forall x.F(x)$ *iff for every* $\beta \in P$ *such that* $\alpha \leq \beta$ *and for every* $b \in \mathcal{D}(\beta)$*,* $\beta \Vdash F(b)$*.*

Axioms for existential quantifier:

$$\begin{array}{rl} Ax11: & A(t/x) \rightarrow \exists x.A(x) \\ {}^{*}Ax12: & \forall x.(A(x) \rightarrow B) \rightarrow (\exists x.A(x) \rightarrow B) \end{array}$$

Axioms for universal quantifier:

$$\begin{array}{rl} Ax13: & \forall x.A(x) \rightarrow A(t/x) \\ {}^{*}Ax14: & \forall x.(B \rightarrow A(x)) \rightarrow (B \rightarrow \forall x.A(x)) \end{array}$$

Rules:

$$\frac{A(x)}{\forall x.A(x)}\ \text{GEN}$$

* x does not appear in the free variables of B

Figure 2.2: Axioms and rules of $\mathcal{H}_{\mathbf{QInt}}$ for quantifiers

Given a model $\underline{K} = \langle P, \leq, \rho, \mathcal{D}, \Vdash \rangle$, $\alpha \in P$ and an open formula F of $\mathcal{FL}$, $\alpha \Vdash F$ iff, for every $\beta \in P$ such that $\alpha \leq \beta$ and for every closed substitution σ, $\beta \Vdash F\sigma$. As in the propositional case, a formula F is *valid in the model* $\underline{K}$, and we write $\underline{K} \Vdash F$, iff $\alpha \Vdash F$ for every $\alpha \in P$ (or $\rho \Vdash F$), while F is **QInt**-*valid* iff it is valid in every model for **QInt**. We identify the logic **QInt** with the set of all of its valid formulas. Formally, let $\text{Mod}_{\mathbf{QInt}}$ be the class of all first order Kripke models, then we can state:

$$\mathbf{QInt} = \{\, F \mid F \text{ is a formula of } \mathcal{FL} \text{ and, for every } \underline{K} \in \text{Mod}_{\mathbf{QInt}}, \underline{K} \Vdash F \,\}$$

By the semantics definition, the monotonicity property is naturally preserved in the first order case.

Proposition 2.6 (Monotonicity property)
Let $\underline{K} = \langle P, \leq, \rho, \mathcal{D}, \Vdash \rangle$ be a first order intuitionistic Kripke model, $\alpha \in P$ and F be a closed formula. If $\alpha \Vdash F$ in $\underline{K}$, then $\beta \Vdash F$ in $\underline{K}$ for every $\beta \in P$ such that $\alpha \leq \beta$. □

We can also prove that **QInt** meets the Disjunction Property: moreover, the first order intuitionistic logic meets another constructive property known as *Explicit Definability Property (ED)*.

Proposition 2.7 (Disjunction Property)
*If $F_1 \vee F_2 \in$ **QInt**, then either $F_1 \in$ **QInt** or $F_2 \in$ **QInt**.* □

Proposition 2.8 (Explicit Definability Property)
*If $\exists x.F(x) \in$ **QInt**, then there exists some term t for which $F(t) \in$ **QInt**.* □

Intuitively the explicit definability property states that, whenever $\exists x.F(x)$ is a valid formula of **QInt**, then we can always find a term t verifying the validity of $F(t)$.

We can now extend the Hilbert-style calculus $\mathcal{H}_{\mathbf{Int}}$ presented for propositional intuitionistic logic to the first order formalization of **QInt**. We call $\mathcal{H}_{\mathbf{QInt}}$ the resulting calculus: the calculus $\mathcal{H}_{\mathbf{QInt}}$ is obtained by adding the axioms and rules listed in Figure 2.2, necessary for the treatment of quantifiers, to the ones already shown in Figure 2.1. As in the propositional case, $\mathcal{H}_{\mathbf{QInt}}$ is complete with respect to **QInt**: the set of formulas provable in $\mathcal{H}_{\mathbf{QInt}}$ coincides with the set of all **QInt**-valid formulas [Smorynski(1973)].

3

$\mathcal{KALC}$

In this chapter we introduce the constructive description logic $\mathcal{KALC}$. This logic is based on the language of $\mathcal{ALC}$ and relies on a Kripke-style semantics which corresponds to a reformulation of the Kripke semantics for first-order intuitionistic logic. We provide a sound, complete and terminating tableau calculus for $\mathcal{KALC}$ which allow us to state the decidability of the usual reasoning problems for our constructive semantics. In the following sections we firstly present the syntax and semantics of $\mathcal{KALC}$ and we discuss the properties of our constructive interpretation. Over this foundation, we then introduce the tableau calculus $\mathcal{T}_{\mathcal{K}}$ for $\mathcal{KALC}$ and we prove its soundness: completeness is a consequence of the termination proof that we give in the following section. The chapter concludes with a discussion on the complexity of the tableau procedure derived by the calculus. Some of the results introduced in this chapter are presented in [Bozzato *et al.*(2010)].

3.1 $\mathcal{KALC}$ syntax and semantics

We introduce the language $\mathcal{L}$ of $\mathcal{KALC}$. It coincides with the language of $\mathcal{ALC}$ given in the Preliminaries minus some modifications. In particular, concept definitions include implication as a concept constructor while the falsum is now represented as a concept. Finally, subsumptions are defined as formulas of $\mathcal{L}$.

Formally, concepts C,D and formulas H of $\mathcal{L}$ are defined as follows:

$$\begin{array}{rcl} C, D & ::= & \bot \mid A \mid C \sqcap D \mid C \sqcup D \mid C \rightarrow D \mid \exists R.C \mid \forall R.C \\ H & ::= & (c,d) : R \mid c : C \mid C \sqsubseteq D \end{array}$$

where $c, d \in \mathtt{NI}$, $A \in \mathtt{NC}$ and $R \in \mathtt{NR}$. Note that $\neg C$ is not defined as a constructor of the language, but, as usual in non-classical logics, we write $\neg C$ as an abbreviation for $C \rightarrow \bot$.

Given a finite $\mathcal{N} \subseteq \mathtt{NI}$, we write $\mathcal{L}_{\mathcal{N}}$ to denote the language of formulas of $\mathcal{L}$ in which the occurring individual names belong to $\mathcal{N}$.

A *$\mathcal{KALC}$-model* for $\mathcal{L}_\mathcal{N}$ is a quadruple

$$\underline{K} = \langle P, \leq, \rho, \iota \rangle$$

where:

– $(P, \leq)$ is a finite poset with minimum element ρ;

– ι is a function associating with every $\alpha \in P$ a model $\iota(\alpha) = (\mathcal{D}^\alpha, \cdot^\alpha)$ for $\mathcal{L}_\mathcal{N}$ such that, for every $\alpha \leq \beta$, the following holds:

(K1) $\mathcal{D}^\alpha \subseteq \mathcal{D}^\beta$;

(K2) for every $c \in \mathcal{N}$, $c^\alpha = c^\beta$;

(K3) for every $A \in \texttt{NC}$, $A^\alpha \subseteq A^\beta$;

(K4) for every $R \in \texttt{NR}$, $R^\alpha \subseteq R^\beta$.

In the following we refer to elements α of P as *worlds* of $\underline{K}$. Intuitively, consistently with the usual reading of Kripke models, worlds of $\underline{K}$ represent states of knowledge that can be updated or refined by the $\leq$ relation: we note that conditions (K1)–(K4) settle that the knowledge is monotonic. Moreover, we note that the set P of such worlds is finite by definition.

Given $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ and $\alpha \in P$, we denote with $\mathcal{L}_\alpha$ the language obtained by adding to the individual names of $\mathcal{L}_\mathcal{N}$ every element $d \in \mathcal{D}^\alpha$ and setting $d^\alpha = d$. We assume that $\mathcal{N} \cap \mathcal{D}^\alpha = \emptyset$ for every $\alpha \in P$. Note that, by Condition (K1), $\alpha \leq \beta$ implies $\mathcal{L}_\alpha \subseteq \mathcal{L}_\beta$. We remark that such introduction of a language for every world of a $\mathcal{KALC}$-model is just a technical machinery needed for the treatment of quantifiers: in fact, this is simply a way to directly name individuals of the domain in the syntax of formulas.

Let α be a world of $\underline{K}$ and H a formula of $\mathcal{L}_\alpha$; we inductively define the *forcing relation* $\alpha \Vdash H$ as follows:

– $\alpha \Vdash c : A$, where $A \in \texttt{NC}$ or $A = \bot$, iff $c^\alpha \in A^\alpha$;

– $\alpha \Vdash (c, d) : R$ where $R \in \texttt{NR}$, iff $(c^\alpha, d^\alpha) \in R^\alpha$;

– $\alpha \Vdash c : C \sqcap D$ iff $\alpha \Vdash c : C$ and $\alpha \Vdash c : D$;

– $\alpha \Vdash c : C \sqcup D$ iff $\alpha \Vdash c : C$ or $\alpha \Vdash c : D$;

– $\alpha \Vdash c : C \rightarrow D$ iff, for every $\beta \geq \alpha$, $\beta \Vdash c : C$ implies $\beta \Vdash c : D$;

– $\alpha \Vdash c : \exists R.C$ iff there is $d \in \mathcal{D}^\alpha$ such that $\alpha \Vdash (c, d) : R$ and $\alpha \Vdash d : C$;

– $\alpha \Vdash c : \forall R.C$ iff, for every $\beta \geq \alpha$ and $d \in \mathcal{D}^\beta$, $\beta \Vdash (c, d) : R$ implies $\beta \Vdash d : C$;

– $\alpha \Vdash C \sqsubseteq D$ iff, for every $\beta \geq \alpha$ and $c \in \mathcal{D}^\beta$, $\beta \Vdash c : C$ implies $\beta \Vdash c : D$.

We remark that, differently from $\mathcal{ALC}$, the logical connectives are not interdefinable; e.g., $C \to D$ is not equivalent to $\neg C \sqcup D$. As for negation, being $\neg C$ an abbreviation for $C \to \bot$, we get

$$\alpha \Vdash c : \neg C \quad \text{iff} \quad \text{for every } \beta \geq \alpha, \beta \nVdash c : C$$

By conditions (K1)–(K4) the forcing relation satisfies the monotonicity property, which explains the monotonic increase of knowledge along the $\leq$ relation between worlds:

Proposition 3.1 (Monotonicity property)
Let $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ be a $\mathcal{KALC}$-model, $\alpha \in P$ and H be a formula of $\mathcal{L}_\alpha$. Then, $\alpha \Vdash H$ implies $\beta \Vdash H$ for every $\beta \geq \alpha$.

Proof: The assertion follows by induction on the structure of the formula H. We only discuss some of the most relevant cases, while the other cases directly follow by the induction hypothesis.

If $H = c : A$ with $A = \bot$ or $A \in \mathtt{NC}$, the assertion directly follows from condition (K3) on the definition of $\underline{K}$. Moreover, by condition (K4), the same holds if $H = (c, d) : R$ where $R \in \mathtt{NR}$.

If $H = c : C \to D$, suppose that $\alpha \Vdash H$ but there exists a $\beta \geq \alpha$ such that $\beta \nVdash H$. This means that there exists a $\gamma \geq \beta$ such that $\gamma \Vdash c : C$ but $\gamma \nVdash c : D$. This is an absurd, since by definition and by the fact that $\gamma \geq \alpha$ this would imply that $\alpha \nVdash H$ against our assumptions.

If $H = C \sqsubseteq D$, suppose that $\alpha \Vdash H$ but there exists a $\beta \geq \alpha$ such that $\beta \nVdash H$. This means that there exists a $\gamma \geq \beta$ and $c \in \mathcal{D}^\gamma$ such that $\gamma \Vdash c : C$ but $\gamma \nVdash c : D$. This is an absurd, since by definition and by the fact that $\gamma \geq \alpha$ it would hold that that $\alpha \nVdash H$ against our assumptions. □

Moreover, we can show the relation between the interpretation of implication and subsumption by the following property.

Proposition 3.2
Let $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ be a $\mathcal{KALC}$-model and $\alpha \in P$. Then, $\alpha \Vdash C \sqsubseteq D$ iff for every $\beta \geq \alpha$ and $c \in \mathcal{D}^\beta$, $\beta \Vdash c : C \to D$.

Proof: Let us suppose that there exists $\beta \geq \alpha$ and $c \in \mathcal{D}^\beta$ such that $\beta \nVdash c : C \to D$. Thus, there must exist a $\gamma \geq \beta$ such that $\gamma \Vdash c : C$ and $\gamma \nVdash c : D$. Thus we have that $\alpha \nVdash C \sqsubseteq D$.

On the other hand, let us suppose that $\alpha \nVdash C \sqsubseteq D$. Then there exists a $\gamma \geq \alpha$ and $d \in \mathcal{D}^\gamma$ such that $\gamma \Vdash d : C$ but $\gamma \nVdash d : D$. That is, $\gamma \nVdash d : C \to D$ and the assertion is proved. □

Given a $\mathcal{KALC}$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ for $\mathcal{L}_\mathcal{N}$, we say that a formula H of $\mathcal{L}_\mathcal{N}$ is *valid* in $\underline{K}$, and we write $\underline{K} \Vdash H$ if $\alpha \Vdash H$ for every $\alpha \in P$ (or equivalently if $\rho \Vdash H$). A formula H is *$\mathcal{KALC}$-valid* if H is valid in every $\mathcal{KALC}$-model for $\mathcal{L}_\mathcal{N}$. Finally, H

is *satisfiable* in $\mathcal{KALC}$ iff there exists a $\mathcal{KALC}$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ and $\alpha \in P$ such that $\alpha \Vdash H$ in $\underline{K}$.

A *final* world ϕ of $\underline{K}$ is a maximal element of $(P, \leq)$: that is, for every $\alpha \in P$, $\phi \leq \alpha$ implies $\phi = \alpha$. Note that $\phi \nVdash H$ implies $\phi \Vdash \neg H$. As a consequence, in ϕ any formula $c : C \sqcup \neg C$ is valid, as in classical models for $\mathcal{ALC}$, hence a final world represents a state of complete knowledge.

In particular, we can define the relation between classical models for $\mathcal{ALC}$ and final states of $\mathcal{KALC}$ models by the following considerations. Given an $\mathcal{ALC}$-model $\mathcal{M}$ for $\mathcal{L}$, let us consider the $\mathcal{KALC}$-model

$$\underline{K}_{\mathcal{M}} = \langle \{\rho\}, \{(\rho, \rho)\}, \rho, \iota \rangle$$

where $\iota(\rho) = \mathcal{M}$. It is easy to check that $\underline{K}_{\mathcal{M}}$ is equivalent to $\mathcal{M}$ in the following sense: for every closed formula $H \in \mathcal{L}$

$$\mathcal{M} \models H \text{ iff } \rho \Vdash H \text{ in } \underline{K}_{\mathcal{M}}$$

Hence, if H does not hold in an $\mathcal{ALC}$-model $\mathcal{M}$, $\underline{K}_{\mathcal{M}}$ is a countermodel for H. Thus, we get that the set of formulas of $\mathcal{L}_{\mathcal{N}}$, valid in every $\mathcal{KALC}$-model is a subset of the $\mathcal{ALC}$-valid formulas. In this sense, $\mathcal{KALC}$ is classically consistent.

On the other hand, the final elements of a $\mathcal{KALC}$-model essentially coincide with classical interpretations. Indeed, given a $\mathcal{KALC}$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ and a final element $\phi \in P$, let $\mathcal{M}_\phi = \iota(\phi)$. It is easy to check that, for every closed formula H of $\mathcal{L}_\phi$

$$\mathcal{M}_\phi \models H \text{ iff } \phi \Vdash H \text{ in } \underline{K}$$

The following proposition also holds:

Proposition 3.3
If $c : C \in \mathcal{L}$ is valid in $\mathcal{ALC}$, then $c : \neg\neg C$ is valid in $\mathcal{KALC}$.

Proof: Let us suppose that $c : \neg\neg C$ is not valid in $\mathcal{KALC}$. Then there exists a $\mathcal{KALC}$-model $\underline{K}' = \langle P, \leq, \rho, \iota \rangle$ and a world $\alpha \in P$ such that $\alpha \nVdash c : \neg\neg C$. By semantics definition, there exists $\beta \in P$ such that $\beta \geq \alpha$ and $\beta \Vdash c : \neg C$. This implies that, for every $\gamma \in P$ with $\gamma \geq \beta$, $\gamma \nVdash c : C$. Since every $\mathcal{KALC}$-model is finite, there exists a final world $\phi \in P$ such that $\phi \geq \beta$ and $\phi \nVdash c : C$. If we define, as above, $\mathcal{M}_\phi = \iota(\phi)$, it holds that $\mathcal{M}_\phi \not\models c : C$ which implies that $c : C$ is not valid in $\mathcal{ALC}$. □

We can also state the following result with respect to satisfiability:

Proposition 3.4
A formula $H \in \mathcal{L}_{\mathcal{N}}$ is satisfiable in $\mathcal{KALC}$ iff H is satisfiable in $\mathcal{ALC}$.

Proof: Given H satisfiable in $\mathcal{KALC}$, then there exists $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ and a world $\alpha \in P$ such that $\alpha \Vdash H$ in $\underline{K}$. Since P is finite, there exists a final world $\phi \geq \alpha$ such that $\phi \Vdash H$ in $\underline{K}$. If we define as above the classical model $\mathcal{M}_\phi = \iota(\phi)$, we have that $\mathcal{M}_\phi \models H$, meaning that H is satisfiable in $\mathcal{ALC}$.

On the other hand, if H is satisfiable in $\mathcal{ALC}$ then there exists an $\mathcal{ALC}$-model $\mathcal{M}$ such that $\mathcal{M} \models H$. Defining, as above, the $\mathcal{KALC}$-model $\underline{K}_{\mathcal{M}} = \langle \{\rho\}, \{(\rho, \rho)\}, \rho, \iota \rangle$ with $\iota(\rho) = \mathcal{M}$, we immediately obtain that $\rho \Vdash H$, which shows that H is satisfiable in $\mathcal{KALC}$. □

We now introduce the notion of *$\mathcal{KALC}$-logical consequence*, denoted by $\overset{\mathrm{k}}{\models}$, which allows us to represent in $\mathcal{KALC}$ the usual inference problems for DLs. Let $\mathcal{F}$ be a set of formulas, by $\alpha \Vdash \mathcal{F}$ we mean that $\alpha \Vdash H$ for every $H \in \mathcal{F}$. Given a formula H, the relation $\mathcal{F} \overset{\mathrm{k}}{\models} H$ holds iff for every $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ and $\alpha \in P$, if $\alpha \Vdash \mathcal{F}$ then $\alpha \Vdash H$. By the above discussion, it follows that if $\mathcal{F} \overset{\mathrm{k}}{\models} H$ then H is a logical consequence of $\mathcal{F}$ as understood in $\mathcal{ALC}$. Thus, $\mathcal{KALC}$-logical consequence refines the corresponding notion for $\mathcal{ALC}$.

In the context of $\mathcal{KALC}$, we assume that TBoxes are only composed of concept inclusions of the form $C \sqsubseteq D$. The inference problems for $\mathcal{KALC}$ are formulated as given for $\mathcal{ALC}$ by referring to the $\overset{\mathrm{k}}{\models}$ relation. Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

– *Concept satisfiability*: C is satisfiable w.r.t. $\mathcal{K}$ iff $\mathcal{K} \cup \{q : C\} \not\overset{\mathrm{k}}{\models} q : \bot$, with q not occurring in $\mathcal{A}$.

– *Concept subsumption*: D subsumes C w.r.t. $\mathcal{K}$ iff $\mathcal{K} \overset{\mathrm{k}}{\models} C \sqsubseteq D$.

– *Instance checking*: $c : C$ is entailed by $\mathcal{K}$ iff $\mathcal{K} \overset{\mathrm{k}}{\models} c : C$.

In the next example we show how our semantics allows us to represent partial and incomplete information and supports constructive reasoning.

Example 1 (Auditing)
We reconsider the auditing example of [Mendler and Scheele(2010)]. Let us consider the knowledge base defined by the following ABox $\mathcal{A}$:

`a:Company`	`(a,b):hasCustomer`	`b:Insolvent`
`b:Company`	`(a,c):hasCustomer`	`d` : ¬`Insolvent`
`c:Company`	`(b,c):hasCustomer`	`a` : $D \rightarrow$ `CW`
`d:Company`	`(c,d):hasCustomer`	

where D is the concept:

$$\exists \texttt{hasCustomer}.(\texttt{Insolvent} \sqcap \exists \texttt{hasCustomer}.\neg \texttt{Insolvent})$$

In $\mathcal{A}$, the concept `CW` stands for "Credit Worthy" and the formula `a` : $D \rightarrow$ `CW` states that if the company `a` has an insolvent customer *ins* which in turn can rely on at least one non-insolvent customer *nins*, then `a` can be trusted as `CW`. Here and in Figure 3.1, I abbreviates `Insolvent`.

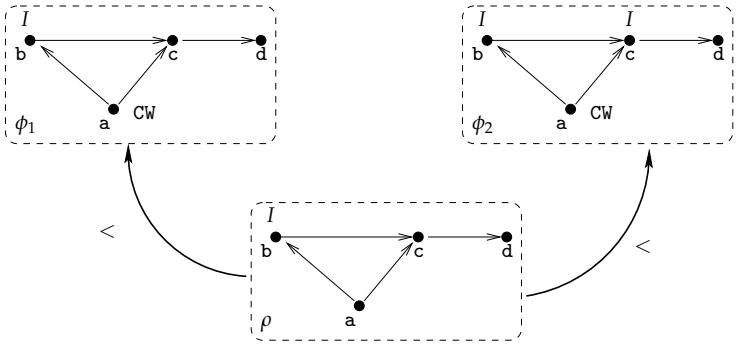

Figure 3.1: The model $\underline{K}$

In $\mathcal{ALC}$ the assertion $\mathtt{a}:\mathtt{CW}$ is entailed by $\mathcal{A}$. To prove this, let $\mathcal{M}$ be any model of $\mathcal{A}$. We first show that $\mathtt{a} : D$ holds in $\mathcal{M}$. Since *tertium non datur* classically holds, in $\mathcal{M}$ the customer $\mathtt{c}$ is either insolvent or not insolvent. Let us consider the worlds ϕ_2 and ϕ_1 in Figure 3.1 representing the two possibilities; in both cases, we can find out the clients *ins* and *nins* required by D: $ins = \mathtt{c}$ and $nins = \mathtt{d}$ in the former case, $ins = \mathtt{b}$ and $nins = \mathtt{c}$ in the latter. Since $\mathtt{a} : D \rightarrow \mathtt{CW}$ holds in $\mathcal{M}$, it follows that $\mathtt{a} : \mathtt{CW}$ holds in $\mathcal{M}$. Thus, in $\mathcal{ALC}$ the company $\mathtt{a}$ is trusted as $\mathtt{CW}$ though we have not any knowledge about the identity of the customers *ins* and *nins*.

On the contrary, in $\mathcal{KALC}$ the information in $\mathcal{A}$ does not enable to assert that $\mathtt{a}$ is $\mathtt{CW}$. The point is that we have not enough knowledge on $\mathtt{c}$, thus neither "$\mathtt{c}$ is insolvent" nor "$\mathtt{c}$ is not insolvent" can be asserted (indeed, in a future world $\mathtt{c}$ might become insolvent). This can be formalized in $\mathcal{KALC}$ semantics as follows. Let $\mathcal{N} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ and let us consider the Kripke model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ for $\mathcal{L}_{\mathcal{N}}$ in Figure 3.1, consisting of the root ρ and two final worlds ϕ_1 and ϕ_2 such that:

– for every $\alpha \in P$, $\mathcal{D}^\alpha = \mathcal{N}$ and, for every $z \in \mathcal{N}$, $z^\alpha = z$.

– atomic concepts and roles are interpreted as follows:

World	Company	Insolvent	CW	hasCustomer
ρ	$\mathcal{N}$	$\{\mathtt{b}\}$	$\emptyset$	$\{(\mathtt{a},\mathtt{b}), (\mathtt{a},\mathtt{c}), (\mathtt{b},\mathtt{c}), (\mathtt{c},\mathtt{d})\}$
ϕ_1	$\mathcal{N}$	$\{\mathtt{b}\}$	$\{\mathtt{a}\}$	$\{(\mathtt{a},\mathtt{b}), (\mathtt{a},\mathtt{c}), (\mathtt{b},\mathtt{c}), (\mathtt{c},\mathtt{d})\}$
ϕ_2	$\mathcal{N}$	$\{\mathtt{b}, \mathtt{c}\}$	$\{\mathtt{a}\}$	$\{(\mathtt{a},\mathtt{b}), (\mathtt{a},\mathtt{c}), (\mathtt{b},\mathtt{c}), (\mathtt{c},\mathtt{d})\}$

Since $\phi_1 \Vdash \mathtt{c} : \neg\mathtt{Insolvent}$ and $\phi_2 \Vdash \mathtt{c} : \mathtt{Insolvent}$ we have that

$$\rho \nVdash \mathtt{c} : \mathtt{Insolvent} \qquad \rho \nVdash \mathtt{c} : \neg\mathtt{Insolvent}$$

hence $\rho \nVdash \mathtt{c} : \mathtt{Insolvent} \sqcup \neg\mathtt{Insolvent}$. Note that, $\mathtt{b}$ and $\mathtt{c}$ are the only individuals such that

$$(\mathtt{a},\mathtt{b}) \in \mathtt{hasCustomer}^\rho \qquad \mathtt{b} \in \mathtt{Insolvent}^\rho \qquad (\mathtt{b},\mathtt{c}) \in \mathtt{hasCustomer}^\rho$$

but $\rho \nVdash \mathtt{c} : \neg\mathtt{Insolvent}$. It follows that in ρ we can not find out for $\mathtt{a}$ the customers *ins* and *nins* required by the concept D, hence $\rho \nVdash \mathtt{a} : D$. Since

$$\phi_1 \Vdash \mathtt{a} : \mathtt{CW} \qquad \phi_2 \Vdash \mathtt{a} : \mathtt{CW}$$

we have $\rho \Vdash \mathtt{a} : D \rightarrow \mathtt{CW}$. To sum up, $\rho \Vdash \mathcal{A}$ and $\rho \nVdash \mathtt{a} : \mathtt{CW}$; we conclude that $\mathtt{a} : \mathtt{CW}$ is not a $\mathcal{KALC}$-logical consequence of $\mathcal{A}$. Observe that the final worlds correspond to the two possible way of acquiring a complete knowledge about the insolvency of $\mathtt{c}$: clearly, in the final worlds $\mathtt{a}$ must be $\mathtt{CW}$. $\diamond$

We conclude the discussion on Kripke semantics by remarking that $\mathcal{KALC}$ satisfies the *Disjunction Property (DP)*: in particular, we can prove that the property holds when assuming a set of a specific kind of formulas in the premises of $\mathcal{KALC}$-logical consequences, which can be seen as the counterpart in the description logics context of Harrop formulas [Troelstra(1973a)].

We define the *Harrop concepts* of $\mathcal{L}$ as concepts of $\mathcal{L}$ with the following form:

$$H \quad ::= \quad \bot \mid A \mid H \sqcap H \mid C \rightarrow H \mid \forall R.H$$

with $A \in \mathtt{NC}$, $R \in \mathtt{NR}$ and C any concept. The *Harrop formulas* of $\mathcal{L}$ are formulas $K \in \mathcal{L}$ defined by the following grammar:

$$K \quad ::= \quad (c,d) : R \mid c : H \mid C \sqsubseteq H$$

with $c, d \in \mathtt{NI}$, $R \in \mathtt{NR}$, H a Harrop concept and C any concept.

Proposition 3.5 (Disjunction Property)
Let $\mathcal{F}$ be a set of Harrop formulas of $\mathcal{L}$, then $\mathcal{F} \overset{\mathrm{k}}{\models} c : C_1 \sqcup C_2$ implies $\mathcal{F} \overset{\mathrm{k}}{\models} c : C_1$ or $\mathcal{F} \overset{\mathrm{k}}{\models} c : C_2$.

Proof: Let us suppose that $\mathcal{F} \overset{\mathrm{k}}{\models} c : C_1 \sqcup C_2$, but $\mathcal{F} \overset{\mathrm{k}}{\not\models} c : C_1$ and $\mathcal{F} \overset{\mathrm{k}}{\not\models} c : C_2$. Then by definition, there exist two $\mathcal{KALC}$-models:

$$\underline{K}_1 = \langle P_1, \leq_1, \rho_1, \iota_1 \rangle \qquad \underline{K}_2 = \langle P_2, \leq_2, \rho_2, \iota_2 \rangle$$

such that $\rho_1 \Vdash \mathcal{F}$ and $\rho_2 \Vdash \mathcal{F}$, but $\rho_1 \nVdash c : C_1$ and $\rho_2 \nVdash c : C_2$.

We show that we can construct a $\mathcal{KALC}$-model from $\underline{K}_1$ and $\underline{K}_2$ that contradicts the hypothesis. Let us assume, without loss of generality, that:

$$\mathtt{NI} \cap \bigcup_{\alpha \in P_1} \mathcal{D}^\alpha = \mathtt{NI} \cap \bigcup_{\alpha \in P_2} \mathcal{D}^\alpha = \emptyset$$

That is, we assume that individual names of $\mathtt{NI}$ do not appear as elements of any $\mathcal{D}^\alpha$ in $\underline{K}_1$ and $\underline{K}_2$. Let $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ be the structure defined as follows:

$$P = \{\, (1,\alpha) \mid \alpha \in P_1 \,\} \cup \{\, (2,\alpha) \mid \alpha \in P_2 \,\} \cup \{\rho\}$$

with ρ a new element and

$$\leq = \{\, (\rho,\alpha) \mid \alpha \in P \,\} \cup \{(\alpha,\beta) \in P \times P \mid \alpha = (i,\gamma), \beta = (i,\gamma') \text{ with } \gamma \leq_i \gamma'\}$$

Let $\alpha \in P$, the model $\iota(\alpha) = (\mathcal{D}^\alpha, \cdot^\alpha)$ is defined as follows:

– if $\alpha = (i, \gamma)$ for some $\gamma \in P_i$, then let $m_\alpha : \mathtt{NI} \cup \mathcal{D}^\gamma \to \mathcal{D}^\gamma$ be the function defined as follows:

$$m_\alpha(d) = \begin{cases} d & \text{if } d \in \mathcal{D}^\gamma \\ d^\gamma & \text{if } d \in \mathtt{NI} \end{cases}$$

We define:

- $\mathcal{D}^\alpha = \mathcal{D}^\gamma \cup \mathtt{NI}$;
- for every $t \in \mathtt{NI}$, $t^\alpha = t$;
- for $A \in \mathtt{NC}$ and $d \in \mathcal{D}^\alpha$, $d \in A^\alpha$ iff $m_\alpha(d) \in A^\gamma$;
- for $R \in \mathtt{NR}$ and $d, e \in \mathcal{D}^\alpha$, $(d, e) \in R^\alpha$ iff $(m_\alpha(d), m_\alpha(e)) \in R^\gamma$.

– if $\alpha = \rho$, we define:

- $\mathcal{D}^\rho = \mathtt{NI}$;
- for every $t \in \mathtt{NI}$, $t^\rho = t$
- for $A \in \mathtt{NC}$, $A^\rho = (A^{\rho_1} \cup A^{\rho_2}) \cap \mathtt{NI}$;
- for $R \in \mathtt{NR}$, $R^\rho = (R^{\rho_1} \cup R^{\rho_2}) \cap (\mathtt{NI} \times \mathtt{NI})$.

Let M be the family of all the functions in $\{m_\alpha \mid \alpha \in P \setminus \{\rho\}\}$. We note that M has the following property:

(P1) if $\alpha = (i, \alpha')$ and $\beta = (i, \beta')$ in $P \setminus \{\rho\}$ with $(i, \alpha') \leq (i, \beta')$, then for every $d \in \mathcal{D}^{\alpha'} \cup \mathtt{NI}$, $m_\alpha(d) = m_\beta(d)$.

This holds since, if $d \in \mathcal{D}^{\alpha'}$ then $d \in \mathcal{D}^{\beta'}$ and we have that $m_\alpha(d) = m_\beta(d) = d$. On the other hand, if $d \in \mathtt{NI}$ then, by definition of each model $\underline{K}_i$, it holds $d^{\alpha'} = d^{\beta'}$: this implies that $m_\alpha(d) = (d)^{\alpha'} = (d)^{\beta'} = m_\beta(d)$.

We can now verify that $\underline{K}$ is indeed a $\mathcal{KALC}$-model:

– the set P is finite by definition;

– if $\alpha, \beta \in P$ and $\alpha \leq \beta$ then, by definition, $\mathcal{D}^\alpha \subseteq \mathcal{D}^\beta$;

– for every $\alpha, \beta \in P$, by definition, it holds that $t^\alpha = t^\beta$, for $t \in \mathtt{NI}$;

– let $\alpha \leq \beta$ with $\alpha \neq \beta$. If $\alpha = \rho$ and $\beta = (i, \gamma)$, then by definition we have that, for every $A \in \mathtt{NC}$, $A^\rho = A^\beta$ and, for every $R \in \mathtt{NR}$, $R^\rho = R^\beta$;

Now let us suppose that $\alpha = (i, \gamma)$ and $\beta = (i, \delta)$. Let $A \in \mathtt{NC}$ and $d \in \mathcal{D}^\alpha$: then $d \in A^\alpha$ iff $m_\alpha(d) \in A^\gamma$. Given that $A^\gamma \subseteq A^\delta$, this holds if $m_\alpha(d) \in A^\delta$ and, by (P1), iff $m_\beta(d) \in A^\delta$. Thus this holds iff $d \in A^\beta$: this implies that $A^\alpha \subseteq A^\beta$. Let $R \in \mathtt{NR}$ and $d, e \in \mathcal{D}^\alpha$: then $(d, e) \in R^\alpha$ iff $(m_\alpha(d), m_\alpha(e)) \in R^\gamma$. Given that $R^\gamma \subseteq R^\delta$, this holds if $(m_\alpha(d), m_\alpha(e)) \in R^\delta$ and, by (P1), iff $(m_\beta(d), m_\beta(e)) \in R^\delta$. Thus the fact holds iff $(d, e) \in R^\beta$: this implies that $R^\alpha \subseteq R^\beta$.

We can prove the following property:

(P2) For every $\alpha = (i, \gamma) \in P \setminus \{\rho\}$ and every $H \in \mathcal{L}_\alpha$, $\alpha \Vdash H$ iff $\gamma \Vdash H$.

We first note that, by definition, $\mathcal{L}_\alpha = \mathcal{L}_\gamma$. The assertion can be proved by cases on the structure of the formula H: we only give the proof for some of the relevant cases.

Let $H = (d, e) : R$ with $R \in \mathtt{NR}$. Then, $\alpha \Vdash (d, e) : R$ iff $(d^\alpha, e^\alpha) \in R^\alpha$. By definition, this holds iff $(m_\alpha(d^\alpha), m_\alpha(e^\alpha)) \in R^\gamma$. The fact holds iff $(d^\gamma, e^\gamma) \in R^\gamma$ and thus iff $\gamma \Vdash (d, e) : R$.

Let $H = d : C$: we can show the assertion by induction on the definition of the concept C. If $H = t : A$ with $A \in \mathtt{NC}$ or $A = \bot$, then $\alpha \Vdash d : A$ iff $d^\alpha \in A^\alpha$. By definition, this holds iff $m_\alpha(d^\alpha) \in A^\gamma$. We have that $m_\alpha(d^\alpha) = d^\gamma$: thus this holds iff $d^\gamma \in A^\gamma$ and finally iff $\gamma \Vdash d : A$.

Let $H = d : C \to D$. Let us suppose that $\alpha \nVdash d : C \to D$: then there exists a $\beta \in P$ such that $\alpha \leq \beta$ and $\beta \Vdash d : C$ but $\beta \nVdash d : D$. If $\beta = (i, \gamma')$, by induction hypothesis we have that $\gamma' \Vdash d : C$ but $\gamma' \nVdash d : D$: this implies that $\gamma \nVdash d : C \to D$. On the other hand, suppose that $\gamma \nVdash d : C \to D$. Then there exists $\gamma' \in P_i$ such that $\gamma \leq \gamma'$ and $\gamma' \Vdash d : C$ but $\gamma' \nVdash d : D$. By induction hypothesis, $(i, \gamma') \Vdash d : C$ and $(i, \gamma') \nVdash d : D$, which implies $\alpha \nVdash d : C \to D$.

Let $H = d : \forall R.C$. If $\alpha \nVdash d : \forall R.C$, then there exists $\beta \in P$ and $e \in \mathcal{D}^\beta$ such that $\alpha \leq \beta$, $\beta \Vdash (d, e) : R$ and $\beta \nVdash e : C$. If $\beta = (i, \gamma')$, by induction hypothesis we have that $\gamma' \Vdash (d, e) : R$ but $\gamma' \nVdash e : C$ which implies $\gamma \nVdash d : \forall R.C$. On the other hand, suppose that $\gamma \nVdash d : \forall R.C$. Then there exists a $\gamma' \in P_i$ and $e \in \mathcal{D}^{\gamma'}$ such that $\gamma \leq \gamma'$, $\gamma' \Vdash (d, e) : R$ and $\gamma' \nVdash e : C$. By induction hypothesis, we have that $(i, \gamma') \Vdash (d, e) : R$ but $(i, \gamma') \nVdash e : C$ which implies $\alpha \nVdash d : \forall R.C$.

Let $H = C \sqsubseteq D$. Suppose that $\alpha \nVdash C \sqsubseteq D$. Then there exists $\beta \in P$ and $d \in \mathcal{D}^\beta$ with $\alpha \leq \beta$ such that $\beta \Vdash d : C$ but $\beta \nVdash d : D$. Let $\beta = (i, \gamma')$. By the induction hypothesis $\gamma' \Vdash d : C$ but $\gamma' \nVdash d : D$. This implies $\gamma \nVdash C \sqsubseteq D$. On the other hand, suppose that $\gamma \nVdash C \sqsubseteq D$. Then there exists a $\gamma' \in P_i$ and $d \in \mathcal{D}^{\gamma'}$ with $\gamma \leq \gamma'$ such that $\gamma' \Vdash d : C$ but $\gamma' \nVdash d : D$. By induction hypothesis over concept formulas, we have that $(i, \gamma') \Vdash d : C$ and $(i, \gamma') \nVdash d : D$. This implies $\alpha \nVdash C \sqsubseteq D$.

Now, we can show the following property:

(P3) For every Harrop formula $K \in \mathcal{L}$, $\rho \Vdash K$ iff $\rho_1 \Vdash K$ and $\rho_2 \Vdash K$.

This can be shown by cases on the structure of the Harrop formula K.

Let $K = (c, d) : R$. Then by construction we have that $R^\rho = (R^{\rho_1} \cap R^{\rho_2}) \cap (\mathtt{NI} \times \mathtt{NI})$ and the assertion immediately follows.

Let $K = c : H$ with H an Harrop concept: we show the assertion by induction on the structure of H. For $H = \bot$ the assertion holds by definition.

For $H = A$ with $A \in \mathtt{NC}$, by construction we have that $A^\rho = (A^{\rho_1} \cap A^{\rho_2}) \cap \mathtt{NI}$ thus the assertion immediately follows.

For $H = H_1 \sqcap H_2$, we have that $\rho \Vdash c : H_1 \sqcap H_2$ iff $\rho \Vdash c : H_1$ and $\rho \Vdash c : H_2$. By induction hypothesis, this holds iff $\rho_i \Vdash c : H_1$ and $\rho_i \Vdash c : H_2$ for $i \in \{1, 2\}$ and this holds iff $\rho_i \Vdash c : H_1 \sqcap H_2$ for $i \in \{1, 2\}$.

For $H = C \to H'$, let us suppose that $\rho_i \Vdash c : C \to H'$ for each $i \in \{1,2\}$, but $\rho \nVdash c : C \to H'$. Then there exists $\alpha \in P$ with $\alpha \geq \rho$ such that $\alpha \Vdash c : C$ but $\alpha \nVdash c : H'$. If $\alpha = \rho$ then, by induction hypothesis, for an $i \in \{1,2\}$ it holds $\rho_i \nVdash c : H'$. Moreover, since $\alpha \leq (i, \rho_i)$, we have that $(i, \rho_i) \Vdash c : C$ which implies, by (P1), that $\rho_i \Vdash c : C$. This implies $\rho_i \nVdash c : C \to H'$, which contradicts our hypothesis. If $\alpha \neq \rho$ then there exists $i \in \{1,2\}$ and $\gamma \in P_i$ such that $(i, \gamma) \Vdash c : C$ and $(i, \gamma) \nVdash c : H'$. By (P1) this implies that $\rho_i \nVdash c : C \to H'$, which contradicts our hypothesis. Now let us suppose that $\rho \Vdash c : C \to H'$ but $\rho_i \nVdash c : C \to H'$ for at least an $i \in \{1,2\}$. Then there exists $\gamma \in P$ with $\gamma \geq \rho_i$ such that $\gamma \Vdash c : C$ but $\gamma \nVdash c : H'$. By (P1) we have that $(i, \gamma) \Vdash c : C$ and $(i, \gamma) \nVdash c : H'$. This implies that $\rho \nVdash c : C \to H'$, against our assumptions.

For $H = \forall R.H'$, let us suppose that $\rho \nVdash c : \forall R.H'$. Then there exist $\alpha \geq \rho$ and $\delta \in \mathcal{D}^\alpha$ such that $\alpha \Vdash (c, d) : R$ but $\alpha \nVdash d : H'$. If $\alpha = \rho$ then, by induction hypothesis, we have that there exists $i \in \{1,2\}$ such that $\rho_i \Vdash (c, d) : R$ and $\rho_i \nVdash d : H'$. Thus, $\rho_i \nVdash c : \forall R.H'$. If $\alpha \neq \rho$ then there exist $i \in \{1,2\}$ and $\gamma \in P_i$ such that $\alpha = (i, \gamma)$: by (P1) we have that $\gamma \Vdash (c, d) : R$ and $\gamma \nVdash d : H'$, thus $\rho_i \nVdash c : \forall R.H'$. On the other hand, let $\rho_i \nVdash c : \forall R.H'$ for at least an $i \in \{1,2\}$. Then there exist $\gamma \in P_i$ and $d \in \mathcal{D}^\gamma$ such that $\gamma \Vdash (c, d) : R$ and $\gamma \nVdash d : H'$. By (P1) we have that $(i, \gamma) \Vdash (c, d) : R$ and $(i, \gamma) \nVdash d : H'$ and this implies $\rho \nVdash c : \forall R.H'$.

Let $K = C \sqsubseteq H'$. Let us suppose that $\rho \nVdash C \sqsubseteq H'$. Then there exist $\alpha \in P$ with $\alpha \geq \rho$ and $d \in \mathcal{D}^\alpha$ such that $\alpha \Vdash d : C$ but $\alpha \nVdash d : H'$. If $\alpha = \rho$, then by induction hypothesis over Harrop concepts we have that there exists $i \in \{1,2\}$ such that $\rho_i \nVdash d : H'$. Moreover, as $\rho \leq \rho_i$ implies $(i, \rho_i) \Vdash d : C$, by (P1) we have that $\rho_i \Vdash d : C$. Thus it follows that $\rho_i \nVdash C \sqsubseteq H'$. If $\alpha \neq \rho$, then there exist $i \in \{1,2\}$ and $\gamma \in P_i$ such that $(i, \gamma) \Vdash d : C$ and $(i, \gamma) \nVdash d : H'$. By (P1) we deduce that $\gamma \Vdash d : C$ but $\gamma \nVdash d : H'$: thus, it follows that $\rho_i \nVdash C \sqsubseteq H'$. On the other hand, if there is a $i \in \{1,2\}$ such that $\rho_i \nVdash C \sqsubseteq H'$, then there exists a $\gamma \in P_i$ with $\rho_i \leq \gamma$ and $d \in \mathcal{D}^\gamma$ such that $\gamma \Vdash d : C$ but $\gamma \nVdash d : H'$. By (P1), it follows that $(i, \gamma) \Vdash d : C$ but $(i, \gamma) \nVdash d : H'$: thus, we have that $\rho \nVdash C \sqsubseteq H'$.

Finally, we can prove our main assertion as follows: since $\rho_1 \Vdash \mathcal{F}$ and $\rho_2 \Vdash \mathcal{F}$ with $\mathcal{F}$ a set of Harrop formulas, by (P3) we have that $\rho \Vdash \mathcal{F}$. Moreover, given that $\rho_1 \nVdash c : C_1$ and $\rho_2 \nVdash c : C_2$, by (P2) it holds that $(1, \rho_1) \nVdash c : C_1$ and $(2, \rho_2) \nVdash c : C_2$, thus $\rho \nVdash c : C_1$ and $\rho \nVdash c : C_2$. Then $\rho \nVdash c : C_1 \sqcup C_2$. This is an absurd, since we assumed that $\mathcal{F} \overset{\mathrm{k}}{\models} c : C_1 \sqcup C_2$. □

The usual formulation of the Disjunction Property holds in $\mathcal{KALC}$ as a corollary of the previous proposition, by assuming an empty set of formulas in the premises of the logical consequence.

Corollary 3.6 (Disjunction Property)
$\overset{\mathrm{k}}{\models} c : C_1 \sqcup C_2$ *implies* $\overset{\mathrm{k}}{\models} c : C_1$ *or* $\overset{\mathrm{k}}{\models} c : C_2$. □

We also note that, as an immediate consequence of the Disjunction Property, *tertium non datur* does not hold in $\mathcal{KALC}$, albeit it is a valid principle in $\mathcal{ALC}$.

3.2 The tableau calculus $\mathcal{T}_{\mathcal{K}}$

In this section we introduce the *tableau calculus* $\mathcal{T}_{\mathcal{K}}$ for $\mathcal{KALC}$ and we prove its properties with respect to $\mathcal{KALC}$ semantics. This calculus is defined inspiring to tableau calculi for classical and intuitionistic logics: as in the case for these calculi, $\mathcal{T}_{\mathcal{K}}$ is a goal oriented refutation calculus which proofs are built by decomposing sets of signed formulas. For a detailed presentation of tableaux calculi and their customary notation, we refer the reader to [Smullyan(1968), Fitting(1983)].

The tableau calculus $\mathcal{T}_{\mathcal{K}}$ works on *signed formulas* $W = \mathcal{S}(H)$, with H a formula of $\mathcal{L}$ and $\mathcal{S}$ a sign in $\{\mathbf{T}, \mathbf{F}, \mathbf{T_s}\}$. Formally:

$$W \ ::= \ \mathbf{T}((c,d) : R) \mid \mathbf{F}(c : C) \mid \mathbf{T}(c : C) \mid \mathbf{T_s}(c : C) \mid \mathbf{T}(C \sqsubseteq D)$$

Given a set of formulas $\mathcal{F}$ and a sign $\mathcal{S}$, $\mathcal{S}(\mathcal{F})$ denotes the set of signed formulas $\mathcal{S}(H)$ such that $H \in \mathcal{F}$.

Semantics of signed formulas is obtained by extending the interpretation of formulas with the meaning of each sign. Given a $\mathcal{KALC}$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$, a world $\alpha \in P$ and a signed formula W, α *realizes* W *in* $\underline{K}$, and we write $\underline{K}, \alpha \rhd W$, iff:

– $W = \mathbf{T}(H)$ and $\alpha \Vdash H$.

– $W = \mathbf{F}(H)$ and $\alpha \nVdash H$.

– $W = \mathbf{T_s}(H)$ and, for every $\beta \in P$ such that $\alpha < \beta$, $\underline{K}, \beta \rhd \mathbf{T}(H)$.

The signs **T** and **F** represent the notion of known and unknown formula and are customary in the presentation of tableau calculi [Fitting(1983)]. On the other hand, the sign $\mathbf{T}_s$ is new to our calculus and refers to the successors of a world. In particular, $\mathbf{T}_s(H)$ is realized in a world α if $\mathbf{T}(H)$ holds in every successor of α. We remind that it is not required $\mathbf{T}(H)$ to hold in α. We also remark that $\underline{K}, \alpha \rhd \mathbf{T_s}(c : \bot)$ iff α is final.

By the monotonicity property (Proposition 3.1), we get that **T**-signed formulas are *persistent*: formally, if $\alpha, \beta \in P$ and $\alpha \leq \beta$, then $\underline{K}, \alpha \rhd \mathbf{T}(H)$ implies $\underline{K}, \beta \rhd \mathbf{T}(H)$. On the other hand, **F**-signed formulas are not persistent: however, we can say that they are "downward" persistent, in the sense that if $\alpha, \beta \in P$ and $\alpha \leq \beta$, then $\underline{K}, \beta \rhd \mathbf{F}(H)$ implies $\underline{K}, \alpha \rhd \mathbf{F}(H)$.

Given a set of signed formulas Δ, we write $\underline{K}, \alpha \rhd \Delta$ if $\underline{K}, \alpha \rhd W$ for every $W \in \Delta$. We say that Δ *is realizable* if $\underline{K}, \alpha \rhd \Delta$ for some $\underline{K}$ and α.

The relations among realizability, $\mathcal{KALC}$-logical consequence and $\mathcal{ALC}$-logical consequence are stated by the following theorem:

Theorem 3.7
Let $\mathcal{F}$ be a set of formulas and q an individual name not in $\mathcal{F}$.

(i) $\mathcal{F} \models^{\mathrm{k}} c : C$ *iff the set* $\mathbf{T}(\mathcal{F}) \cup \{\mathbf{F}(c : C)\}$ *is not realizable.*

(ii) $\mathcal{F} \models^{\mathrm{k}} C \sqsubseteq D$ *iff the set* $\mathbf{T}(\mathcal{F}) \cup \{\mathbf{F}(q : C \to D)\}$ *is not realizable.*

(iii) H *is an* $\mathcal{ALC}$*-logical consequence of* $\mathcal{F}$ *iff* $\mathbf{T}(\mathcal{F}) \cup \{\mathbf{T_s}(c : \bot), \mathbf{F}(H)\}$ *is not realizable.*

(iv) $c : C$ *is an* $\mathcal{ALC}$*-logical consequence of* $\mathcal{F}$ *iff* $\mathcal{F} \mid\!\!\overset{\mathrm{k}}{=} c : \neg\neg C$.

Proof:

(i) Suppose that $S = \mathbf{T}(\mathcal{F}) \cup \{\mathbf{F}(c : C)\}$ is realizable. Then there exist a model $\underline{K}'$ and a world γ of $\underline{K}'$ such that $\underline{K}', \gamma \rhd S$. This implies that $\gamma \Vdash \mathcal{F}$ and $\gamma \nVdash c : C$. This is an absurd, since assuming $\mathcal{F} \mid\!\!\overset{\mathrm{k}}{=} c : C$ implies that for every model $\underline{K}$ and world α of $\underline{K}$, if $\alpha \Vdash \mathcal{F}$ then $\alpha \Vdash c : C$.

On the other hand, let us suppose that $\mathcal{F} \not\mid\!\!\overset{\mathrm{k}}{=} c : C$. This implies that there exists a model $\underline{K}'$ and a world γ of $\underline{K}'$ such that $\gamma \Vdash \mathcal{F}$ and $\gamma \nVdash c : C$. However, since S is not realized, for every model $\underline{K}$ and world α of $\underline{K}$, $\alpha \nVdash \mathcal{F}$ or $\alpha \Vdash c : C$: that is, if $\alpha \Vdash \mathcal{F}$ then $\alpha \Vdash c : C$. This contradicts the previous hypothesis, thus the assertion is proved.

(ii) Suppose that $S = \mathbf{T}(\mathcal{F}) \cup \{\mathbf{F}(q : C \to D)\}$ is realizable. Then there exists a model $\underline{K}'$ and a world γ of $\underline{K}'$ such that $\underline{K}', \gamma \rhd S$, that is $\gamma \Vdash \mathcal{F}$ and $\gamma \nVdash q : C \to D$. However, by assuming $\mathcal{F} \mid\!\!\overset{\mathrm{k}}{=} C \sqsubseteq D$ we have that for every model $\underline{K}$ and world α of $\underline{K}$, if $\alpha \Vdash \mathcal{F}$ then $\alpha \Vdash C \sqsubseteq D$. This implies that, for every $c \in \mathcal{D}^\alpha$ and $\beta \geq \alpha$, $\beta \Vdash c : C$ implies $\beta \Vdash c : D$. This is an absurd, since this implies that, for every $c \in \mathcal{D}^\alpha$, $\alpha \Vdash c : C \to D$ against the previous assertions.

On the other hand, let us suppose that $\mathcal{F} \not\mid\!\!\overset{\mathrm{k}}{=} C \sqsubseteq D$. Then there exists a model $\underline{K}'$ and a world γ of $\underline{K}'$ such that $\gamma \Vdash \mathcal{F}$ and $\gamma \nVdash C \sqsubseteq D$: thus there exist a world β of $\underline{K}'$ and a $d \in \mathcal{D}^\beta$ such that $\beta \geq \gamma$ and $\beta \Vdash d : C$ but $\beta \nVdash d : D$. This implies that $\beta \nVdash d : C \to D$: however, since S is not realizable, for every model $\underline{K}$ and world α of $\underline{K}$, $\alpha \nVdash \mathcal{F}$ or $\alpha \Vdash q : C \to D$. This is an absurd since this would imply that, for every individual name q not occurring in $\mathcal{F}$, $\alpha \Vdash \mathcal{F}$ implies $\alpha \Vdash q : C \to D$.

(iii) Suppose that $S = \mathbf{T}(\mathcal{F}) \cup \{\mathbf{T_s}(c : \bot), \mathbf{F}(H)\}$ is realizable. Then there exist a $\mathcal{KALC}$-model $\underline{K}$ and a world ϕ of $\underline{K}$ such that $\phi \Vdash \mathcal{F}$ and $\phi \nVdash H$ with ϕ a final state of $\underline{K}$. Consider the model:

$$\mathcal{M}_\phi = (\mathcal{D}^\phi, \cdot^\phi)$$

As we previously discussed, $\mathcal{M}_\phi$ represents a classical $\mathcal{ALC}$-model. Moreover, for every formula $K \in \mathcal{L}_\phi$, $\mathcal{M}_\phi \models K$ iff $\phi \Vdash K$. Thus we have $\mathcal{M}_\phi \models \mathcal{F}$ but $\mathcal{M}_\phi \not\models H$: this contradicts our assumption that $\mathcal{F} \models H$, thus the assertion is proved.

On the other hand, suppose that $\mathcal{F} \not\models H$. This implies that there exists an $\mathcal{ALC}$-model $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{F}$ but $\mathcal{M} \not\models H$. Let us define the structure

$$\underline{K}_\mathcal{M} = \langle \{\rho\}, \{(\rho, \rho)\}, \rho, \iota \rangle$$

with $\iota(\rho) = \mathcal{M}$. As we previously discussed $\underline{K}_{\mathcal{M}}$ is a $\mathcal{KALC}$-model and, for every formula $K \in \mathcal{L}_{\rho}$, $\mathcal{M} \models K$ iff $\rho \Vdash K$ in $\underline{K}_{\mathcal{M}}$. Thus, ρ is final in $\underline{K}_{\mathcal{M}}$ and $\rho \Vdash \mathcal{F}$ but $\rho \nVdash H$. However, given that S is not realizable, we have that for every $\mathcal{KALC}$-model $\underline{K}$ and world α of $\underline{K}$, if α is final, then $\alpha \Vdash \mathcal{F}$ implies $\alpha \Vdash H$. This is an absurd, thus the assertion is proved.

(iv) Suppose that $\mathcal{F} \not\models^{\underline{k}} c : \neg\neg C$. Then there exists a $\mathcal{KALC}$-model $\underline{K}$ and a world α of $\underline{K}$ such that $\alpha \Vdash \mathcal{F}$ and $\alpha \nVdash c : \neg\neg C$. This means that there exists a world β of $\underline{K}$ with $\beta \geq \alpha$ such that $\beta \Vdash c : \neg C$: that is, for every γ of $\underline{K}$ with $\gamma \geq \beta$, it holds that $\gamma \Vdash \mathcal{F}$ but $\gamma \nVdash c : C$. Given that every $\mathcal{KALC}$-model is finite, there always exists a $\phi \geq \beta$ that is final and $\phi \nVdash c : C$. As in the previous case, we can consider the $\mathcal{ALC}$-model:

$$\mathcal{M}_{\phi} = (\mathcal{D}^{\phi}, \cdot^{\phi})$$

Thus we have $\mathcal{M}_{\phi} \models \mathcal{F}$ but $\mathcal{M}_{\phi} \not\models c : C$: since this contradicts our assumption that $\mathcal{F} \models c : C$, the assertion is proved.

On the other hand, suppose that $\mathcal{F} \not\models c : C$. Then there exists an $\mathcal{ALC}$-model $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{F}$ but $\mathcal{M} \not\models c : C$. As above, we can define the $\mathcal{KALC}$-model

$$\underline{K}_{\mathcal{M}} = \langle \{\rho\}, \{(\rho, \rho)\}, \rho, \iota \rangle$$

with $\iota(\rho) = \mathcal{M}$. We have that $\rho \Vdash \mathcal{F}$ but $\rho \nVdash c : C$ and, since ρ is final, $\rho \nVdash c : \neg\neg C$. This is an absurd, since by the hypothesis that $\mathcal{F} \models^{\underline{k}} c : \neg\neg C$ we have that, for every $\mathcal{KALC}$-model $\underline{K}$ and α of $\underline{K}$, $\alpha \Vdash \mathcal{F}$ implies $\alpha \Vdash c : \neg\neg C$.

□

The rules of the tableau calculus $\mathcal{T}_{\mathcal{K}}$ are shown in Figure 3.2. The rules of $\mathcal{T}_{\mathcal{K}}$ have the form:

$$\frac{\Delta}{\Delta_1 \mid \ldots \mid \Delta_n}\, r$$

where we call Δ the *premise* of the rule r, and $\Delta_1, \ldots, \Delta_n$ are the *consequences* of r. Every rule applies to a set of signed formulas, but only acts on the signed formula W explicitly indicated in the premise. In the rules $\mathbf{F}\rightarrow$, $\mathbf{T}\rightarrow$, $\mathbf{F}\forall$ we define the set Δ_s as follows:

$$\Delta_s \;=\; \{\, \mathbf{T}(H) \mid \mathbf{T}(H) \in \Delta \,\} \;\cup\; \{\, \mathbf{T}(H) \mid \mathbf{T_s}(H) \in \Delta \,\}$$

We remark that $\underline{K}, \alpha \rhd \Delta$ implies $\underline{K}, \beta \rhd \Delta_s$ for every $\beta > \alpha$. The application of the rules $\mathbf{T}\sqsubseteq$, $\mathbf{F}\exists$ and $\mathbf{T}\forall$ is constrained by the presence of the additional formula $\mathbf{T}(c : A)$ or $\mathbf{T}((c, d) : R)$ in the premises. In the rules we write Δ, W as a shorthand for $\Delta \cup \{W\}$. If Δ, W is the premise of a rule, we assume $W \notin \Delta$.

In the rules $\mathbf{T}\exists$ and $\mathbf{F}\forall$, q is a fresh individual name. Formulas of the kind

$$\mathbf{F}(c : \exists R.C) \qquad \mathbf{T}(c : \forall R.C) \qquad \mathbf{T}(A \sqsubseteq C)$$

$$\frac{\Delta, \mathbf{T}(c : C \sqcap D)}{\Delta, \mathbf{T}(c : C), \mathbf{T}(c : D)}\,\mathbf{T}\sqcap \qquad \frac{\Delta, \mathbf{F}(c : C \sqcap D)}{\Delta, \mathbf{F}(c : C) \mid \Delta, \mathbf{F}(c : D)}\,\mathbf{F}\sqcap$$

$$\frac{\Delta, \mathbf{T}(c : C \sqcup D)}{\Delta, \mathbf{T}(c : C) \mid \Delta, \mathbf{T}(c : D)}\,\mathbf{T}\sqcup \qquad \frac{\Delta, \mathbf{F}(c : C \sqcup D)}{\Delta, \mathbf{F}(c : C), \mathbf{F}(c : D)}\,\mathbf{F}\sqcup$$

$$\frac{\Delta, \mathbf{F}(c : C \to D)}{\Delta, \mathbf{T}(c : C), \mathbf{F}(c : D) \mid \Delta_s, \mathbf{T}(c : C), \mathbf{F}(c : D)}\,\mathbf{F}\to$$

$$\frac{\Delta, \mathbf{T}(c : C \to D)}{\Delta, \mathbf{T}(c : D) \mid \Delta, \mathbf{F}(c : C), \mathbf{T_s}(c : D) \mid \Delta_s, \mathbf{F}(c : C), \mathbf{T_s}(c : D)}\,\mathbf{T}\to$$

$$\frac{\Delta, \mathbf{T}(c : A), \mathbf{T}(A \sqsubseteq C)}{\Delta, \mathbf{T}(c : A), \mathbf{T}(A \sqsubseteq C), \mathbf{T}(c : C)}\,\mathbf{T}\sqsubseteq$$

$$\frac{\Delta, \mathbf{T}(c : \exists R.C)}{\Delta, \mathbf{T}((c, q) : R), \mathbf{T}(q : C)}\,\mathbf{T}\exists^* \qquad \frac{\Delta, \mathbf{T}((c, d) : R), \mathbf{F}(c : \exists R.C)}{\Delta, \mathbf{T}((c, d) : R), \mathbf{F}(c : \exists R.C), \mathbf{F}(d : C)}\,\mathbf{F}\exists$$

$$\frac{\Delta, \mathbf{T}((c, d) : R), \mathbf{T}(c : \forall R.C)}{\Delta, \mathbf{T}((c, d) : R), \mathbf{T}(c : \forall R.C), \mathbf{T}(d : C)}\,\mathbf{T}\forall$$

$$\frac{\Delta, \mathbf{F}(c : \forall R.C)}{\Delta, \mathbf{T}((c, q) : R), \mathbf{F}(q : C) \mid \Delta_s, \mathbf{T}((c, q) : R), \mathbf{F}(q : C)}\,\mathbf{F}\forall^*$$

$^*q$ does not occur in the premise

Figure 3.2: Rules of $\mathcal{T}_\mathcal{K}$

must be *duplicated* in rule application to guarantee the completeness: in other words, these signed formulas are repeated in the consequences of their respective rules, thus the rules can be applied indefinitely many times. We call these formulas *dup-formulas*. Note that in the intuitionistic case the treatment of $\mathbf{T}\ \rightarrow$-rule is problematic and requires duplications [Avellone *et al.*(1999)] while in $\mathcal{T}_{\mathcal{K}}$ duplications are avoided by the introduction of the sign $\mathbf{T_s}$. The same problem, in general, also appears on the treatment of subsumption: however, in our case this problem is prevented by the limitation on the form of TBox formulas.

A set Δ *clashes* iff:

$$\{\, \mathbf{F}(c : C), \mathbf{T}(c : C) \,\} \subseteq \Delta \quad \text{or} \quad \mathbf{T}(c : \bot) \in \Delta$$

Clearly, a clashing set is not realizable.

A *proof table* for Δ is a finite tree τ such that:

- the root of τ is Δ;
- given a node Δ' of τ, the successors of Δ' in τ are the sets in the consequences of an instance of a rule having Δ' as premise.

If all the leaves of τ clash, τ is a *closed proof table* for Δ. If there exists a closed proof table for Δ, then we say that Δ is *provable* in $\mathcal{T}_{\mathcal{K}}$. We say that Δ is *consistent* iff Δ is not provable.

Before proving soundness and completeness of the calculus, we give some examples of proofs.

Example 2
Let $H = C \sqcup \neg C$. Since $c : H$ is valid in $\mathcal{ALC}$, by Theorem 3.7 the formula $c : \neg\neg H$ is valid in $\mathcal{KALC}$. We show a proof of $c : \neg\neg H$, recalling that $\neg D = D \rightarrow \bot$ in $\mathcal{KALC}$. The proof-tree is displayed according to standard notation used in tableau systems [Fitting(1983)]. In the proof we underline the clashing formulas, we abbreviate with X a clashing set and we label with an integer the formulas treated by the rules when needed.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{F}(c : \neg\neg H)}{\mathbf{T}(c : \neg H), \mathbf{F}(c : \bot)}\,\mathbf{F}\rightarrow}{\underline{\mathbf{T}(c : \bot)}, \mathbf{F}(c : \bot) \mid \mathbf{F}(c : H)^1, \mathbf{T_s}(c : \bot), \mathbf{F}(c : \bot) \mid \mathbf{F}(c : H)^2, \mathbf{T_s}(c : \bot)}\,\mathbf{T}\rightarrow}{\mathsf{X} \mid \mathbf{F}(c : C), \mathbf{F}(c : \neg C)^3, \mathbf{T_s}(c : \bot), \mathbf{F}(c : \bot) \mid \Delta = \mathbf{F}(c : C), \mathbf{F}(c : \neg C)^4, \mathbf{T_s}(c : \bot)}\,\mathbf{F}\sqcup^1,\mathbf{F}\sqcup^2}{\mathsf{X} \mid \underline{\mathbf{F}(c : C)}, \underline{\mathbf{T}(c : C)}, \mathbf{T_s}(c : \bot), \mathbf{F}(c : \bot) \mid \mathbf{T}(c : C), \mathbf{F}(c : \bot), \underline{\mathbf{T}(c : \bot)} \mid \Delta}\,\mathbf{F}\rightarrow^3}{\mathsf{X} \mid \mathsf{X} \mid \mathsf{X} \mid \underline{\mathbf{F}(c : C)}, \underline{\mathbf{T}(c : C)}, \mathbf{F}(c : \bot), \mathbf{T_s}(c : \bot) \mid \mathbf{T}(c : C), \mathbf{F}(c : \bot), \underline{\mathbf{T}(c : \bot)}}\,\mathbf{F}\rightarrow^4$$

Note that, if $\mathbf{T_s}(c : \bot) \in \Delta$, then Δ_s clashes. In this case, in applying one of the rules $\mathbf{F} \rightarrow$, $\mathbf{T} \rightarrow$ and $\mathbf{F}\forall$ to Δ, we can drop out the rightmost set in the conclusion: thus, a proof table for $\Delta, \mathbf{T_s}(c : \bot)$ resembles an $\mathcal{ALC}$ proof table. $\diamond$

An interesting axiom schema that can be proved to be valid in $\mathcal{KALC}$ is a reformulation of the *Kuroda principle* for first order logic

$$\forall x.\neg\neg H(x) \rightarrow \neg\neg\forall x.H(x)$$

which is a well-known principle in the constructive logic literature [Gabbay(1981), Troelstra(1973b)]. In the description logics setting, the principle can be restated as

$$(\text{Kur}) \quad \forall R.\neg\neg A \rightarrow \neg\neg\forall R.A$$

We will discuss in more detail the role of this axiom schema and its reformulation in the context of modal logics in Chapter 4.

Example 3
The following proof table exhibits a proof of a general instance of Kur.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{F}(c : \forall R.\neg\neg A \rightarrow \neg\neg\forall R.A)}{\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{F}(c : \neg\neg\forall R.A)}\,\mathbf{F}\rightarrow}{\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}(c : \neg\forall R.A)}\,\mathbf{F}\rightarrow}{\mathbf{T}(c : \forall R.\neg\neg A), \underline{\mathbf{T}(c : \bot)} \mid \mathbf{T}(c : \forall R.\neg\neg A), \mathbf{F}(c : \forall R.A), \mathbf{T_s}(c : \bot)}\,\mathbf{T}\rightarrow}{\times \;\Big|\; \begin{array}{l}\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}((c,q) : R),\\ \mathbf{F}(q : A), \mathbf{T_s}(c : \bot)\end{array} \;\Big|\; \begin{array}{l}\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}((c,q) : R),\\ \mathbf{F}(q : A), \underline{\mathbf{T}(c : \bot)}\end{array}}\,\mathbf{F}\forall}{\times \;\Big|\; \begin{array}{l}\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}((c,q) : R),\\ \mathbf{F}(q : A), \mathbf{T_s}(c : \bot), \mathbf{T}(q : \neg\neg A)\end{array} \;\Big|\; \times}\,\mathbf{T}\forall}{\times \;\Big|\; \begin{array}{l}\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}((c,q) : R),\\ \mathbf{F}(q : A), \mathbf{T_s}(c : \bot), \underline{\mathbf{T}(q : \bot)}\end{array} \;\Big|\; \begin{array}{l}\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}((c,q) : R),\\ \mathbf{F}(q : A), \mathbf{T_s}(c : \bot), \mathbf{F}(q : \neg A)\end{array} \;\Big|\; \times}\,\mathbf{T}\rightarrow}{\times \mid \times \;\Big|\; \begin{array}{l}\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}((c,q) : R),\\ \mathbf{F}(q : A), \underline{\mathbf{T}(c : \bot)}, \mathbf{T}(q : A)\end{array} \;\Big|\; \begin{array}{l}\mathbf{T}(c : \forall R.\neg\neg A), \mathbf{T}((c,q) : R),\\ \underline{\mathbf{F}(q : A)}, \mathbf{T_s}(c : \bot), \underline{\mathbf{T}(q : A)}\end{array} \;\Big|\; \times}\,\mathbf{F}\rightarrow$$

$\diamond$

3.3 Soundness of $\mathcal{T}_{\mathcal{K}}$

By the definition of $\mathcal{T}_{\mathcal{K}}$, we can prove its soundness with respect to $\mathcal{KALC}$: we prove the result by showing that given a realizable set of signed formulas Δ, then Δ is consistent w.r.t. $\mathcal{T}_{\mathcal{K}}$.

Lemma 3.8
Let Δ be a set of signed formulas, $\underline{K} = \langle P, \leq, \rho, \iota\rangle$ a $\mathcal{KALC}$-model such that $\underline{K}, \alpha \rhd \Delta$, with $\alpha \in P$, and r a rule of $\mathcal{T}_{\mathcal{K}}$ applicable to Δ. Then, there is a set Δ' in the consequence of r and $\beta \in P$ such that $\underline{K}, \beta \rhd \Delta'$.

Proof: The assertion can be proved by cases on the rule r: we only discuss some of the relevant cases.

Let $W = \mathbf{T}(c : C \sqcup D)$ and let us assume $\underline{K}, \alpha \rhd \Delta, W$. Then it holds that $\alpha \Vdash c : C \sqcup D$ and, by definition of the semantics, $\alpha \Vdash c : C$ or $\alpha \Vdash c : D$. Supposing that $\alpha \Vdash c : C$, then $\underline{K}, \alpha \rhd \mathbf{T}(c : C)$ and the assertion holds. The other case is similar, since we obtain $\underline{K}, \alpha \rhd \mathbf{T}(c : D)$.

Let $W = \mathbf{T}(c : C \rightarrow D)$ and let us assume $\underline{K}, \alpha \rhd \Delta, W$. If $\underline{K}, \alpha \rhd \mathbf{T}(c : D)$, the assertion holds. Otherwise, it holds that $\underline{K}, \alpha \rhd \mathbf{F}(c : C)$. Given that $\underline{K}$ is finite, there exists $\beta \geq \alpha$ such that $\underline{K}, \beta \rhd \mathbf{F}(c : C)$ and $\underline{K}, \beta \rhd \mathbf{T_s}(c : C)$, which implies $\underline{K}, \beta \rhd \mathbf{T_s}(c : D)$. If $\beta = \alpha$, then $\underline{K}, \beta \rhd \Delta$. Otherwise it holds that $\underline{K}, \beta \rhd \Delta_s$ and the assertion is proved.

Let $W = \mathbf{F}(c : C \rightarrow D)$ and let us assume $\underline{K}, \alpha \rhd \Delta, W$. Then, it holds that $\alpha \nVdash c : C \rightarrow D$ and thus, by definition, there exists a $\beta \geq \alpha$ such that $\underline{K}, \beta \rhd \mathbf{T}(c : C)$ and $\underline{K}, \beta \rhd \mathbf{F}(c : D)$. If $\beta = \alpha$, then $\underline{K}, \beta \rhd \Delta$. Otherwise it holds that $\underline{K}, \beta \rhd \Delta_s$ and the assertion is proved.

Let $W = \mathbf{F}(c : \forall R.C)$ and let us assume $\underline{K}, \alpha \rhd \Delta, W$. Then, by definition, there exists a $\beta \geq \alpha$ and a $q \in \mathcal{D}^\beta$ such that $\underline{K}, \beta \rhd \mathbf{T}((c, q) : R)$ but $\underline{K}, \beta \rhd \mathbf{F}(q : C)$. If $\beta = \alpha$, then $\underline{K}, \beta \rhd \Delta$. Otherwise it holds that $\underline{K}, \beta \rhd \Delta_s$ and the assertion is proved.

Let $W = \mathbf{T}(A \sqsubseteq C)$ and let us assume $\underline{K}, \alpha \rhd \Delta, \mathbf{T}(A \sqsubseteq C)$ and $\underline{K}, \alpha \rhd \mathbf{T}(c : A)$. By definition, given that $\alpha \Vdash A \sqsubseteq C$, then for every $\beta \geq \alpha$ and $d \in \mathcal{D}^\beta$ it holds that $\underline{K}, \beta \rhd \mathbf{T}(d : A)$ implies $\underline{K}, \beta \rhd \mathbf{T}(d : C)$. As $\underline{K}, \alpha \rhd \mathbf{T}(c : A)$, then $\underline{K}, \alpha \rhd \mathbf{T}(c : C)$ and the assertion holds. □

By the previous lemma we directly deduce the Soundness Theorem:

Theorem 3.9 (Soundness)
Let Δ be a set of signed formulas. If Δ is realizable, then Δ is consistent.

Proof: Suppose that Δ is not consistent: then Δ is provable and there exists a closed proof table τ for Δ. If, by absurd, Δ is realizable, then by the previous lemma there must be a leaf Δ_f of τ such that Δ_f is realizable: this is a contradiction, since any Δ_f is a clashing set. Thus, Δ is not realizable, and this concludes the proof. □

3.4 Completeness and termination of $\mathcal{T}_\mathcal{K}$

In this section we prove the completeness of $\mathcal{T}_\mathcal{K}$ and we provide a decision procedure for $\mathcal{KALC}$ based on $\mathcal{T}_\mathcal{K}$. We prove that the result holds for acyclic TBoxes and in the next section we discuss the complexity of the procedure.

Given a set of signed formulas Δ we say that Δ is *acyclic* iff the set of $A \sqsubseteq C$ such that $\mathbf{T}(A \sqsubseteq C) \in \Delta$ is an acyclic TBox, as defined in Definition 2.1. Note that, according to Theorem 3.7, to solve the inference problems w.r.t. an acyclic TBox is equivalent to decide the realizability of an acyclic set.

Thus the completeness result is shown as follows: given a finite acyclic consistent set Δ, we show how we can build in finite time a countermodel for Δ, i.e. a

Kripke model $\underline{K}(\Delta) = \langle P, \leq, \rho, \iota \rangle$ such that $\underline{K}(\Delta), \rho \rhd \Delta$. The following sections detail the steps and components of the construction.

3.4.1 Labelled graphs

Our construction is inspired to the standard technique used for $\mathcal{ALC}$ satisfiability [Baader and Nutt(2003)] based on graph expansion. In our case, a labelled graph $\mathcal{G}$ refers to a world α of the countermodel $\underline{K}(\Delta)$ under construction and it is a representation of the structure of its associated model $\iota(\alpha)$. In particular:

– the nodes of $\mathcal{G}$ form the domain $\mathcal{D}^\alpha$ of α;

– the labelled arcs (c, d, R) of $\mathcal{G}$, where c, d are nodes of $\mathcal{G}$ and the label R is a role name, define the interpretation of R in α;

– each node c is associated with a finite set of signed formulas $\mathcal{S}(c : C)$, representing the concept formulas that must be realized in α.

To get this, we repeatedly apply the following transformation rules on $\mathcal{G}$:

1. Firstly, we apply to $\mathcal{G}$ *expansion rules* as in the standard construction of a downward saturated set. We call *expanded graph* the graph $\mathrm{Exp}(\mathcal{G})$ obtained at the end of this step. $\mathrm{Exp}(\mathcal{G})$ completely describes a world α of $\underline{K}(\Delta)$.

2. Let $\mathcal{G}_e$ be an expanded graph describing a world α. We give rules to compute the *successor graphs* $\mathcal{G}'$ of $\mathcal{G}_e$ so that the expanded graphs $\mathrm{Exp}(\mathcal{G}')$ will be all the immediate successors of α in $\underline{K}(\Delta)$.

We need some care to guarantee the termination. We partition the formulas associated with a node in *primary* and *secondary formulas*. Roughly speaking, primary formulas drive the graph construction: at every step a primary formula or a TBox axiom is selected and the graph is expanded according to the chosen formula. The formulas already considered are collected in the set of secondary formulas, which are no longer considered during the expansion. Dup-formulas require an ad-hoc treatment to avoid infinite loops: for every dup-formula we store the individual names already considered in the expansion procedure, so to apply dup-formula rules at most once for each individual name. The TBox formulas can be seen as "global constraints" on $\mathcal{G}$: they are not affected by the transformation rules, thus we take them apart.

Thus, we can formally define labelled graphs as follows:

Definition 3.10 (Labelled graph)
A labelled graph $\mathcal{G}$ *is a structure of the kind*

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \mathrm{PF}, \mathrm{SF}, \mathrm{TB}, \mathrm{DF} \rangle$$

where:

– $\mathcal{N}$ *is the set of* nodes*, with* $\mathcal{N}$ *a finite subset of* $\mathtt{NI}$*.*

– $\mathcal{E}$ *is the set of* labelled edges (c, d, R)*, with* $c, d \in \mathcal{N}$ *and* $R \in \mathtt{NR}$*.*

– $\textsc{Pf}$ *and* $\textsc{Sf}$ *are functions associating with every node* $c \in \mathcal{N}$ *a finite set of signed formulas* $\mathcal{S}(c : C)$*, called the* primary *and* secondary *formulas of* c *respectively.*

– $\textsc{Tb}$ *is the set* $\mathbf{T}(\mathcal{T}) = \{\mathbf{T}(H) \mid H \in \mathcal{T}\}$*, with* $\mathcal{T}$ *a finite acyclic TBox.*

– $\textsc{Df}$ *is a function mapping a dup-formula to a finite set of nodes.*

Given a graph $\mathcal{G}$, we also define the sets $\textsc{Form}(\mathcal{G})$ and $\textsc{Form}^*(\mathcal{G})$ as:

$$\begin{aligned} \textsc{Form}(\mathcal{G}) &= \textstyle\bigcup_{c \in \mathcal{N}} \textsc{Pf}(c) \cup \{\, \mathbf{T}((c,d) : R) \mid (c,d,R) \in \mathcal{E} \,\} \cup \textsc{Tb} \\ \textsc{Form}^*(\mathcal{G}) &= \textsc{Form}(\mathcal{G}) \cup \textstyle\bigcup_{c \in \mathcal{N}} \textsc{Sf}(c) \end{aligned}$$

In the following we assume that at any step of the countermodel construction a graph $\mathcal{G}$ satisfies the following properties:

(G1) $\textsc{Form}(\mathcal{G})$ is consistent.

(G2) The following closure properties hold:

– If $\mathbf{T}(c : C \sqcap D) \in \textsc{Sf}(c)$, then $\{\mathbf{T}(c : C), \mathbf{T}(c : D)\} \subseteq \textsc{Form}^*(\mathcal{G})$.
– If $\mathbf{F}(c : C \sqcap D) \in \textsc{Sf}(c)$, then $\mathbf{F}(c : C) \in \textsc{Form}^*(\mathcal{G})$ or $\mathbf{F}(c : D) \in \textsc{Form}^*(\mathcal{G})$.
– If $\mathbf{T}(c : C \sqcup D) \in \textsc{Sf}(c)$, then $\mathbf{T}(c : C) \in \textsc{Form}^*(\mathcal{G})$ or $\mathbf{T}(c : D) \in \textsc{Form}^*(\mathcal{G})$.
– If $\mathbf{F}(c : C \sqcup D) \in \textsc{Sf}(c)$, then $\{\mathbf{F}(c : C), \mathbf{F}(c : D)\} \subseteq \textsc{Form}^*(\mathcal{G})$.
– If $\mathbf{T}(c : C \to D) \in \textsc{Sf}(c)$, then $\mathbf{T}(c : D) \in \textsc{Form}^*(\mathcal{G})$ or $\{\mathbf{F}(c : C), \mathbf{T_s}(c : D)\} \subseteq \textsc{Form}^*(\mathcal{G})$.
– If $\mathbf{F}(c : C \to D) \in \textsc{Sf}(c)$, then $\{\mathbf{T}(c : C), \mathbf{F}(c : D)\} \in \textsc{Form}^*(\mathcal{G})$.
– If $\mathbf{T}(c : \exists R.C) \in \textsc{Sf}(c)$, then there is $(c, q, R) \in \mathcal{E}$ s.t. $\mathbf{T}(q : C) \in \textsc{Form}^*(\mathcal{G})$.
– If $W = \mathbf{F}(c : \exists R.C) \in \textsc{Pf}(c)$ and $d \in \textsc{Df}(W)$, then $\mathbf{F}(d : C) \in \textsc{Form}^*(\mathcal{G})$.
– If $W = \mathbf{T}(c : \forall R.C) \in \textsc{Pf}(c)$ and $d \in \textsc{Df}(W)$, then $\mathbf{T}(d : C) \in \textsc{Form}^*(\mathcal{G})$.
– If $\mathbf{F}(c : \forall R.C) \in \textsc{Sf}(c)$, then there is $(c, q, R) \in \mathcal{E}$ s.t. $\mathbf{F}(q : C) \in \textsc{Form}^*(\mathcal{G})$.
– If $W = \mathbf{T}(A \sqsubseteq C) \in \textsc{Tb}$ and $c \in \textsc{Df}(W)$ with $\mathbf{T}(c : A) \in \textsc{Pf}(c)$, then $\mathbf{T}(c : C) \in \textsc{Form}^*(\mathcal{G})$.

Intuitively, the previous conditions guarantee that every step of our construction maintains the consistency of the graphs. Moreover, for every formula considered in their expansion, graphs are closed with respect to the application of the rules of the calculus: that is, the closure properties assure that the result of each step is retained in the formulas of the graph. It is easy to check that the transformation rules provided in the following sections preserve these properties.

3.4.2 Starting graph $\mathcal{G}_\Delta$

Let Δ be a finite acyclic consistent set of signed formulas. The countermodel construction for Δ begins from the *starting graph* $\mathcal{G}_\Delta$

$$\mathcal{G}_\Delta \;=\; \langle\, \mathcal{N}_\Delta,\, \mathcal{E}_\Delta,\, \textsc{Pf}_\Delta,\, \textsc{Sf}_\Delta,\, \textsc{Tb},\, \textsc{Df}_\Delta \,\rangle$$

where:

– $\mathcal{N}_\Delta$ is the set of individual names occurring in Δ;

– $\mathcal{E}_\Delta$ is the set of (c, d, R) such that $\mathbf{T}((c, d) : R) \in \Delta$;

– $\textsc{Pf}_\Delta(c)$ is the set of $\mathcal{S}(c : A) \in \Delta$;

– $\textsc{Sf}_\Delta$ and $\textsc{Df}_\Delta$ map any element to the empty set;

– $\textsc{Tb}$ is the set of $\mathbf{T}(A \sqsubseteq C) \in \Delta$.

One can easily check that $\mathcal{G}_\Delta$ satisfies the properties (G1) and (G2).

3.4.3 Expansion of a graph $\mathcal{G}$

Let $\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \textsc{Pf}, \textsc{Sf}, \textsc{Tb}, \textsc{Df} \rangle$ be a finite graph and $W \in \textsc{Form}(\mathcal{G})$. *Expansion rules* are defined in Figure 3.3. Given W, the corresponding expansion rule transforms $\mathcal{G}$ in a new graph

$$\mathcal{G}' = \langle\, \mathcal{N}',\, \mathcal{E}',\, \textsc{Pf}',\, \textsc{Sf}',\, \textsc{Tb},\, \textsc{Df}' \,\rangle$$

In the rules, $\mathcal{S}_W$ denotes the sign of W. Note that we only indicate the components of the graph that are actually modified: that is, if an element E of $\mathcal{G}$ is not mentioned, it is understood that the corresponding element E' of $\mathcal{G}'$ coincides with E. Note also that, in some cases rules have no effect: for example, in the case $W = \mathbf{F}(c : C \to D)$ the graph is not modified if the set $(\Delta \setminus \{W\}) \cup \{\mathbf{T}(c : C), \mathbf{F}(c : D)\}$ is not consistent.

We repeatedly apply expansion rules to $\mathcal{G}$ until no rule is applicable. We will discuss in Section 3.4.7 that the construction actually terminates. Let $\mathrm{Exp}(\mathcal{G})$ denote the *expanded graph*

$$\mathcal{G}_e = \langle\, \mathcal{N}_e,\, \mathcal{E}_e,\, \textsc{Pf}_e,\, \textsc{Sf}_e,\, \textsc{Tb},\, \textsc{Df}_e \rangle$$

obtained at the end of the expansion step.

Let $\mathrm{Mod}(\mathcal{G}_e)$ be the model $(\mathcal{D}^\alpha, \cdot^\alpha)$ for $\mathcal{L}_{\mathcal{N}_e}$ representing a world α such that:

– $\mathcal{D}^\alpha = \mathcal{N}_e$;

– for every $c \in \mathcal{N}_e$, $c^\alpha = c$;

– for every $A \in \mathtt{NC}$, A^α is the set of c such that $\mathbf{T}(c : A) \in \textsc{Pf}_e(c)$;

– for every $R \in \mathtt{NR}$, R^α is the set of pairs (c, d) such that $(c, d, R) \in \mathcal{E}_e$.

Formula	**Expansion rule**
$W = \mathbf{T}(c : C \sqcap D)$ $W = \mathbf{F}(c : C \sqcup D)$	$\text{P}_\text{F}'(c) = (\text{P}_\text{F}(c) \setminus \{W\}) \cup \{ \mathcal{S}_W(c : C), \mathcal{S}_W(c : D) \}$ $\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$

$W = \mathbf{F}(c : C \sqcap D)$
$W = \mathbf{T}(c : C \sqcup D)$
	If $(\Delta \setminus \{W\}) \cup \{ \mathcal{S}_W(c : C) \}$ is consistent then
	$\text{P}_\text{F}'(c) = (\text{P}_\text{F}(c) \setminus \{W\}) \cup \{ \mathcal{S}_W(c : C) \}$
	$\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$
	else
	$\text{P}_\text{F}'(c) = (\text{P}_\text{F}(c) \setminus \{W\}) \cup \{ \mathcal{S}_W(c : D) \}$
	$\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$

$W = \mathbf{F}(c : C \rightarrow D)$	If $(\Delta \setminus \{W\}) \cup \{ \mathbf{T}(c : C), \mathbf{F}(c : D) \}$ is consistent then $\text{P}_\text{F}'(c) = (\text{P}_\text{F}(c) \setminus \{W\}) \cup \{ \mathbf{T}(c : C), \mathbf{F}(c : D) \}$ $\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$
$W = \mathbf{T}(c : C \rightarrow D)$	If $(\Delta \setminus \{W\}) \cup \{\mathbf{T}(c : D)\}$ is consistent then $\text{P}_\text{F}'(c) = (\text{P}_\text{F}(c) \setminus \{W\}) \cup \{ \mathbf{T}(c : D) \}$ $\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$ else if $(\Delta \setminus \{W\}) \cup \{\mathbf{F}(c : C), \mathbf{T_s}(c : D)\}$ is consistent then $\text{P}_\text{F}'(c) = (\text{P}_\text{F}(c) \setminus \{W\}) \cup \{ \mathbf{F}(c : C), \mathbf{T_s}(c : D) \}$ $\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$
$W = \mathbf{T}(c : \exists R.C)$	Let $q \notin \mathcal{N}$ $\mathcal{N}' = \mathcal{N} \cup \{q\}$ $\quad \mathcal{E}' = \mathcal{E} \cup \{ (c, q, R) \}$ $\text{P}_\text{F}'(c) = \text{P}_\text{F}(c) \setminus \{W\}$ $\quad \text{P}_\text{F}'(q) = \{ \mathbf{T}(q : C) \}$ $\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$ $\quad \text{S}_\text{F}'(q) = \emptyset$
$W = \mathbf{F}(c : \forall R.C)$	Let $q \notin \mathcal{N}$ If $(\Delta \setminus \{W\}) \cup \{ \mathbf{T}((c, q) : R), \mathbf{F}(q : C) \}$ is consistent then $\mathcal{N}' = \mathcal{N} \cup \{q\}$ $\quad \mathcal{E}' = \mathcal{E} \cup \{ (c, q, R) \}$ $\text{P}_\text{F}'(c) = \text{P}_\text{F}(c) \setminus \{W\}$ $\quad \text{P}_\text{F}'(q) = \{ \mathbf{F}(q : C) \}$ $\text{S}_\text{F}'(c) = \text{S}_\text{F}(c) \cup \{W\}$ $\quad \text{S}_\text{F}'(q) = \emptyset$
$W = \mathbf{F}(c : \exists R.C)$ $W = \mathbf{T}(c : \forall R.C)$	Let $d \in \mathcal{N}$ such that $(c, d, R) \in \mathcal{E}$ and $d \notin \text{D}_\text{F}(W)$ $\text{P}_\text{F}'(d) = \text{P}_\text{F}(d) \cup \{ \mathcal{S}_W(d : C) \}$ $\text{D}_\text{F}'(W) = \text{D}_\text{F}(W) \cup \{d\}$
$W = \mathbf{T}(A \sqsubseteq C)$	Let $c \in \mathcal{N} \setminus \text{D}_\text{F}(W)$ If $\mathbf{T}(c : A) \in \text{P}_\text{F}(c)$ then $\text{P}_\text{F}'(c) = \text{P}_\text{F}(c) \cup \{ \mathbf{T}(c : C) \}$ $\text{D}_\text{F}'(W) = \text{D}_\text{F}(W) \cup \{c\}$

Figure 3.3: Expansion rules

Formula	**Successor graph**
$W = \mathbf{F}(c : C \to D)$ $W = \mathbf{T}(c : C \to D)$	$\mathcal{N}' = \mathcal{N} \quad \mathcal{E}' = \mathcal{E} \quad \mathrm{D_F}' = \mathrm{D_{F_s}}$ $\mathrm{P_F}'(c) = (\mathrm{P_F}(c))_s \cup \mathcal{R}_W \qquad \mathrm{P_F}'(d) = (\mathrm{P_F}(d))_s$ for every $d \neq c$ $\mathrm{S_F}'(c) = (\mathrm{S_F}(c))_s \cup \{W\} \qquad \mathrm{S_F}'(d) = (\mathrm{S_F}(d))_s$ for every $d \neq c$ where $\mathcal{R}_{\mathbf{F}(c:C\to D)} = \{\mathbf{T}(c : C), \mathbf{F}(c : D)\}$ and $\mathcal{R}_{\mathbf{T}(c:C\to D)} = \{\mathbf{F}(c : C), \mathbf{T_s}(c : D)\}$
$W = \mathbf{F}(c : \forall R.C)$	Let $q \notin \mathcal{N}$ $\mathcal{N}' = \mathcal{N} \cup \{q\} \quad \mathcal{E}' = \mathcal{E} \cup \{\, (c, q, R) \,\} \quad \mathrm{D_F}' = \mathrm{D_{F_s}}$ $\mathrm{P_F}'(q) = \{\, \mathbf{F}(q : C) \,\} \qquad \mathrm{P_F}'(d) = (\mathrm{P_F}(d))_s$ for every $d \in \mathcal{N}$ $\mathrm{S_F}'(c) = (\mathrm{S_F}(c))_s \cup \{W\} \quad \mathrm{S_F}'(q) = \emptyset$ $\mathrm{S_F}'(e) = (\mathrm{S_F}(e))_s$ for every $e \in \mathcal{N} \setminus \{c\}$

Figure 3.4: Successor rules

3.4.4 Successor of an expanded graph

After the expansion step, that intuitively builds a world in the construction of a countermodel, the *successor rules* are then applied to define the immediate successors of such world.

Let $\mathcal{G}$ be an expanded graph and let W be a formula of $\mathrm{FORM}(\mathcal{G})$. The *successor graph* of $\mathcal{G}$ generated by W is the graph

$$\mathcal{G}' = \langle\, \mathcal{N}',\, \mathcal{E}',\, \mathrm{P_F}',\, \mathrm{S_F}',\, \mathrm{T_B},\, \mathrm{D_F}' \,\rangle$$

defined according to the rules in Figure 3.4. In the rules, $\mathrm{D_{F_s}}(Z) = \mathrm{D_F}(Z)$ if $Z = \mathbf{T}(H)$, otherwise $\mathrm{D_{F_s}}(Z) = \emptyset$.

3.4.5 Countermodel construction

Using the previous definitions, we can now show how to define a finite construction of a countermodel for a realizable set of formulas.

The countermodel for Δ

$$\underline{K}(\Delta) \;=\; \langle\, P,\, \leq,\, \rho,\, \iota \,\rangle$$

is built as follows:

– $\mathrm{Exp}(\mathcal{G}_\Delta)$ belongs to P and it is the root ρ of $\underline{K}(\Delta)$;

– Let $\alpha = \mathcal{G}_\alpha \in P$ and let $\mathcal{G}_1, \ldots, \mathcal{G}_m$ be all the successors of $\mathcal{G}_\alpha$. Then, the graphs

$$\mathrm{Exp}(\mathcal{G}_1),\, \ldots,\, \mathrm{Exp}(\mathcal{G}_m)$$

belong to P and are the immediate successors of α in $\underline{K}(\Delta)$;

– $\leq$ is the reflexive and transitive closure of the immediate successor relation;

– For every $\alpha = \mathrm{Exp}(\mathcal{G}) \in P$, $\iota(\alpha) = \mathrm{Mod}(\mathrm{Exp}(\mathcal{G}))$.

3.4.6 Concepts and formulas measures

The following properties are needed to prove the finiteness of every expanded graph $\mathcal{G}_e$ in the countermodel construction. Let C be a concept, the *depth* of C, denoted by $\text{depth}(C)$, measures the maximum nesting depth of quantifiers in C. Formally:

– $\text{depth}(A) = 0$, with A an atomic concept.

– $\text{depth}(C \odot D) = \max\{\text{depth}(C), \text{depth}(D)\}$, for $\odot \in \{\sqcap, \sqcup, \rightarrow\}$.

– $\text{depth}(\exists R.C) = \text{depth}(\forall R.C) = \text{depth}(C) + 1$.

We extend the definition to signed formulas as follows:

– $\text{depth}(\mathbf{T}((c,d) : R)) = 0$.

– $\text{depth}(\mathcal{S}(c : C)) = \text{depth}(C)$.

– $\text{depth}(\mathbf{T}(C \sqsubseteq D)) = \max\{\text{depth}(C), \text{depth}(D)\}$.

Given a finite set of signed formulas Δ, we define

$$\text{depth}(\Delta) = \max\{\, \text{depth}(H) \mid H \in \Delta \,\}$$

Let

$$\mathcal{T} = \{\, \mathbf{T}(A_1 \sqsubseteq C_1), \ldots, \mathbf{T}(A_n \sqsubseteq C_n) \,\}$$

be an acyclic TBox. We assume that the A_j are pairwise distinct. This assumption does not imply a loss in generality, since we could rewrite any pair of formulas with equal subsumer concept as a single formula: for example, the formulas

$$\mathbf{T}(A \sqsubseteq C_1) \qquad \mathbf{T}(A \sqsubseteq C_2)$$

can be restated as $\mathbf{T}(A \sqsubseteq C_1 \sqcap C_2)$.

Given an atomic concept A, we define the *degree* of A with respect to $\mathcal{T}$, denoted by $\text{dg}_{\mathcal{T}}(A)$, as follows:

– If $A \notin \{A_1, \ldots, A_n\}$, we set $\text{dg}_{\mathcal{T}}(A) = 0$.

– Let $A = A_j$, let $A'_1, \ldots, A'_k$ be all the atomic concepts occurring in C_j and assume that $\text{dg}_{\mathcal{T}}(A'_1), \ldots, \text{dg}_{\mathcal{T}}(A'_k)$ have already been defined. Then:

$$\text{dg}_{\mathcal{T}}(A) = 1 + \max\{\text{dg}_{\mathcal{T}}(A'_1), \ldots, \text{dg}_{\mathcal{T}}(A'_k)\}$$

Since $\mathcal{T}$ is acyclic, $\text{dg}_{\mathcal{T}}(A)$ is defined for every atomic concept A. Intuitively, $\text{dg}_{\mathcal{T}}(A)$ corresponds to the maximum depth of applications of the rule $\mathbf{T} \sqsubseteq$ in a

branch of a proof table for $\{\mathbf{T}(c : A)\} \cup \mathcal{T}$. For example, according to this intuitive explanation, given the following TBox:

$$A \sqsubseteq A_1 \sqcap A_2 \sqcap A_3$$
$$A_1 \sqsubseteq B_1 \quad A_2 \sqsubseteq B_2 \quad A_3 \sqsubseteq B_3$$

we have that:

$$\mathrm{dg}_{\mathcal{T}}(B_1) = \mathrm{dg}_{\mathcal{T}}(B_2) = \mathrm{dg}_{\mathcal{T}}(B_3) = 0$$
$$\mathrm{dg}_{\mathcal{T}}(A_1) = \mathrm{dg}_{\mathcal{T}}(A_2) = \mathrm{dg}_{\mathcal{T}}(A_3) = 1$$
$$\mathrm{dg}_{\mathcal{T}}(A) = 2$$

We notice that, for every atomic concept A, $\mathrm{dg}_{\mathcal{T}}(A) \leq |\mathcal{T}|$.

We extend the definition of $\mathrm{dg}_{\mathcal{T}}$ to signed role and concept formulas as follows:

– $\mathrm{dg}_{\mathcal{T}}(\mathbf{T}((c,d) : R)) = 0$.

– $\mathrm{dg}_{\mathcal{T}}(\mathcal{S}(c : C)) \;=\; \max\{\, \mathrm{dg}_{\mathcal{T}}(A) \mid A \in \mathtt{NC} \text{ and } A \text{ occurs in } C \,\}$.

We remark that we do not need to extended the definition to signed TBox formulas of the kind $\mathbf{T}(A \sqsubseteq C)$. Given a set of signed formulas Δ, we define

$$\mathrm{dg}_{\mathcal{T}}(\Delta) \;=\; \max\{\, \mathrm{dg}_{\mathcal{T}}(H) \mid H \in \Delta \,\}$$

Given a concept C, its *size* $\|C\|$ represents the number of concept constructors occurring in C. Formally, we define it as follows:

– $\|A\| = 0$ for $A \in \mathtt{NC}$ or $A = \bot$;

– $\|C_1 \sqcap C_2\| = \|C_1 \sqcup C_2\| = \|C_1 \rightarrow C_2\| = \|C_1\| + \|C_2\| + 1$;

– $\|\exists R.C'\| = \|\forall R.C'\| = \|C'\| + 1$.

We extend the definition to signed formulas as follows:

– $\|\mathbf{T}((c,d) : R)\| = 0$;

– $\|\mathcal{S}(c : C)\| = \|C\|$;

– $\|\mathbf{T}(C \sqsubseteq D)\| = \|C\| + \|D\| + 1$.

Given a set of signed formulas Δ, we define its size $\|\Delta\|$ as

$$\|\Delta\| = \sum_{H \in \Delta} \|H\|$$

Given a concept C, we define the set $\mathrm{Sub}(C)$ of *subconcepts occurring in* C as:

– $\mathrm{Sub}(A) = A$, for $A \in \mathtt{NC}$ or $A = \bot$;

– $\mathrm{Sub}(C \odot D) = \{C \odot D\} \cup \mathrm{Sub}(C) \cup \mathrm{Sub}(D)$, for $\odot \in \{\sqcap, \sqcup, \rightarrow\}$;

– $\text{Sub}(\mathcal{Q}R.C) = \{\mathcal{Q}R.C\} \cup \text{Sub}(C)$, for $\mathcal{Q} \in \{\exists, \forall\}$.

We extend this definition to signed formulas as follows:

– $\text{Sub}(\mathbf{T}((c,d) : R)) = \emptyset$;

– $\text{Sub}(\mathcal{S}(c : C)) = \text{Sub}(C)$;

– $\text{Sub}(\mathbf{T}(C \sqsubseteq D)) = \text{Sub}(C) \cup \text{Sub}(D)$.

Given a set of signed formulas Δ,

$$\text{Sub}(\Delta) = \bigcup_{W \in \Delta} \text{Sub}(W)$$

3.4.7 Completeness and termination proof

From the previous definitions we are now able to prove that the proposed procedure is complete with respect to the semantics of $\mathcal{KALC}$ and that it is terminating. The termination proof allow us to assert that $\mathcal{KALC}$ is decidable. Let

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \text{PF}, \text{SF}, \text{TB}, \text{DF} \rangle$$

be a finite graph, let $\Delta = \text{FORM}(\mathcal{G})$, and let $\text{Exp}(\mathcal{G})$ be an expanded graph

$$\mathcal{G}_e = \langle \mathcal{N}_e, \mathcal{E}_e, \text{PF}_e, \text{SF}_e, \text{TB}, \text{DF}_e \rangle$$

To simplify the presentation, we consider the graph $\widehat{\mathcal{G}}_e$ having nodes $\mathcal{N}_e$ and edges $\mathcal{E}_e \setminus \mathcal{E}$. In other words, we only consider new edges that have been generated through the expansion of $\mathcal{G}_e$. Note that $\widehat{\mathcal{G}}_e$ is a forest, whereas $\mathcal{G}_e$ might contain cycles deriving from the edges originally contained in $\mathcal{G}$. More precisely, the nodes of $\mathcal{N}$ are the roots of the trees in $\widehat{\mathcal{G}}_e$. The parent of a new node $c \in \mathcal{N}_e \setminus \mathcal{N}$ is the only $b \in \mathcal{N}$ such that c is a *successor* of b in $\widehat{\mathcal{G}}_e$, namely there exists in $\widehat{\mathcal{G}}_e$ an edge (b, c, R). The children of $c \in \mathcal{N}$ are all the successors of c in $\widehat{\mathcal{G}}_e$.

The following results, which are based on these considerations, provide us with an upper bound for $\mathcal{N}_e$.

(P1) Let $c \in \mathcal{N}_e$. Then, the number of children of c in $\widehat{\mathcal{G}}_e$ is at most $|\text{Sub}(\Delta)|$.

Proof: Let d be a successor of c in $\widehat{\mathcal{G}}_e$. Then, d is generated by a signed formula W of the form $\mathbf{T}(c : \exists R.C)$ or $\mathbf{F}(c : \forall S.D)$. More precisely, in the construction of $\mathcal{G}_e$ we get a graph $\mathcal{G}'$ such that W belongs to $\text{PF}'(c)$. In both cases, the concepts $\exists R.C$ or $\forall S.D$ must belong to $\text{Sub}(\Delta)$, hence c has at most $|\text{Sub}(\Delta)|$ successors. □

A *path* π of $\widehat{\mathcal{G}}_e$ is a sequence $(c_1, \ldots, c_n)$ of nodes of $\mathcal{N}_e$ such that, for every $1 < j \leq n$, c_j is a successor of c_{j-1} in $\widehat{\mathcal{G}}_e$. The *length* of π, we denote with $\text{length}(\pi)$, is n. We denote with $\text{FORM}(c)$ the set $\text{PF}(c) \cup \text{SF}(c)$.

(P2) Let $c \in \mathcal{N}_e$, $\mathrm{dg}_{\mathrm{TB}}(\textsc{Form}(c)) = h$, $\mathrm{depth}(\textsc{Form}(c) \cup \mathrm{TB}) = d$ and let π be a path of $\widehat{\mathcal{G}}_e$ starting from c. Then, $\mathrm{length}(\pi) \leq (h+1)(d+1)$.

Proof: We prove (P2) by induction on h. If $h = 0$, then in the construction of π in $\mathcal{G}_e$ the expansion rule for formulas of the kind $\mathbf{T}(A \sqsubseteq C)$ has never been applied; this implies $\mathrm{length}(\pi) \leq \mathrm{depth}(\textsc{Form}(c))$, from which the assertion follows. Let $h > 0$ and suppose

$$\pi = (c, c_2, \ldots, c_n)$$

with $n > d+1$ (if $n \leq d+1$, (P2) trivially holds). We can split π into:

$$\pi_1 = (c, \ldots, c_{d+1}) \quad \text{and} \quad \pi_2 = (c_{d+1}, \ldots, c_n)$$

Since $\mathrm{depth}(\textsc{Form}(c)) \leq d$, the formulas in $\textsc{Form}(c_{d+1})$ are not obtained by decomposing formulas in $\textsc{Form}(c)$, but must be subformulas of formulas of TB. By definition of $\mathrm{dg}_{\mathrm{TB}}$, it follows that

$$\mathrm{dg}_{\mathrm{TB}}(\textsc{Form}(c_{d+1})) < h$$

Let $d' = \mathrm{depth}(\textsc{Form}(c_{d+1}))$; by the induction hypothesis and being $d' \leq d$, we get

$$\mathrm{length}(\pi_2) \leq h(d+1)$$

It follows that $\mathrm{length}(\pi) \leq (d+1) + h(d+1)$, hence

$$\mathrm{length}(\pi) \leq (h+1)(d+1)$$

□

By the previous properties we can prove the following result, which provides an upper bound for the number of nodes of the expanded graph.

(P3) Let $k = |\mathrm{Sub}(\Delta)|$, $d = \mathrm{depth}(\Delta)$ and $n = |\mathrm{TB}|$. Then:

$$|\mathcal{N}_e| \;\leq\; |\mathcal{N}| \frac{k^{(n+1)(d+1)} - 1}{k - 1}$$

Proof: The graph $\widehat{\mathcal{G}}_e$ is composed by $|\mathcal{N}|$ trees $\tau_1, \ldots, \tau_m$, each having as root a node of $\mathcal{N}$. Each tree τ_j has degree at most $k = |\mathrm{Sub}(\Delta)|$ by (P1) and height at most $(n+1)(d+1)$ by (P2). Since a k-ary complete tree of height h contains $\frac{k^h - 1}{k-1}$ nodes, the above upper bound follows. □

By the previous properties, it follows that $\mathcal{G}_e$ does not contain infinite paths starting from a node $c_0 \in \mathcal{N}_e \setminus \mathcal{N}$.

Lemma 3.11
Let $\mathcal{G}'$ be obtained by applying to $\mathcal{G}$ one of the rules of Figure 3.3 and Figure 3.4 defined by W. Then, one of the following facts holds:

(1) $\textsc{Form}(\mathcal{G}')$ *is obtained by replacing W with one or more formulas W' such that $\|W'\| < \|W\|$, possibly substituting $\mathbf{T_s}$ with $\mathbf{T}$ and discharging the $\mathbf{F}$-formulas.*

(2) If W is a dup-formula, $\textsc{Form}(\mathcal{G}') = \textsc{Form}(\mathcal{G}) \cup \{W'\}$*, with* $\|W'\| < \|W\|$*, and* $\textsc{Df}(W) \subset \textsc{Df}'(W)$.

Proof: We simply note that Point (1) applies to the expansion and successor rules for non dup-formulas. As can be easily verified by cases, in these rules for any newly added formula W' it holds that $\|W'\| < \|W\|$.

In the case of dup-formulas Point (2) holds. In fact, after the application of the rule for $W = \mathbf{F}(c : \exists R.C)$ and $W = \mathbf{T}(c : \forall R.C)$ it holds that

$$\textsc{Form}(\mathcal{G}') = \textsc{Form}(\mathcal{G}) \cup \{W' = \mathcal{S}_W(d : C)\}$$

with $(c, d, R) \in \mathcal{E}$ and $\|W'\| < \|W\|$; moreover

$$\textsc{Df}'(W) = \textsc{Df}(W) \cup \{d\}$$

After the application of the rule for $W = \mathbf{T}(A \sqsubseteq C)$ it holds that

$$\textsc{Form}(\mathcal{G}') = \textsc{Form}(\mathcal{G}) \cup \{W' = \mathbf{T}(d : C)\}$$

with $\mathbf{T}(d : A) \in \textsc{Pf}(d)$ and $\|W'\| < \|W\|$; also in this case

$$\textsc{Df}'(W) = \textsc{Df}(W) \cup \{d\}$$

□

Let $\Delta = \textsc{Form}(\mathcal{G})$ and $\textsc{Exp}(\mathcal{G}) = \mathcal{G}_e$. We can now give a measure for $\textsc{Form}^*(\mathcal{G}_e)$ by giving a measure of the number of applications of the expansion rules to Δ and the size of formulas added by each rule. Let d-Sf(Δ) be the set of dup-subformulas of Δ and nd-Sf(Δ) be the set of non dup-subformulas of Δ. Obviously, we have:

$$|\text{d-Sf}(\Delta)| \leq \|\Delta\| \quad \text{and} \quad |\text{nd-Sf}(\Delta)| \leq \|\Delta\|$$

We can show that:

(P4) The expansion rules for non dup-formulas add at most $\|\Delta\|$ formulas to $\textsc{Form}(\mathcal{G}_e)$.

Proof: Every rule for non dup-formulas is applied at most once for every element of nd-Sf(Δ): thus we have at most $\|\Delta\|$ applications of such rules during the expansion of $\mathcal{G}_e$. By Lemma 3.11, every application of such rules to a formula H can lead to the addition to $\textsc{Form}(\mathcal{G}_e)$ of one or more subformulas H' of H with $\|H'\| < \|H\|$: since there exist at most $\|\Delta\|$ of such subformulas, the application of every possible expansion rule to $\mathcal{G}_e$ adds $\|\Delta\|$ formulas to $\textsc{Form}(\mathcal{G}_e)$ at most. □

(P5) The expansion rules for dup-formulas add at most $|\mathcal{N}_e| \cdot \|\Delta\|$ formulas to $\textsc{Form}(\mathcal{G}_e)$.

Proof: By their definition, for every dup-formula in d-Sf(Δ), the corresponding rule is applied at most one time for every node in $\mathcal{N}_e$ (indeed, the map $\textsc{Df}$ is used to control that each dup-formula is applied at most once for each individual name). Hence, the expansion rules for dup-formulas are applied at most $|\mathcal{N}_e| \cdot \|\Delta\|$ times in the expansion of $\mathcal{G}_e$. By Lemma 3.11, every application of such rules to a formula H can lead to the addition to $\textsc{Form}(\mathcal{G}_e)$ of a single subformula H' of H: then, the application of dup-formulas can at most add $|\mathcal{N}_e| \cdot \|\Delta\|$ formulas to $\textsc{Form}(\mathcal{G}_e)$. □

By the previous properties, we can conclude the finiteness of the expanded graph:

Lemma 3.12
Given a finite labelled graph $\mathcal{G}$, $\mathrm{Exp}(\mathcal{G})$ *is finite.*

Proof: First of all, $\mathcal{N}_e$ and $\mathcal{E}_e$ are finite: this directly follows from (P3) and the finiteness of $\mathcal{N}$ and $\mathcal{E}$.

Moreover, by the properties (P4) and (P5), we have that $\textsc{Form}(\mathcal{G}_e)$ is finite since the number of formulas added by expansion rules is finite and the number of rule applications is bounded. Similarly, we can show that, for every $c \in \mathcal{N}_e$, $\mathrm{SF}_e(c)$ is finite, since every application of the expansion rules adds at most one formula to the secondary formulas of $\mathcal{G}_e$: hence, we have that $\textsc{Form}^*(\mathcal{G}_e)$ is finite. We remark that, obviously, also the map DF_e is finite since it is defined between dup-formulas in $\textsc{Form}^*(\mathcal{G}_e)$ and finite subsets of $\mathcal{N}_e$, both of which are proved to be finite from our previous considerations. □

By now, we have proved that every expanded graph can be constructed in finite time: in the following we prove that the number of worlds in $\underline{K}(\Delta)$ is finite.

Let us assume the following facts:

(A1) Given an expanded graph $\mathcal{G}_\alpha$, consider the graph

$$\mathcal{G}_\beta = \mathrm{Exp}(\mathcal{G}'_\alpha)$$

where $\mathcal{G}'_\alpha$ is obtained from $\mathcal{G}_\alpha$ by applying one of the successor rules of Figure 3.4. In the countermodel construction, the world $\beta = \mathcal{G}_\beta$ represents an immediate successor of the world $\alpha = \mathcal{G}_\alpha$.

(A2) Let us assume that $\mathcal{G}'_\alpha$ is obtained from $\mathcal{G}_\alpha$ by the application of the rule associated with the signed formula W over the individual name c:

$$W = \mathcal{S}(c : \mathrm{C})$$

Consider the graph $\widehat{\mathcal{G}_\beta}$, having nodes in $\mathcal{N}_\beta$ and edges $\mathcal{E}_\beta \setminus \mathcal{E}_\alpha$. Then, in $\widehat{\mathcal{G}_\beta}$ the node c is the root of a tree τ_c. The nodes in $\widehat{\mathcal{G}_\beta}$ can be partitioned as $\mathcal{N}_\alpha \cup \{c\} \cup \mathcal{N}_c$ where:

- $\mathcal{N}_\alpha$ are the nodes of $\mathcal{G}_\alpha$ different from c.
- $\mathcal{N}_c$ are the new nodes generated in the construction of $\mathcal{G}_\beta$. In particular, they are the descendant of c in the tree τ_c and they are generated either by the expansion steps towards $\mathcal{G}_\beta$ or by the application of the successor rule on W.

(A3) Let

$$\Delta_\alpha = \bigcup_{d \in \mathcal{N}_\alpha} \mathrm{PF}_\alpha(d)$$

with PF_α the primary formulas map associated with $\mathcal{G}_\alpha$. Note that, as $\mathcal{G}_\alpha$ is an expanded graph, Δ_α only contains:

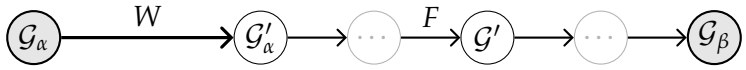

Figure 3.5: Assumptions for Proposition 3.13

- signed dup-formulas of the kind $\mathbf{F}(d : \exists R.C')$ and $\mathbf{T}(d : \forall R.C')$;
- signed atomic formulas;
- $\mathbf{T_s}$-formulas, i.e. signed formulas of the form $\mathbf{T_s}(d : C')$;
- signed formulas of the kind $\mathcal{S}(d : C' \rightarrow D')$ or $\mathbf{F}(d : \forall R.C')$.

(A4) Let

$$\mathcal{G}' = \langle\, \mathcal{N}', \mathcal{E}', \mathrm{PF}', \mathrm{SF}', \mathrm{TB}, \mathrm{DF}' \,\rangle$$

be one of the graphs obtained in the expansion of $\mathcal{G}'_\alpha$, during the construction of $\mathcal{G}_\beta$. Moreover, let us assume that $\mathcal{G}'$ is obtained by applying an expansion rule on the signed formula F.

We represent in Figure 3.5 the succession of steps leading the construction of $\mathcal{G}_\beta$ from $\mathcal{G}_\alpha$. In the figure we depict expanded graphs $\mathcal{G}_\alpha$ and $\mathcal{G}_\beta$ with thick borders and gray background. We distinguish successor rules applications from expansion rules applications with a thicker arrow line. Arrow labels, when present, describe the formula over which the rule is applied.

We can prove the following facts:

Proposition 3.13
Using the assumptions (A1) – (A4), it holds that:

1. *Let $Z = \mathcal{S}(c : D) \in \mathrm{PF}'(c)$ in $\mathcal{G}'$. Then, one of the following properties holds:*
   - *(i) $Z \in \mathrm{PF}_\alpha(c)$ or $Z = \mathbf{T}(c : D)$ with $\mathbf{T_s}(c : D) \in \mathrm{PF}_\alpha(c)$;*
   - *(ii) $\|Z\| < \|\Delta_\alpha\|$;*
   - *(iii) $dg_{\mathrm{TB}}(Z) < dg_{\mathrm{TB}}(\Delta_\alpha)$.*
2. *Let d be a descendant of c in τ_c and let $V = \mathcal{S}(d : D) \in \mathrm{PF}'(d)$ in $\mathcal{G}'$. Then, one of the following properties holds:*
   - *(i) $\|V\| < \|\Delta_\alpha\|$;*
   - *(ii) $dg_{\mathrm{TB}}(V) < dg_{\mathrm{TB}}(\Delta_\alpha)$.*

Proof:

1. Let $\mathcal{G}' = \mathcal{G}'_\alpha$, that is no expansion rule has been applied on $\mathcal{G}'_\alpha$. Then $\mathrm{PF}'(c)$ is equal to $\mathrm{PF}_\alpha(c)$ minus the formulas modified in the application of the successor rule on W. In particular, this application modifies $\mathrm{PF}_\alpha(c)$ by adding one or more formulas W', discharging **F**-formulas and substituting $\mathbf{T_s}$-formulas with

the corresponding **T**-formulas. In the case of the modified $\mathbf{T_s}$-formulas and the remaining formulas of $\mathrm{PF}_\alpha(c)$, the assertion holds for Point (i). In the case of the newly added formulas W', the assertion holds by Point (ii), since by Lemma 3.11 we have that $\|W'\| < \|W\|$ and $W \in \mathrm{PF}_\alpha(c)$.

Now, let us assume, without loss of generality, that Z is a formula added to $\mathrm{PF}'(c)$ in the last expansion step of $\mathcal{G}'$: we proceed by induction on the form of the formula F.

Let $F = \mathbf{T}(A' \sqsubseteq C')$, then $Z = \mathbf{T}(c : C')$. We have that $c \notin \mathrm{DF}'_\alpha(F)$ and $\mathbf{T}(c : A') \in \mathrm{PF}'(c)$ in $\mathcal{G}'$. Since $\mathcal{G}_\alpha$ is an expanded graph, $\mathbf{T}(c : A') \notin \mathrm{PF}_\alpha(c)$ but it is obtained during the expansion of $\mathcal{G}'_\alpha$ either by decomposing W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in $\mathrm{PF}_\alpha(c)$. Hence, in both cases $\mathrm{dg}_{\mathrm{TB}}(Z) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

Let $F = \mathcal{S}(c : C' \sqcap D')$ or $F = \mathcal{S}(c : C' \sqcup D')$, then $Z = \mathcal{S}(c : C')$ or $Z = \mathcal{S}(c : D')$. By the form of Δ_α, F can only be generated from a previous rule application over W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in $\mathrm{PF}_\alpha(c)$. In both cases, we have that $\|Z\| < \|\Delta_\alpha\|$ or $\mathrm{dg}_{\mathrm{TB}}(Z) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

Let $F = \mathbf{F}(c : C' \to D')$, then $Z = \mathbf{T}(c : C')$ or $Z = \mathbf{F}(c : D')$. Note that $F \notin \mathrm{PF}_\alpha(c)$, since it would be discharged by the application of the successor rule on W. Then, F is generated from a previous rule application over W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in $\mathrm{PF}_\alpha(c)$. Hence, in both cases $\|Z\| < \|\Delta_\alpha\|$ or $\mathrm{dg}_{\mathrm{TB}}(Z) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

Let $F = \mathbf{T}(c : C' \to D')$, then $Z \in \{\mathbf{T}(c : C'), \mathbf{F}(c : C'), \mathbf{T_s}(c : D')\}$. If $F \in \mathrm{PF}_\alpha(c)$ then $F \neq W$ and, since $\mathcal{G}_\alpha$ is a fully expanded graph, $Z \notin \mathrm{PF}_\alpha(c)$: this implies that $\|Z\| < \|F\|$ and thus $\|Z\| < \|\Delta_\alpha\|$. Otherwise, if $F \notin \mathrm{PF}_\alpha(c)$, then F is generated from a previous rule application over W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in $\mathrm{PF}_\alpha(c)$. Hence $\|Z\| < \|\Delta_\alpha\|$ or $\mathrm{dg}_{\mathrm{TB}}(Z) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

Let $F = \mathbf{F}(d : \exists R.C')$ or $F = \mathbf{T}(d : \forall R.C')$ with $(d, c, R) \in \mathcal{E}'$, then $Z = \mathcal{S}_W(c : C')$. Since edges of the kind (d, c, R) are not generated during any of the steps of the construction of $\mathcal{G}_\beta$, then $(d, c, R) \in \mathcal{E}_\alpha$ and F is a formula generated by a previous expansion step on a formula $\mathbf{T}(d : B)$ with $\mathbf{T}(d : B)$ or $\mathbf{T_s}(d : B)$ in Δ_α. Thus, we have that $\|Z\| < \|\Delta_\alpha\|$ or $\mathrm{dg}_{\mathrm{TB}}(Z) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

2. Let $\mathcal{G}' = \mathcal{G}'_\alpha$, that is no expansion rule has been applied on $\mathcal{G}'_\alpha$. Then $\mathrm{PF}'(d) \neq \emptyset$ only if $W = \mathbf{F}(c : \forall R.C)$: in this case, $\mathrm{PF}'(d) = \{\mathbf{F}(d : C)\}$. Hence, by definition, $\|V\| < \|W\|$ and thus $\|V\| < \|\Delta_\alpha\|$.

   Now, let us assume, without loss of generality, that V is a formula added to $\mathrm{PF}'(d)$ in the last expansion step of $\mathcal{G}'$: we proceed by induction on the form of the formula F.

   Let $F = \mathbf{T}(A' \sqsubseteq C')$, then $V = \mathbf{T}(d : C')$. We have that $d \notin \mathrm{DF}'_\alpha(F)$ and $\mathbf{T}(d : A') \in \mathrm{PF}'(c)$ in $\mathcal{G}'$. Since $d \notin \mathcal{N}_\alpha$, $\mathbf{T}(d : A') \notin \Delta_\alpha$ but it is obtained during the expansion of $\mathcal{G}'_\alpha$ by decomposing W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in $\mathrm{PF}_\alpha(c)$. Hence, in both cases $\mathrm{dg}_{\mathrm{TB}}(V) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

Let $F = \mathcal{S}(d : C' \sqcap D')$ or $F = \mathcal{S}(d : C' \sqcup D')$, then $V = \mathcal{S}(d : C')$ or $V = \mathcal{S}(d : D')$. By the form of Δ_α, F can only be generated from a previous rule application over W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in $\mathrm{PF}_\alpha(c)$. In both cases, we have that $\|V\| < \|\Delta_\alpha\|$ or $\mathrm{dg}_{\mathrm{TB}}(V) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$. We also note that the cases for $F = \mathcal{S}(d : C' \rightarrow D')$ are similar.

Let $F = \mathbf{T}(q : \forall R.C')$ or $F = \mathbf{F}(q : \exists R.C')$ with q in τ_c and $(q, d, R) \in \mathcal{E}'$, then $V = \mathcal{S}_W(d : C')$. Every element q of τ_c either is c or is generated in the construction of $\mathcal{G}_\beta$: thus F is a dup-formula of $\mathrm{PF}_\alpha(c)$ or it is generated by a previous step on a formula over the individual name c, that is W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in Δ_α. Thus, we have that $\|V\| < \|\Delta_\alpha\|$ or $\mathrm{dg}_{\mathrm{TB}}(V) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

Let $F = \mathbf{T}(q : \exists R.C')$ or $F = \mathbf{F}(q : \forall R.C')$ with q in τ_c, then $V = \mathcal{S}_W(d : C')$. Note that this is the expansion step in which the individual d is introduced in $\mathcal{N}'$. As above, since every element q of τ_c is c or it is generated in the construction of $\mathcal{G}_\beta$, then F is a formula generated by a previous step on a formula over the individual name c, that is W or a formula $\mathbf{T}(c : B)$ with $\mathbf{T}(c : B)$ or $\mathbf{T_s}(c : B)$ in Δ_α. Hence, $\|V\| < \|\Delta_\alpha\|$ or $\mathrm{dg}_{\mathrm{TB}}(V) < \mathrm{dg}_{\mathrm{TB}}(\Delta_\alpha)$.

□

From these properties, one can prove that the depth of α in $\underline{K}(\Delta)$ is bounded. Intuitively, Proposition 3.13 defines a well-founded measure that monotonically decreases from a graph to its successors.

We now state the main results of this section.

Lemma 3.14
Let Δ be a finite acyclic consistent set of signed formulas.

(i) The model $\underline{K}(\Delta)$ is finite.

(ii) Let $\alpha = \mathrm{Mod}(\mathcal{G}_\alpha)$ be a world of $\underline{K}(\Delta)$. Then, $\underline{K}(\Delta), \alpha \rhd \textsc{Form}^*(\mathcal{G}_\alpha)$.

(iii) $\underline{K}(\Delta), \rho \rhd \Delta$.

Proof:

(i) The assertion directly follows by Lemmas 3.11, 3.12 and Proposition 3.13. In fact, for every $\alpha \in P$, $\iota(\alpha) = \mathrm{Mod}(\mathcal{G}_\alpha)$ is finite by Lemma 3.12. Moreover, from Lemma 3.11 and Proposition 3.13, if $\beta \in P$ is a successor of α, passing from the set of formulas of $\mathcal{G}_\alpha$ to the set of $\mathcal{G}_\beta$, there is at least a formula for which one between its size and degree w.r.t. TB decreases. Given that the starting set Δ is finite, this implies that it is not possible to build an infinite sequence of successors.

(ii) The fact can be shown by proving that $W \in \textsc{Form}^*(\mathcal{G}_\alpha)$ implies $\underline{K}(\Delta), \alpha \rhd W$. We show this by cases on the structure of the formula W and by induction on the definition of concept for concept formulas.

Let $W = \mathbf{T}((c,d) : R)$. Thus $W \in \{\mathbf{T}((c,d) : R) \mid (c,d,R) \in \mathcal{E}_\alpha\}$. By definition, R^α is the set of (c,d) such that $(c,d,R) \in \mathcal{E}_\alpha$, thus it holds that $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{T}(c : A)$ with $A \in \mathtt{NC}$. Note that, by the definition of expansion rules, atomic formulas of this kind can only appear in $\mathrm{PF}(c)$. By the definition of $\mathrm{Mod}(\mathcal{G}_\alpha)$, A^α is the set of c such that $\mathbf{T}(c : A) \in \mathrm{PF}_\alpha(c)$: it follows that $\underline{K}(\Delta), \alpha \rhd W$. For the same reasons, if $W = \mathbf{F}(c : A)$, $W \in \mathrm{PF}_\alpha(c)$: given that by (G1) $\textsc{Form}(\mathcal{G}_\alpha)$ is consistent, $\mathbf{T}(c : A) \notin \mathrm{PF}_\alpha(c)$, and hence $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{F}(c : \bot)$. Then as above, W can only appear in $\mathrm{PF}(c)$. Given that, by (G1) $\textsc{Form}(\mathcal{G}_\alpha)$ is consistent, $\mathbf{T}(c : \bot) \notin \mathrm{PF}_\alpha(c)$, and hence $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{T}(c : C \sqcap D)$ or $W = \mathbf{F}(c : C \sqcup D)$. By (G2) and the expansion rules for W, $\{\mathcal{S}_W(c : C), \mathcal{S}_W(c : D)\} \subseteq \textsc{Form}^*(\mathcal{G}_\alpha)$. By induction hypothesis, $\underline{K}(\Delta), \alpha \rhd \mathcal{S}_W(c : C)$ and $\underline{K}(\Delta), \alpha \rhd \mathcal{S}_W(c : D)$ and hence $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{T}(c : C \sqcup D)$ or $W = \mathbf{F}(c : C \sqcap D)$. By (G2) and the expansion rules for W, $\mathcal{S}_W(c : C) \in \textsc{Form}^*(\mathcal{G}_\alpha)$ or $\mathcal{S}_W(c : D) \in \textsc{Form}^*(\mathcal{G}_\alpha)$. By induction hypothesis $\underline{K}(\Delta), \alpha \rhd \mathcal{S}_W(c : C)$ or $\underline{K}(\Delta), \alpha \rhd \mathcal{S}_W(c : D)$, hence $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{F}(c : C \to D)$. By (G2) and the expansion rules for W, if $W \in \mathrm{SF}(c)$, $\{\mathbf{T}(c : C), \mathbf{F}(c : D)\} \subseteq \textsc{Form}^*(\mathcal{G}_\alpha)$. By induction hypothesis, $\underline{K}(\Delta), \alpha \rhd \mathbf{T}(c : C)$ and $\underline{K}(\Delta), \alpha \rhd \mathbf{F}(c : D)$, which implies $\underline{K}(\Delta), \alpha \rhd W$. On the other hand, if $W \in \mathrm{PF}(c)$, by the successor rule for W, there exists a world $\beta > \alpha$ such that $W \in \mathrm{SF}_\beta(c)$ and, as in the previous case, $\underline{K}(\Delta), \beta \rhd W$. Hence, $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{T}(c : C \to D)$. By (G2) and the expansion rules for W, if $W \in \mathrm{SF}(c)$, $\{\mathbf{T}(c : D)\} \subseteq \textsc{Form}^*(\mathcal{G}_\alpha)$ or $\{\mathbf{F}(c : C), \mathbf{T_s}(c : D)\} \subseteq \textsc{Form}^*(\mathcal{G}_\alpha)$. By induction hypothesis, $\underline{K}(\Delta), \alpha \rhd \mathbf{T}(c : D)$ or $\underline{K}(\Delta), \alpha \rhd \mathbf{F}(c : C), \mathbf{T_s}(c : D)$, which in both cases imply $\underline{K}(\Delta), \alpha \rhd W$. If $W \in \mathrm{PF}(c)$, by the successor rule for W, there exists a world $\beta > \alpha$ such that $W \in \mathrm{SF}_\beta(c)$ and, as in the previous case, $\underline{K}(\Delta), \beta \rhd W$. Hence, $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{T}(c : \exists R.C)$. By (G2) and the expansion rules for W, there exists $q \in \mathcal{N}_\alpha$ such that $(c,q,R) \in \mathcal{E}_\alpha$ and $\mathbf{T}(q : C) \in \textsc{Form}^*(\mathcal{G}_\alpha)$. This implies that there exists $q \in \mathcal{D}_\alpha$ such that $\underline{K}(\Delta), \alpha \rhd \mathbf{T}((c,q) : R)$ and $\underline{K}(\Delta), \alpha \rhd \mathbf{T}(q : C)$ and hence $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{F}(c : \exists R.C)$ or $W = \mathbf{T}(c : \forall R.C)$. By (G2) and the expansion rules for W, for every $\beta \geq \alpha$ and every $d \in \mathrm{DF}_\beta(W)$, $\mathcal{S}_W(d : C) \in \textsc{Form}^*(\mathcal{G}_\beta)$. As a result, by induction hypothesis, for every $\beta \geq \alpha$ and $d \in \mathcal{D}^\beta$, such that $\underline{K}(\Delta), \beta \rhd \mathbf{T}((c,d) : R)$, it holds $\underline{K}(\Delta), \beta \rhd \mathcal{S}_W(d : C)$ and hence $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{F}(c : \forall R.C)$. By (G2) and the expansion rules for W, if $W \in \mathrm{SF}(c)$, there exists $(c,q,R) \in \mathcal{E}_\alpha$ such that $\{\mathbf{F}(q : C)\} \subseteq \textsc{Form}^*(\mathcal{G}_\alpha)$. It follows that there exists $q \in \mathcal{D}^\alpha$ such that $\underline{K}(\Delta), \alpha \rhd \mathbf{T}((c,q) : R)$ and $\underline{K}(\Delta), \alpha \rhd \mathbf{F}(q : C)$. By definition, this implies $\underline{K}(\Delta), \alpha \rhd W$. On the other hand, if $W \in \mathrm{PF}(c)$, by the successor rule for W, there exists a world $\beta > \alpha$ such that $W \in \mathrm{SF}_\beta(c)$ and, as in the previous case, $\underline{K}(\Delta), \beta \rhd W$. Hence, it holds that $\underline{K}(\Delta), \alpha \rhd W$.

Let $W = \mathbf{T}(A \sqsubseteq C)$. Then, $W \in \text{TB}$: by (G2) and the expansion rules for W, for every world $\beta \geq \alpha$ and every $c \in \text{DF}_\beta(W)$, if $\mathbf{T}(c : A) \in \text{PF}_\beta(c)$, then $\mathbf{T}(c : C) \in \text{FORM}^*(\mathcal{G}_\beta)$. As a result, by induction hypothesis, for every $\beta \geq \alpha$ and $c \in \mathcal{D}^\beta$, $\underline{K}(\Delta), \beta \rhd \mathbf{T}(c : A)$ implies $\underline{K}(\Delta), \beta \rhd \mathbf{T}(c : C)$. Hence, it holds that $\underline{K}(\Delta), \alpha \rhd W$.

(iii) Since $\Delta \subseteq \text{FORM}(\mathcal{G}_0)$, by Point (ii) we get $\underline{K}(\Delta), \rho \rhd \Delta$.

□

By the previous lemma and by the Soundness Theorem, we conclude:

Theorem 3.15 (Completeness)
Let Δ be a finite acyclic set of signed formulas. Δ is realizable iff Δ is consistent. □

The countermodel construction procedure can be used to decide the realizability of an acyclic Δ. Indeed, one tries to build $\underline{K}(\Delta)$ by applying the transformation rules in all possible ways; by Lemma 3.11, the search space is finite. If all the attempts fail, yielding to a clashing set $\text{PF}(c)$, Δ is not consistent (Lemma 3.14), hence it is not realizable (Theorem 3.15). In this case, the failed branches correspond to the branches of a closed proof table for Δ.

In the following example we show how to use such procedure to build a countermodel from a set of realizable signed formulas.

Example 4 (Countermodel construction)
Let us consider again the ABox $\mathcal{A}$ of Example 1:

a:Company	(a,b):hasCustomer	b:Insolvent
b:Company	(a,c):hasCustomer	d : ¬Insolvent
c:Company	(b,c):hasCustomer	a : $D \rightarrow$ CW
d:Company	(c,d):hasCustomer	

where D is the concept:

$$\exists \texttt{hasCustomer}.(\texttt{Insolvent} \sqcap \exists \texttt{hasCustomer}.\neg \texttt{Insolvent})$$

We show how our procedure can build a countermodel for the set $\Delta = \mathbf{T}(\mathcal{A}) \cup \{\mathbf{F}(\texttt{a:CW})\}$, proving that a:CW is not a $\mathcal{KALC}$-logical consequence of $\mathcal{A}$. In fact, at the end of the construction, we obtain the finite countermodel $\underline{K}$ given in Example 1.

We start the construction by defining the starting graph

$$\mathcal{G}_\Delta \;=\; \langle\, \mathcal{N}_\Delta,\, \mathcal{E}_\Delta,\, \text{PF}_\Delta,\, \text{SF}_\Delta,\, \emptyset,\, \text{DF}_\Delta \,\rangle$$

where:

– $\mathcal{N}_\Delta = \mathcal{N} = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$;

– $\mathcal{E}_\Delta = \{(\mathtt{a}, \mathtt{b}, R), (\mathtt{a}, \mathtt{c}, R), (\mathtt{b}, \mathtt{c}, R), (\mathtt{c}, \mathtt{d}, R)\}$ with $R = \mathtt{hasCustomer}$;

– PF_Δ is defined by the map:

$$\begin{aligned}
\mathtt{a} &\mapsto \{\, \mathbf{T}(\mathtt{a}\!:\!\mathtt{Company}), \mathbf{T}(\mathtt{a}\!:\!D \rightarrow \mathtt{CW}), \mathbf{F}(\mathtt{a}\!:\!\mathtt{CW}) \,\} \\
\mathtt{b} &\mapsto \{\, \mathbf{T}(\mathtt{b}\!:\!\mathtt{Company}), \mathbf{T}(\mathtt{b}\!:\!\mathtt{Insolvent}) \,\} \\
\mathtt{c} &\mapsto \{\, \mathbf{T}(\mathtt{c}\!:\!\mathtt{Company}) \,\} \\
\mathtt{d} &\mapsto \{\, \mathbf{T}(\mathtt{d}\!:\!\mathtt{Company}), \mathbf{T}(\mathtt{d}\!:\!\neg\mathtt{Insolvent}) \,\}
\end{aligned}$$

– SF_Δ and DF_Δ map every element of $\mathcal{N}_\Delta$ to the empty set.

We now proceed to expand the initial graph to the expanded graph $\mathrm{Exp}(\mathcal{G}_\Delta) = \mathcal{G}'_\Delta$ that represents the root of our final countermodel. Following the conditions on the expansion rules, a possible application sequence is:

– Let $W = \mathbf{T}(\mathtt{a}\!:\!D \rightarrow \mathtt{CW})$: we fall in the second case, thus $\mathbf{F}(\mathtt{a}\!:\!D)$ and $\mathbf{T_s}(\mathtt{a}\!:\!\mathtt{CW})$ are added to $\mathrm{PF}(\mathtt{a})$;

– Let

$$W = \mathbf{F}(\mathtt{a}\!:\!\exists\mathtt{hasCustomer}.(\mathtt{Insolvent} \sqcap \exists\mathtt{hasCustomer}.\neg\mathtt{Insolvent}))$$

For $t \in \{\mathtt{b}, \mathtt{c}\}$, $\mathbf{F}(t : \mathtt{Insolvent} \sqcap \exists\mathtt{hasCustomer}.\neg\mathtt{Insolvent})$ is added to $\mathrm{PF}(t)$ and $\mathrm{DF}(W) = \{\mathtt{b}, \mathtt{c}\}$;

– Let

$$W = \mathbf{F}(\mathtt{b}\!:\!\mathtt{Insolvent} \sqcap \exists\mathtt{hasCustomer}.\neg\mathtt{Insolvent})$$

We fall in the second case, thus the formula $\mathbf{F}(\mathtt{b}\!:\!\exists\mathtt{hasCustomer}.\neg\mathtt{Insolvent})$ is added to $\mathrm{PF}(\mathtt{b})$;

– Let

$$W = \mathbf{F}(\mathtt{c}\!:\!\mathtt{Insolvent} \sqcap \exists\mathtt{hasCustomer}.\neg\mathtt{Insolvent})$$

We can choose the first case, thus the formula $\mathbf{F}(\mathtt{c}\!:\!\mathtt{Insolvent})$ is added to $\mathrm{PF}(\mathtt{c})$;

– Let

$$W = \mathbf{F}(\mathtt{b}\!:\!\exists\mathtt{hasCustomer}.\neg\mathtt{Insolvent})$$

The rule adds $\mathbf{F}(\mathtt{c}\!:\!\neg\mathtt{Insolvent})$ to $\mathrm{PF}(\mathtt{c})$ and $\mathrm{DF}(W) = \{\mathtt{c}\}$.

The expansion stops here, since trying to apply the rules for $\mathbf{F}(\mathtt{c}\!:\!\neg\mathtt{Insolvent})$ or $\mathbf{T}(\mathtt{d}\!:\!\neg\mathtt{Insolvent})$ would violate the consistency conditions in the definition of these rules. The obtained expanded graph

$$\mathcal{G}'_\Delta = \langle\, \mathcal{N}'_\Delta,\, \mathcal{E}'_\Delta,\, \mathrm{PF}'_\Delta,\, \mathrm{SF}'_\Delta,\, \emptyset,\, \mathrm{DF}'_\Delta \,\rangle$$

is composed as follows:

– $\mathcal{N}'_\Delta = \mathcal{N}_\Delta$, $\mathcal{E}'_\Delta = \mathcal{E}_\Delta$;

– PF'_Δ is defined by the map:

$$\begin{aligned}
\texttt{a} &\mapsto \{\, \mathbf{T}(\texttt{a:Company}), \mathbf{F}(\texttt{a:CW}), \mathbf{F}(\texttt{a:}D), \mathbf{T_s}(\texttt{a:CW}) \,\} \\
\texttt{b} &\mapsto \{\, \mathbf{T}(\texttt{b:Company}), \mathbf{T}(\texttt{b:Insolvent}), \mathbf{F}(\texttt{b:}\exists\texttt{hasCustomer.}\neg\texttt{Insolvent}) \,\} \\
\texttt{c} &\mapsto \{\, \mathbf{T}(\texttt{c:Company}), \mathbf{F}(\texttt{c:Insolvent}), \mathbf{F}(\texttt{c:}\neg\texttt{Insolvent}) \,\} \\
\texttt{d} &\mapsto \{\, \mathbf{T}(\texttt{d:Company}), \mathbf{T}(\texttt{d:}\neg\texttt{Insolvent}) \,\}
\end{aligned}$$

– SF'_Δ is defined by the map:

$$\begin{aligned}
\texttt{a} &\mapsto \{\, \mathbf{T}(\texttt{a:}D \rightarrow \texttt{CW}) \,\} \\
\texttt{b} &\mapsto \{\, \mathbf{F}(\texttt{b:Insolvent} \sqcap \exists\texttt{hasCustomer.}\neg\texttt{Insolvent}) \,\} \\
\texttt{c} &\mapsto \{\, \mathbf{F}(\texttt{c:Insolvent} \sqcap \exists\texttt{hasCustomer.}\neg\texttt{Insolvent}) \,\} \\
\texttt{d} &\mapsto \emptyset
\end{aligned}$$

– $\text{DF}'_\Delta(\mathbf{F}(\texttt{a:}D)) = \{\texttt{b}, \texttt{c}\}$ and $\text{DF}'_\Delta(\mathbf{F}(\texttt{b:}\exists\texttt{hasCustomer.}\neg\texttt{Insolvent})) = \{\texttt{c}\}$.

We continue the construction of the countermodel by identifying the successors of the expanded graph: there are only two formulas in $\text{FORM}(\mathcal{G}'_\Delta)$ that lead to the application of successor rules, namely

$$W = \mathbf{T}(\texttt{d:}\neg\texttt{Insolvent}) \quad \text{and} \quad W = \mathbf{F}(\texttt{c:}\neg\texttt{Insolvent})$$

We call $\mathcal{G}_1$ and $\mathcal{G}_2$ the corresponding successor graphs.

After their expansion, the graphs $\text{Exp}(\mathcal{G}_1) = \mathcal{G}'_1$ and $\text{Exp}(\mathcal{G}_2) = \mathcal{G}'_2$ look as follows:

– $\mathcal{N}'_1 = \mathcal{N}'_2 = \mathcal{N}_\Delta$;

– $\mathcal{E}'_1 = \mathcal{E}'_2 = \mathcal{E}_\Delta$;

– $\text{DF}'_1 = \text{DF}'_2 = (\text{DF}'_\Delta)_s$;

– For $\mathcal{G}'_1$, its primary and secondary formulas are defined by:

$$\begin{aligned}
\text{PF}'_1 : \ \texttt{a} &\mapsto \{\, \mathbf{T}(\texttt{a:Company}), \mathbf{T}(\texttt{a:CW}) \,\} \\
\texttt{b} &\mapsto \{\, \mathbf{T}(\texttt{b:Company}), \mathbf{T}(\texttt{b:Insolvent}) \,\} \\
\texttt{c} &\mapsto \{\, \mathbf{T}(\texttt{c:Company}) \,\} \\
\texttt{d} &\mapsto \{\, \mathbf{T}(\texttt{d:Company}), \mathbf{T_s}(\texttt{d:}\bot), \mathbf{F}(\texttt{d:Insolvent}) \,\}
\end{aligned}$$

$$\begin{aligned}
\text{SF}'_1 : \ \texttt{a} &\mapsto \{\, \mathbf{T}(\texttt{a:}D \rightarrow \texttt{CW}) \,\} \\
\texttt{b} &\mapsto \emptyset \\
\texttt{c} &\mapsto \emptyset \\
\texttt{d} &\mapsto \{\, \mathbf{T}(\texttt{d:}\neg\texttt{Insolvent}) \,\}
\end{aligned}$$

– For $\mathcal{G}'_2$, its primary and secondary formulas are defined by:

$$
\begin{array}{lll}
\mathrm{PF}'_2: & \mathtt{a} \mapsto & \{\, \mathbf{T}(\mathtt{a{:}Company}), \mathbf{T}(\mathtt{a{:}CW}) \,\} \\
& \mathtt{b} \mapsto & \{\, \mathbf{T}(\mathtt{b{:}Company}), \mathbf{T}(\mathtt{b{:}Insolvent}) \,\} \\
& \mathtt{c} \mapsto & \{\, \mathbf{T}(\mathtt{c{:}Company}), \mathbf{T}(\mathtt{c{:}Insolvent}), \mathbf{F}(\mathtt{c{:}}\bot), \,\} \\
& \mathtt{d} \mapsto & \{\, \mathbf{T}(\mathtt{d{:}Company}), \mathbf{T_s}(\mathtt{d{:}}\bot), \mathbf{F}(\mathtt{d{:}Insolvent}) \,\} \\
\mathrm{SF}'_2: & \mathtt{a} \mapsto & \{\, \mathbf{T}(\mathtt{a{:}}D \rightarrow \mathtt{CW}) \,\} \\
& \mathtt{b} \mapsto & \emptyset \\
& \mathtt{c} \mapsto & \{\, \mathbf{F}(\mathtt{c{:}}\neg\mathtt{Insolvent}) \,\} \\
& \mathtt{d} \mapsto & \{\, \mathbf{T}(\mathtt{d{:}}\neg\mathtt{Insolvent}) \,\}
\end{array}
$$

Note that these graphs do not lead to the construction of further successors. It is easy to check that the model $\underline{K}(\Delta)$ constructed by the above graphs coincides with the model $\underline{K}$ inspected in Example 1 and that it is indeed a countermodel for the initial assertion. $\diamond$

3.4.8 Tableau procedure

We can write in form of an algorithm the intuitive procedure described at the end of the previous section: the decision procedure REAL for realizability is presented in Figure 3.6. The presentation of our algorithm is inspired to the ones provided in [Tobies(2001)] for classical DLs.

The procedure REAL takes as input an acyclic set of signed formulas Δ and answers whether this set is "realizable" or "not realizable". REAL first builds the starting graph $\mathcal{G}_\Delta$ and calls the sub-procedure CONS to decide the consistency (and thus the realizability) of $\mathcal{G}_\Delta$. The procedure CONS represents the actual steps for the construction of each graph $\mathcal{G}$ of the countermodel. The first loop in CONS represents the expansion step: expansion rules of Figure 3.3 are applied to $\mathcal{G}$ until a clash is found or no more rules are applicable. Whenever a clash is found, the procedure answers that the set of formulas associated with $\mathcal{G}$ is "inconsistent". After the expansion step, if no clashes are found, the procedure continues by applying to $\mathcal{G}$ the successor rules of Figure 3.4. Each iteration builds a successor graph $\mathcal{G}'$ and recursively calls CONS on this new graph: if any successor $\mathcal{G}'$ is found to contain a clash, then the procedure returns "inconsistent". If no one of the previous steps leads to a clash, then CONS answers that the formulas of the graph are "consistent".

We remark that this algorithm coincides with the countermodel construction procedure described in the previous sections. The procedure begins by building the starting graph corresponding to the set Δ, it applies exhaustively every possible expansion rule to its formulas and explores the successors of the graph, applying recursively the expansion. However, note that REAL does not return the constructed countermodel in the case that Δ is realizable: the purpose of the algorithm is just to verify the realizability of the given set of formulas.

REAL(Δ){

$\mathcal{N}_\Delta = \{\, c \mid \mathcal{S}(c : D) \in \Delta \,\}$;
$\mathcal{E}_\Delta = \{\, (c, d, R) \mid \mathbf{T}((c, d) : R) \in \Delta \,\}$;
$\text{PF}_\Delta(c) = \{\, \mathcal{S}(c : D) \mid \mathcal{S}(c : D) \in \Delta \text{ and } c \in \mathcal{N}_\Delta \,\}$;
$\text{TB} = \{\, \mathbf{T}(A \sqsubseteq C) \mid \mathbf{T}(A \sqsubseteq C) \in \Delta \,\}$;
$\text{SF}_\Delta = \text{DF}_\Delta = \emptyset$;
$\mathcal{G}_\Delta = \langle \mathcal{N}_\Delta, \mathcal{E}_\Delta, \text{PF}_\Delta, \text{SF}_\Delta, \text{TB}, \text{DF}_\Delta \rangle$;
`if`(CONS($\mathcal{G}_\Delta$) == "inconsistent") `then return` "not realizable";
`else return` "realizable";
}

CONS($\mathcal{G}$){

`while` (exp-rules can be applied) `and` (PF clash-free) `do begin`
pick $W \in$ FORM($\mathcal{G}$);
$\mathcal{G}$ = exp-rules($W, \mathcal{G}$);
`end`
`if` (PF contains a clash) `then return` "inconsistent";
`while` (succ-rules can be applied) `do begin`
pick $W \in$ FORM($\mathcal{G}$);
$\mathcal{G}'$ = succ-rules($W, \mathcal{G}$);
`if` (CONS($\mathcal{G}'$) == "inconsistent") `then return` "inconsistent";
`end`
`return` "consistent";
}

Figure 3.6: Tableau procedure for $\mathcal{T}_\mathcal{K}$

3.5 Discussion on complexity

After proving the completeness of our procedure, we can now briefly discuss the complexity issues of reasoning in $\mathcal{KALC}$. First of all, we highlight the relation between $\mathcal{KALC}$ and **Int** stated by the following result.

Proposition 3.16
*Given a formula $c : D \in \mathcal{L}$ such that D does not contain occurrences of quantifiers, $\models^{\mathrm{k}} c : D$ iff $D \in$ **Int**.* □

This proposition can be shown by providing a straightforward translation of formulas and models between $\mathcal{KALC}$ and **Int**. From this fact and well-known results on the decision of **Int** [Statman(1979)], we can immediately draw the following hardness result:

Theorem 3.17
$\mathcal{KALC}$ realizability is PSPACE*-hard.* □

Now, if we could formulate an algorithm for $\mathcal{KALC}$ realizability that is complete with respect to the proposed proof search strategy and which space cost is polynomial in the size of the initial set of formulas, we would obtain that realizability in $\mathcal{KALC}$ is PSPACE-complete.

However, consistently to what happens in the case of classical reasoning for $\mathcal{ALC}$ [Tobies(2001)], if we follow the proposed procedure and exhaustively expand every graph in the construction of a countermodel, then the number of nodes in each expanded graph (and the formulas associated to them) can be exponential in the size of the initial set of formulas. A proof of this fact can be seen in the bound given in Property (P3) of Section 3.4.7.

On the other hand, we conjecture that we can adopt a proof strategy similar to the *trace technique* [Schmidt-Schauß and Smolka(1991), Tobies(2001)] for $\mathcal{ALC}$ satisfiability: we expand graphs of the countermodel one path at a time, searching for clashes only among formulas for each path. By Property (P2) we know that the length of each path is polynomial in the maximum depth and degree w.r.t. the TBox of the starting set of formulas: thus if we could expand graphs only considering one path at a time, we would obtain a PSPACE procedure. Our intuition is supported by the following proof, which shows a form of independence between paths.

Proposition 3.18 (Path independence)
Given a graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \mathrm{PF}, \mathrm{SF}, \mathrm{TB}, \mathrm{DF} \rangle$, an expansion or successor rule r on $W = \mathcal{S}(c : D) \in \mathrm{PF}(c)$ and $d \in \mathcal{N}$ with $d \neq c$ or $(c, d, R) \notin \mathcal{E}$, then:

- *if r is an expansion rule, its application never directly leads to a clash on $\mathrm{PF}(d)$;*
- *if r is a successor rule, its application directly leads to a clash on $\mathrm{PF}(d)$ only if $\mathbf{T_s}(d : \bot) \in \mathrm{PF}(d)$;*

Proof: The assertion is proved by cases on the form of the rule r and the formula W.

If r is an expansion rule and $W = \mathcal{S}(c : C \sqcap D), \mathcal{S}(c : C \sqcup D), \mathcal{S}(c : C \to D)$, the application of r adds at most formulas of the kind $\mathcal{S}_W(c : C)$ to $\mathrm{PF}(c)$: this implies that r can never generate a clash on an individual $d \neq c$.

If r is an expansion rule and $W = \mathbf{T}(c : \exists R.C), \mathbf{F}(c : \forall R.C)$, the application of r adds at most formulas of the kind $\mathcal{S}_W(q : C)$ to $\mathrm{PF}(q)$ with $q \notin \mathcal{N}$ a new individual. Thus, r can never generate a clash on an individual $d \in \mathcal{N}$.

If r is an expansion rule and $W = \mathbf{F}(c : \exists R.C), \mathbf{T}(c : \forall R.C)$, the application of r adds at most formulas of the kind $\mathcal{S}_W(q : C)$ to $\mathrm{PF}(q)$ with $(c, q, R) \in \mathcal{E}$. Thus, the application of such rules for dup-formulas can not lead to a direct clash on an individual d with $(c, d, R) \notin \mathcal{E}$.

If r is a successor rule, all of the formulas of $\mathrm{PF}(d)$ with $d \in \mathcal{N}$ are modified. In both the cases of $W = \mathcal{S}(c : C \to D)$ and $W = \mathbf{F}(c : \forall R.C)$, the considerations of their counterpart in expansion rules holds for $\mathrm{PF}(c)$ and $\mathrm{PF}(q)$, with $q \notin \mathcal{N}$. However, for any $d \in \mathcal{N}$ with $d \neq c$, the rules define $\mathrm{PF}'(d) = (\mathrm{PF}(d))_s$, thus if a formula $\mathbf{T_s}(d : H)$ belongs to $\mathrm{PF}(d)$, then $\mathbf{T}(d : H) \in \mathrm{PF}'(d)$. Given that any formula of the kind $\mathbf{F}(d : H) \in \mathrm{PF}(d)$ is discharged in the application or r, the only way to obtain a clash on d is the case in which $\mathbf{T_s}(d : \bot) \in \mathrm{PF}(d)$, which would imply $\mathbf{T}(d : \bot) \in \mathrm{PF}'(d)$. □

Thus, a PSPACE algorithm for $\mathcal{KALC}$ realizability should proceed as follows: given a graph, expand each path searching for a clash; if the graph does not contain a formula of the kind $\mathbf{T_s}(d : \bot)$, meaning that the graph does not represent a final world, then apply each possible successor rule and proceed recursively on the successor graphs.

Despite these intuitive considerations, formulating and proving that this strategy is in fact complete is complicated by the necessity of a combinatoric study on the interactions between the rules of the calculus, showing that such strategy covers all of the possible correct ways to expand graphs. In particular it is necessary to understand the interaction between expansion rules and successor rules: while the former ones can be described in proof theory as *invertible rules*, that is the order of their application does not affect the success of the proof, the latter ones are *non-invertible rules*, as their application can remove formulas relevant for the discovery of clashes in a proof branch. We also note that such formulation of the calculus using non-invertible rules is indeed related to the fact that our logic is based on an intuitionistic semantics: in the classical tableau calculi for $\mathcal{ALC}$ every rule is invertible, thus we can apply them with any suitable strategy without losing any relevant information.

4

Relations with other logics

In this section we study the *logic* of $\mathcal{KALC}$ and its relations with some other logics. In particular, we discuss the relations of $\mathcal{KALC}$ with its variant $\mathcal{KALC}^{\infty}$, which admits infinite posets, and the logic $\mathcal{IALC}^{\infty}$, a direct translation in the description logic context of the first order intuitionistic semantics. We then discuss the relations with **QInt** and the intuitionistic multi-modal logic $\mathbf{FS}_n$.

4.1 Concept formulas and the logic $\mathrm{Log}_{\mathcal{KALC}}$

In this section we are interested in studying the *logic* of $\mathcal{KALC}$ that is the set of purely logical formulas valid in $\mathcal{KALC}$ semantics. To do this we need to consider formulas independently from their non-logical features. To this aim we introduce the language $\mathcal{L}^{c}$ that differ from $\mathcal{L}$ only in the definition of formulas. Formally formulas H of $\mathcal{L}^{c}$ are defined according to the following grammar:

$$H \quad ::= \quad (c, d) : R \mid c : C \mid C$$

where $c, d \in \mathtt{NI}$, $R \in \mathtt{NR}$ and C is a concept of $\mathcal{L}^{c}$. We call *concept formulas of* $\mathcal{L}^{c}$ the formulas of the kind C with C a concept. $\mathcal{KALC}$-models for $\mathcal{L}^{c}$ are defined exactly as the models for $\mathcal{L}$.

Given a concept formula H of $\mathcal{L}^{c}$, a $\mathcal{KALC}$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ for $\mathcal{L}^{c}$ and $\alpha \in P$, H is *valid in (the world)* α of $\underline{K}$, and we write $\alpha \Vdash H$, iff the following holds:

– $\alpha \Vdash H$ iff, for every $\beta \in P$ with $\alpha \leq \beta$ and for every $d \in \mathcal{D}^{\beta}$, $\beta \Vdash d : H$.

The concept formula H is *valid* in $\underline{K}$, and we write $\underline{K} \Vdash H$, iff $\alpha \Vdash H$ for every $\alpha \in P$ (or, equivalently, if $\rho \Vdash H$).

We remark that, according to the previous definitions, we have that, given C, D two concepts, $\alpha \Vdash C \to D$ iff $\alpha \Vdash C \sqsubseteq D$. Hence in $\mathcal{L}^{c}$ implicative concept formulas represent subsumptions.

Now, let us denote with $\mathrm{Mod}_{\mathcal{KALC}}$ the class of all $\mathcal{KALC}$-models. We define the *logic of* $\mathcal{KALC}$ as the set of concept formulas valid in every $\mathcal{KALC}$ model, that is:

$$\mathrm{Log}_{\mathcal{KALC}} = \{H \mid H \text{ concept formula of } \mathcal{L}^{\mathrm{c}} \text{ and, for every } \underline{K} \in \mathrm{Mod}_{\mathcal{KALC}}, \underline{K} \Vdash H\}$$

4.2 Intuitionistic description logic $\mathcal{IALC}^{\infty}$

Now we present a Kripke semantics for $\mathcal{ALC}$ that is directly obtained by adapting the semantics provided for the first order formulation of intuitionistic logic **QInt** to the description logic context. We call $\mathcal{IALC}^{\infty}$ the resulting logic: the introduction of such logic is useful to the subsequent comparison with the logic for $\mathcal{KALC}$.

The definition of language, concepts and formulas of $\mathcal{IALC}^{\infty}$ coincide with the ones given for $\mathcal{KALC}$. The semantics of $\mathcal{IALC}^{\infty}$ is given by directly adjusting the Kripke structures for intuitionistic first order logic to the description logic case. A $\mathcal{IALC}^{\infty}$*-model* for $\mathcal{L}_{\mathcal{N}}$ is a quadruple

$$\underline{K} = \langle\, P, \leq, \rho, \iota \,\rangle$$

where:

– $(P, \leq)$ is a poset with root ρ;

– ι is a function associating to every $\alpha \in P$ a model $\iota(\alpha) = (\mathcal{D}^{\alpha}, \cdot^{\alpha})$ for $\mathcal{L}_{\mathcal{N}}$, such that, for every $\alpha, \beta \in P$ with $\alpha \leq \beta$:

  (K1) $\mathcal{D}^{\alpha} \subseteq \mathcal{D}^{\beta}$;
  (K2) for every $c \in \mathcal{N}$, $c^{\alpha} = c^{\beta}$;
  (K3) for every $A \in \mathtt{NC}$, $A^{\alpha} \subseteq A^{\beta}$;
  (K4) for every $R \in \mathtt{NR}$, $R^{\alpha} \subseteq R^{\beta}$.

In other words, $\mathcal{KALC}$-models differ from $\mathcal{IALC}^{\infty}$-models simply by the *finiteness* of the poset on which they are defined: as in the case of **QInt**, $\mathcal{IALC}^{\infty}$-models can indeed have an infinite number of worlds.

The forcing relation over $\mathcal{IALC}^{\infty}$-models is defined as in the case of $\mathcal{KALC}$-models. As a matter of fact, every $\mathcal{KALC}$-model is also a $\mathcal{IALC}^{\infty}$-model.

By using conditions (K1)–(K4) and the definition of the forcing relation, we can prove that also $\mathcal{IALC}^{\infty}$ meets the monotonicity property and the constructive properties distinctive of the intuitionistic semantics.

Let us denote the class of all $\mathcal{IALC}^{\infty}$-models with $\mathrm{Mod}_{\mathcal{IALC}^{\infty}}$. The logic of $\mathcal{IALC}^{\infty}$ is the set of concept formulas valid in every $\mathcal{IALC}^{\infty}$-model:

$$\mathrm{Log}_{\mathcal{IALC}^{\infty}} = \{H \mid H \text{ concept formula of } \mathcal{L}^{\mathrm{c}} \text{ and, for every } \underline{K} \in \mathrm{Mod}_{\mathcal{IALC}^{\infty}}, \underline{K} \Vdash H\}$$

Note that this direct formulation of an intuitionistic semantics for description logics is indeed analogous to the IALC logic provided in [de Paiva(2005)], aside from its slightly different notation. However, as we discuss in the following, both of these direct translations do not seem to be appropriate for the definition of a constructive version of $\mathcal{ALC}$.

4.3 The logic $\mathcal{KALC}^{\infty}$

In this section we present a Kripke semantics which is an extension of the one for $\mathcal{KALC}$ admitting infinite sets of worlds [Bozzato *et al.*(2009b), Villa(2010)] and which represents, in fact, a restriction to the semantics of $\mathcal{IALC}^{\infty}$.

Given a poset $(P, \leq)$ and $\alpha \in P$, a *final element* of P is a element $\phi \in P$ such that, for every $\alpha \in P$, $\phi \leq \alpha$ implies $\phi = \alpha$. Given $\alpha \in P$ we denote with $\mathrm{Fin}(\alpha)$ the set of final elements $\phi \in P$ such that $\alpha \leq \phi$. We call *K-poset* any poset $(P, \leq)$ such that, for every $\alpha \in P$, $\mathrm{Fin}(\alpha) \neq \emptyset$. We remark that every finite poset is a K-poset.

A $\mathcal{KALC}^{\infty}$*-model* for $\mathcal{L}_{\mathcal{N}}$ is a quadruple

$$\underline{K} = \langle\, P, \leq, \rho, \iota \,\rangle$$

where:

– $(P, \leq)$ is a K-poset with root ρ;

– ι is a function associating to every $\alpha \in P$ a model $\iota(\alpha) = (\mathcal{D}^{\alpha}, \cdot^{\alpha})$ for $\mathcal{L}$, such that, for every $\alpha, \beta \in P$ with $\alpha \leq \beta$:

  (K1) $\mathcal{D}^{\alpha} \subseteq \mathcal{D}^{\beta}$;

  (K2) for every $c \in \mathtt{NI}$, $c^{\alpha} = c^{\beta}$;

  (K3) for every $C \in \mathtt{NC}$, $C^{\alpha} \subseteq C^{\beta}$;

  (K4) for every $R \in \mathtt{NR}$, $R^{\alpha} \subseteq R^{\beta}$.

The forcing relation is again defined as in the case of $\mathcal{KALC}$-models. We remark that every $\mathcal{KALC}$-model is in particular a $\mathcal{KALC}^{\infty}$-model and every $\mathcal{KALC}^{\infty}$-model is a $\mathcal{IALC}^{\infty}$-model.

Now let us denote with $\mathrm{Mod}_{\mathcal{KALC}^{\infty}}$ the class of all $\mathcal{KALC}^{\infty}$-models. We define the *logic of* $\mathcal{KALC}^{\infty}$ as the set of concept formulas valid in every $\mathcal{KALC}^{\infty}$ model, that is:

$$\mathrm{Log}_{\mathcal{KALC}^{\infty}} = \{H \mid H \text{ concept formula of } \mathcal{L}^{\mathrm{c}} \text{ and, for every } \underline{K} \in \mathrm{Mod}_{\mathcal{KALC}^{\infty}}, \underline{K} \Vdash H\}$$

Since every $\mathcal{KALC}$ model is a $\mathcal{KALC}^{\infty}$ model we immediately have that:

$$\mathrm{Log}_{\mathcal{IALC}^{\infty}} \subseteq \mathrm{Log}_{\mathcal{KALC}^{\infty}} \subseteq \mathrm{Log}_{\mathcal{KALC}}$$

Now, let us consider the axiom-schema

$$\mathrm{Kur} \quad = \quad \forall R.\neg\neg A \rightarrow \neg\neg \forall R.A$$

By *axiom schema* we mean a concept formula where role-names and concept names are meta-variables ranging over role names of NR and concepts names of NC, respectively. An instance of the axiom schema in $\mathcal{L}$ is obtained by replacing every role meta-variable R with a role name of $\mathcal{L}$ and every concept meta-variable A with a concept name of $\mathcal{L}$.

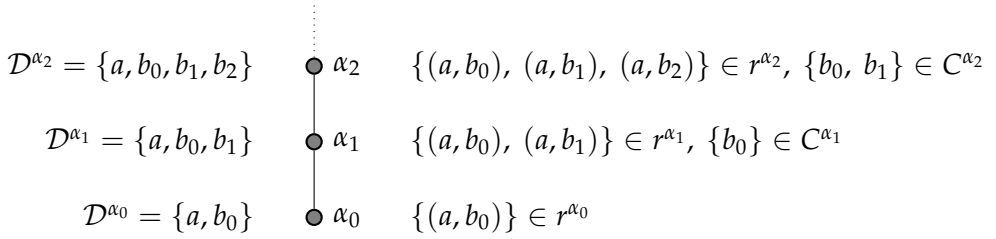

Figure 4.1: A counter-model for Kur.

Theorem 4.1
Let K be any instance of the axiom schema Kur *in $\mathcal{L}$ and let $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ be a $\mathcal{KALC}^{\infty}$-model for $\mathcal{L}$, then $\rho \Vdash K$.*

Proof: Let $K = \forall R.\neg\neg A \rightarrow \neg\neg\forall R.A$ and let us suppose that $\rho \nVdash K$. This implies that there exist $\alpha \in P$ and $d \in \mathcal{D}^{\alpha}$ such that $\alpha \Vdash d : \forall R.\neg\neg A$ and $\alpha \nVdash d : \neg\neg\forall R.A$. $\alpha \Vdash d : \forall R.\neg\neg A$ means that for every $\beta \geq \alpha$ and for every $d' \in \mathcal{D}^{\beta}$, $\beta \Vdash (d, d') : R$ implies that $\beta \Vdash d' : \neg\neg A$ in $\underline{K}$. Since $\underline{K}$ is a $\mathcal{KALC}^{\infty}$-model, for every final element $\gamma \in P$ such that $\alpha \leq \gamma$, $\gamma \Vdash (d, d') : R$ implies that $\gamma \Vdash d' : A$ in $\underline{K}$.

$\alpha \nVdash d : \neg\neg\forall R.A$ means that there exists $\beta \geq \alpha$ such that $\beta \Vdash d : \neg\forall R.A$ in $\underline{K}$. Since $\underline{K}$ is a $\mathcal{KALC}^{\infty}$-model, for every final element $\gamma \in P$ such that $\beta \leq \gamma$, there exists $d' \in \mathcal{D}^{\gamma}$ such that $\gamma \Vdash (d, d') : R$ and $\gamma \Vdash d' : \neg A$ in $\underline{K}$, a contradiction. □

This axiom is not valid in $\mathcal{IALC}^{\infty}$ semantics and the Kripke semantics presented in [de Paiva(2005)]. Indeed, let us consider the Kripke structure

$$\underline{K} = \langle\ \{\alpha_i\}_{i \in \omega},\ \leq,\ \alpha_0,\ \iota\ \rangle$$

represented in Figure 4.1 where:

– $\alpha_i \leq \alpha_j$ iff $i \leq j$;

– For every α_i:

  – $\mathcal{D}^{\alpha_i} = \{a, b_0, \ldots, b_i\}$;
  – $r^{\alpha_i} = \{(a, b_0), \ldots, (a, b_i)\}$;
  – $C^{\alpha_0} = \emptyset$ and $C^{\alpha_i} = \{b_0, \ldots, b_{i-1}\}$ for $i > 0$.

We note that $\underline{K}$ is not a $\mathcal{KALC}^{\infty}$-model, as $\mathrm{Fin}(\alpha_0) = \emptyset$. On the other hand $\underline{K}$ satisfies Conditions (K1)–(K4) and belongs to the class of $\mathcal{IALC}^{\infty}$-models. It is easy to check that $\underline{K}$ is a countermodel for Kur, since

$$\alpha_0 \nVdash a : \forall r.\neg\neg C \rightarrow \neg\neg\forall r.C$$

However, we have that this principle hold for all finite $\mathcal{IALC}^{\infty}$-models, that correspond to finite $\mathcal{KALC}^{\infty}$-models: this shows that $\mathcal{IALC}^{\infty}$ fails to have the finite

model property. Thus, according with the previous considerations, we think that $\mathcal{KALC}^{\infty}$ semantics is more appropriate than the one of [de Paiva(2005)] and than $\mathcal{IALC}^{\infty}$.

We remark that, while we proved:

$$\mathrm{Log}_{\mathcal{KALC}^{\infty}} \subseteq \mathrm{Log}_{\mathcal{KALC}}$$

it is still an open issue to determine whether, as we conjecture:

$$\mathrm{Log}_{\mathcal{KALC}} \subseteq \mathrm{Log}_{\mathcal{KALC}^{\infty}}$$

If this is true, we can conclude that $\mathcal{KALC} = \mathcal{KALC}^{\infty}$ and that this logic meets the finite model property, meaning that also $\mathcal{KALC}^{\infty}$ is decidable. In particular, we point out that a possible way to prove this result is by means of *filtration methods* [Chagrov and Zakharyaschev(1997), Gabbay *et al.*(2003)] on $\mathcal{KALC}^{\infty}$ models: the idea is to obtain, for every possibly infinite $\mathcal{KALC}^{\infty}$-model, its equivalent "finite counterpart" in $\mathcal{KALC}$ models.

To conclude this part, we remark that the Kur axiom schema corresponds to the first order principle $\forall x.\neg\neg H(x) \rightarrow \neg\neg\forall x.H(x)$. This principle is known in the literature of constructive logics as *Kuroda principle* [Gabbay(1981), Troelstra(1973b)]. Adding this schema to intuitionistic first order logic **QInt**, we get a proper extension of **QInt**, that satisfies the *Disjunction Property* and the *Explicit Definability Property*.

4.4 Relations with QInt

In this section we describe some relations between $\mathcal{KALC}^{\infty}$ and the first order intuitionistic logic **QInt** presented in Chapter 2. We begin by providing a syntactic translation between concept formulas of $\mathcal{L}^{c}$ and first order formulas of $\mathcal{FL}$. Let $\Sigma = \langle \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ be a signature in which $\mathcal{C} = \mathcal{F} = \emptyset$ and:

– for every $C_i \in \mathtt{NC}$, there exists an unary predicate $p_i \in \mathcal{R}$;

– for every $R_i \in \mathtt{NR}$, there exists a binary predicate $r_i \in \mathcal{R}$.

For every variable $x \in \mathcal{V}$, we define the translation[1] π^x from concept formulas of $\mathcal{L}^{c}$ to first order formulas of $\mathcal{FL}$:

$$\begin{aligned} \pi^x(A_i) &= p_i(x) \\ \pi^x(\bot) &= \bot \\ \pi^x(\neg C) &= \neg\pi^x(C) \\ \pi^x(C \sqcap D) &= \pi^x(C) \wedge \pi^x(D) \end{aligned}$$

[1] This translation is similar to the usual translation presented in [Sattler *et al.*(2003), de Paiva(2005)] from $\mathcal{ALC}$ to the two-variable first order language $\mathcal{L}^{2}$.

$$\begin{aligned}
\pi^x(C \sqcup D) &= \pi^x(C) \vee \pi^x(D) \\
\pi^x(C \to D) &= \pi^x(C) \to \pi^x(D) \\
\pi^x(\exists R_i.C) &= \exists y.(r_i(x,y) \wedge \pi^y(C)) \\
\pi^x(\forall R_i.C) &= \forall y.(r_i(x,y) \to \pi^y(C))
\end{aligned}$$

for $A_i \in \mathtt{NC}$, $R_i \in \mathtt{NR}$ and $p_i, r_i \in \mathcal{R}$. Given a set Γ of formulas of $\mathcal{L}^c$, we define $\pi^x(\Gamma)$ as the set $\{\pi^x(H) \mid H \in \Gamma\}$.

Given a $\mathcal{KALC}^\infty$ model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$, we define the translation $\pi(\underline{K})$ to first order intuitionistic models as follows:

$$\pi(\underline{K}) = \langle\, P,\, \leq,\, \rho,\, \mathcal{D},\, \Vdash \,\rangle$$

where, for every $\alpha \in P$, if $\iota(\alpha) = (\mathcal{D}^\alpha, \cdot^\alpha)$, then:

– $\mathcal{D}(\alpha) = \mathcal{D}^\alpha$;

– if $d \in \mathcal{D}^\alpha$ and $d \in A_i^\alpha$, then $\alpha \Vdash p_i(d)$ in $\pi(\underline{K})$;

– if $d, e \in \mathcal{D}^\alpha$ and $(d,e) \in R_i^\alpha$, then $\alpha \Vdash r_i(d,e)$ in $\pi(\underline{K})$;

The definition of the relation $\Vdash$ over complex formulas follows the usual one given for the forcing in **QInt**-models. Note that, since $\mathrm{Mod}_{\mathcal{KALC}} \subseteq \mathrm{Mod}_{\mathcal{KALC}^\infty}$, this translation also applies to $\mathcal{KALC}$-models.

Let us consider the usual first order formulation of the Kuroda principle

$$\forall x.\neg\neg H(x) \to \neg\neg \forall x.H(x)$$

and let **Kur** be the logic obtained adding such principle to **QInt**. We can easily prove that, for every $\mathcal{KALC}^\infty$ model $\underline{K}$, its translation $\pi(\underline{K})$ is a model of **Kur**. First of all, $\pi(\underline{K})$ is a **QInt**-model by definition: in fact, for every $\alpha, \beta \in P$ with $\alpha \leq \beta$, by (K1) we have that $\mathcal{D}(\alpha) \subseteq \mathcal{D}(\beta)$; by (K3) it holds that $\alpha \Vdash p_i(d)$ implies $\beta \Vdash p_i(d)$ and the same applies to the r_i binary predicates by (K4). Now, since the poset at the base of the $\mathcal{KALC}^\infty$-model $\underline{K}$ is not modified in the translation, also $\pi(\underline{K})$ is defined over a K-poset. Thus, we can prove a result analogous to Theorem 4.1 over first order translations, that is:

Theorem 4.2
Let K be any instance of the first order Kuroda axiom schema and let $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ be a $\mathcal{KALC}^\infty$-model for $\mathcal{L}$, then $\rho \Vdash K$ in $\pi(\underline{K})$. □

The proof of this result is similar to the one provided for Theorem 4.1, and it is again based on the definition of $\pi(\underline{K})$ over a K-poset. This implies that every model $\pi(\underline{K})$ is indeed a model of **Kur**.

We can prove the soundness of our translations with the following theorem:

Theorem 4.3
Let $H \in \mathcal{L}^c$ be a concept formula and let $\underline{K}$ be a $\mathcal{KALC}^\infty$-model. $\underline{K} \Vdash H$ iff $\pi(\underline{K}) \Vdash \pi^x(H)$.

Proof: The assertion can be proved by induction on the structure of the formula H and the inductive definition of the translation: we only give some of the most relevant cases.

If $H = A_i$, with $A_i \in \mathtt{NC}$, then $\underline{K} \Vdash A_i$ iff for every $\beta \geq \rho$ and $d \in \mathcal{D}^\beta$, $\beta \Vdash d : A_i$. This holds iff for every $d \in \mathcal{D}^\beta$, $d \in A_i^\beta$: by definition of the translation on models, this is true iff for every $d \in \mathcal{D}^\beta$, $\beta \Vdash\!\Vdash p_i(d)$. The latter means that $\rho \Vdash\!\Vdash p_i(x)$, which in turn holds iff $\pi(\underline{K}) \Vdash\!\Vdash p_i(x)$ and iff $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(A_i)$.

If $H = C \sqcup D$, then $\underline{K} \Vdash C \sqcup D$ iff $\underline{K} \Vdash C$ or $\underline{K} \Vdash D$. By induction hypothesis, this holds iff $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(C)$ or $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(D)$. This means $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(C) \vee \pi^x(D)$, which in turn is true iff $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(C \sqcup D)$.

If $H = C \rightarrow D$, then $\underline{K} \Vdash C \rightarrow D$ iff, for every $\beta \geq \rho$ and $d \in \mathcal{D}^\beta$, $\beta \Vdash d : C$ implies $\beta \Vdash d : D$. By induction hypothesis, this holds iff $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(C)$ implies $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(D)$. This can be rewritten as $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(C) \rightarrow \pi^x(D)$, which in turn is true iff $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(C \rightarrow D)$.

If $H = \exists R.C$, then $\underline{K} \Vdash \exists R.C$ iff for every $\beta \geq \rho$ and $d \in \mathcal{D}^\beta$, there exists $e \in \mathcal{D}^\beta$ such that $\beta \Vdash (d, e) : R$ and $\beta \Vdash e : C$. By the definition of the translation on models, this holds iff for every $d \in \mathcal{D}^\beta$ there exists a closing substitution $\sigma = [e/y]$ such that $\beta \Vdash\!\Vdash r_i(d, y)\sigma$ and $\beta \Vdash\!\Vdash \pi^y(C)\sigma$. This can also be rewritten as $\beta \Vdash\!\Vdash \exists y.(r_i(x, y) \wedge \pi^y(C))$, which in turn means that $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(\exists R.C)$.

If $H = \forall R.C$, then $\underline{K} \Vdash \forall R.C$ iff for every $\beta \geq \rho$ and $d, e \in \mathcal{D}^\beta$, if $\beta \Vdash (d, e) : R$ then $\beta \Vdash e : C$. By the definition of the translation on models, this holds iff for every $d \in \mathcal{D}^\beta$ and every closing substitution $\sigma = [e/y]$, $\beta \Vdash\!\Vdash r_i(d, y)\sigma$ implies $\beta \Vdash\!\Vdash \pi^y(C)\sigma$. This can also be rewritten as $\beta \Vdash\!\Vdash \forall y.(r_i(x, y) \rightarrow \pi^y(C))$, which in turn is true iff $\pi(\underline{K}) \Vdash\!\Vdash \pi^x(\forall R.C)$. □

Let H be a concept formula of $\mathcal{L}^c$, then if $H \notin \mathrm{Log}_{\mathcal{KALC}^\infty}$ then there exists a model $\underline{K} \in \mathrm{Mod}_{\mathcal{KALC}^\infty}$ such that $\underline{K} \nVdash H$. This implies that $\pi(\underline{K}) \nVdash\!\Vdash \pi^x(H)$ and $\pi^x(H) \notin$ **Kur**. We have that $\pi^x(.)$ is a mapping on $\mathcal{FL}$: this implies **Kur** $\subseteq \mathrm{Log}_{\mathcal{KALC}^\infty}$.

4.5 Relations with $\mathbf{FS}_n$

In this section we describe some relations between $\mathcal{KALC}$, $\mathcal{KALC}^\infty$ and the intuitionistic multi-modal logic $\mathbf{FS}_n$, which is the multi-modal version of the logic **FS** [Fischer Servi(1981), Fischer Servi(1984), Gabbay *et al.*(2003)].

4.5.1 $\mathbf{FS}_n$: language and semantics

Here we consider the *propositional n-modal language* $\mathcal{ML}_n$ consisting of:

– a fixed countably infinite set $\mathcal{PV}$ of propositional variables;

– the logical constants $\bot$ (false) and $\top$ (true);

– the logical connectives $\wedge, \vee, \rightarrow$ and $\neg$;

– the modal operators $\Box_1, \ldots, \Box_n$ and $\Diamond_1, \ldots, \Diamond_n$;

– the punctuation symbols: '(' and ')'.

$\mathcal{ML}_n$-formulas are built as usual:

– all propositional variables and the constants $\bot$ and $\top$ are $\mathcal{ML}_n$ formulas;

– if A and B are $\mathcal{ML}_n$ formulas then so are $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $\neg A$, $\Box_i A$ and $\Diamond_i A$ with $i \in \{1, \ldots, n\}$.

We write $A \in \mathcal{ML}_n$ to mean that A is a $\mathcal{ML}_n$-formula. To simplify notation, we use the following standard conventions on formula representation: we assume that $\neg$, $\Box_i$ and $\Diamond_i$ ($i \in \{1, \ldots, n\}$) bind stronger than $\wedge$ and $\vee$ which in turn are stronger than $\rightarrow$, and we omit brackets that can be uniquely recovered according to this priority of connectives.

A $\mathbf{FS}_n$*-standard model* for $\mathcal{ML}_n$ is a structure

$$\underline{M} = \langle\, P, \leq, \kappa, V \,\rangle$$

where:

– $(P, \leq)$ is a poset;

– κ is a function associating with every $\alpha \in P$ a structure

$$\kappa(\alpha) = \langle\, \Delta^\alpha, \mathcal{R}_1^\alpha, \ldots, \mathcal{R}_n^\alpha \,\rangle$$

where:

– Δ^α is a non-empty set;

– for every $i \in \{1, \ldots, n\}$, $\mathcal{R}_i^\alpha$ is a binary relation over Δ^α such that, for every $\alpha, \beta \in P$ such that $\alpha \leq \beta$:

(FS1) $\Delta^\alpha \subseteq \Delta^\beta$;

(FS2) for every $i \in \{1, \ldots, n\}$, $\mathcal{R}_i^\alpha \subseteq \mathcal{R}_i^\beta$.

– V is a *valuation* associating with every $\alpha \in P$ and every propositional variable p, a subset of Δ^α, that is $V(\alpha, p) \subseteq \Delta^\alpha$.

Given a $\mathcal{ML}_n$-formula A, $\alpha \in P$ and $w \in \Delta^\alpha$ the forcing relation $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} A$ (or simply $(\alpha, w) \overset{\bullet}{\Vdash} A$) is defined inductively as follows:

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} \top$ and $\underline{M}, (\alpha, w) \overset{\bullet}{\nVdash} \bot$;

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} p$ with $p \in \mathcal{PV}$ iff $w \in V(\alpha, p)$;

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} A \wedge B$ iff $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} A$ and $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} B$;

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} A \vee B$ iff either $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} A$ or $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} B$;

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} \neg A$ iff for all $\beta \in P$ such that $\alpha \leq \beta$, $\underline{M}, (\beta, w) \overset{\bullet}{\nVdash} A$;

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} A \rightarrow B$ iff for all $\beta \in P$ such that $\alpha \leq \beta$, either $\underline{M}, (\beta, w) \overset{\bullet}{\nVdash} A$ or $\underline{M}, (\beta, w) \overset{\bullet}{\Vdash} B$;

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} \Box_i A$ iff, for every $\beta \in P$ such that $\alpha \leq \beta$ and every $v \in \Delta^\beta$ such that $w \mathcal{R}_i^\beta v$, $\underline{M}, (\beta, v) \overset{\bullet}{\Vdash} A$;

– $\underline{M}, (\alpha, w) \overset{\bullet}{\Vdash} \Diamond_i A$ iff there exists $v \in \Delta^\alpha$ such that $w \mathcal{R}_i v$ and $\underline{M}, (\alpha, v) \overset{\bullet}{\Vdash} A$.

It is easy to prove, by induction on the structure of $\mathcal{ML}_n$ formulas, that the forcing relation is monotone with respect to $\leq$.

Now, let $\mathrm{Mod}_{\mathbf{FS}_n}$ be the class of all $\mathbf{FS}_n$-standard models, the *logic of* $\mathbf{FS}_n$ is the set of $\mathcal{ML}_n$ formulas valid in every $\mathbf{FS}_n$-standard model, that is:

$$\mathrm{Log}_{\mathbf{FS}_n} = \{H \mid H \in \mathcal{ML}_n \text{ and, for every } \underline{K} \in \mathrm{Mod}_{\mathbf{FS}_n}, \underline{K} \overset{\bullet}{\Vdash} H\}$$

4.5.2 Translations

In this section we consider the language $\mathcal{L}^{\mathrm{c}}$ where $\mathtt{NR}$ contains exactly n distinct role names, we denote with $R_1, \ldots, R_n$. We define the translation $(.)^\dagger$ from formulas of $\mathcal{L}^{\mathrm{c}}$ to formulas of $\mathcal{ML}_n$ as follows:

$$\begin{array}{rcl}
(C_i)^\dagger & = & p_i \text{ for } C_i \in \mathtt{NC} \\
(C \sqcap D)^\dagger & = & (C)^\dagger \wedge (D)^\dagger \\
(C \sqcup D)^\dagger & = & (C)^\dagger \vee (D)^\dagger \\
(C \rightarrow D)^\dagger & = & (C)^\dagger \rightarrow (D)^\dagger \\
(\forall R.C)^\dagger & = & \Box_i (C)^\dagger \\
(\exists R.C)^\dagger & = & \Diamond_i (C)^\dagger
\end{array}$$

We remark that the above translation is a bijection from $\mathcal{L}^{\mathrm{c}}$ to $\mathcal{ML}_n$. Given a set Γ of concept formulas of $\mathcal{L}^{\mathrm{c}}$ we denote with $(\Gamma)^\dagger$ the set $\{(H)^\dagger \mid H \in \Gamma\}$.

Given a $\mathcal{KALC}^\infty$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ we define the translation $(\underline{K})^\dagger$ of $\underline{K}$ as follows:

$$(\underline{K})^\dagger = \langle\, P,\, \leq,\, \kappa,\, V \,\rangle$$

where:

– for every $\alpha \in P$, if $\iota(\alpha) = (\mathcal{D}^\alpha, \cdot^\alpha)$ then

$$\kappa(\alpha) = \langle \mathcal{D}^\alpha,\, (R_1)^\alpha,\, \ldots,\, (R_n)^\alpha \,\rangle$$

– for every $\alpha \in P$ and for every $p_i \in \mathcal{PV}$

$$V(\alpha, p_i) = (C_i)^{\alpha}$$

It is easy to check that $(\underline{K})^{\dagger}$ is a $\mathbf{FS}_n$-standard model. In particular, properties (FS1) and (FS2) follows, respectively, from properties (K1) and (K4) of $\mathcal{KALC}$-models. Obviously, being $\mathrm{Mod}_{\mathcal{KALC}} \subseteq \mathrm{Mod}_{\mathcal{KALC}^{\infty}}$, the above translation also applies to $\mathcal{KALC}$-models.

It is easy to prove, by induction on the structure of concept formulas, the following result:

Theorem 4.4
Let $H \in \mathcal{L}^{c}$ be a concept formula and let $\underline{K}$ be a $\mathcal{KALC}^{\infty}$-model. Then $\underline{K} \Vdash H$ iff $(\underline{K})^{\dagger} \overset{\bullet}{\Vdash} (H)^{\dagger}$. □

Now, let H be a concept formula of $\mathcal{L}^{c}$. If $H \notin \mathrm{Log}_{\mathcal{KALC}^{\infty}}$ then there exists a model $\underline{K} \in \mathrm{Mod}_{\mathcal{KALC}^{\infty}}$ such that $\underline{K} \nVdash H$ which implies that $(\underline{K})^{\dagger} \overset{\bullet}{\nVdash} (H)^{\dagger}$. Hence $(H)^{\dagger} \notin \mathrm{Log}_{\mathbf{FS}_n}$. Since, as noticed above, $(.)^{\dagger}$ is an onto mapping on $\mathcal{ML}_n$, this implies $\mathrm{Log}_{\mathbf{FS}_n} \subseteq \mathrm{Log}_{\mathcal{KALC}^{\infty}}$.

Now, considering the translation $(\mathrm{Kur})^{\dagger}$ of the Kuroda principle, we call $\mathrm{Kur}_{\Box}$:

$$\mathrm{Kur}_{\Box} \quad = \quad \Box_i \neg\neg A \rightarrow \neg\neg \Box_i A$$

As proved in [Simpson(1994), Gabbay *et al.*(2003)] this formula can not belong to $\mathrm{Log}_{\mathbf{FS}_n}$[2]. By this fact we deduce that, while every instance of $\mathrm{Kur}_{\Box}$ belongs to $(\mathrm{Log}_{\mathcal{KALC}^{\infty}})^{\dagger}$, their translations do not belong to $\mathrm{Log}_{\mathbf{FS}_n}$. Hence:

Theorem 4.5
$\mathrm{Log}_{\mathbf{FS}_n} \subset (\mathrm{Log}_{\mathcal{KALC}^{\infty}})^{\dagger}$. □

[2]Indeed, [Simpson(1994), Gabbay *et al.*(2003)] prove that the uni-modal version $\Box\neg\neg A \rightarrow \neg\neg\Box A$ of this principle does not hold in **FS** by showing a **FS**-standard counter-model for this formula. However, it is easy to adapt such a counter model to the multi-modal case.

5

Services composition in $\mathcal{KALC}$

In this chapter we propose an application for the constructive semantics of $\mathcal{KALC}$ in the context of *Semantic Web services*. In brief, Semantic Web services consists of a semantic description of the capabilities and the structure of services in the languages of the Semantic Web. The current proposals for the representation of Semantic Web services, as OWL-S [Martin *et al.*(2007)], view services as processes with pre- and post- conditions and effects. Pre- and post- conditions describe the requirements and output of a service, while the representation of the process associated with a service describe the interaction with other given services.

In the following, we show how constructive description logics provide a natural framework to represent Semantic Web services and to study the problem of *service composition*, one of the main problems in the context of Web services. Intuitively, the problem can be stated as follows:

> *Given a composition goal, represented as a service with pre- and post- conditions, compose the available services so to satisfy the goal.*

Obviously in this context the challenge is to provide tools to support the definition of the composite service or, at best, to automatize the entire composition process. In the following sections we discuss the problem of Semantic Web services composition in an information terms semantics for $\mathcal{KALC}$. After introducing the definitions for the constructive semantics and a sound natural deduction calculus for it, we present a calculus for the definition of services compositions and we study its properties. Part of the work described in this chapter is presented in [Bozzato and Ferrari(2010a), Bozzato and Ferrari(2010b)].

5.1 Information terms semantics

In the following we give an information terms semantics for $\mathcal{KALC}$: we call $\mathcal{BCDL}_{\mathcal{K}}$ the logic corresponding to this semantics. We assume the definitions for classical model $\mathcal{M}$ for $\mathcal{L}_{\mathcal{N}}$ and classical validity as stated in the context of $\mathcal{KALC}$.

In order to provide a formalization for such semantics, we need to extend the language for $\mathcal{KALC}$. Let VAR be a denumerable set of *individual variables*. Concepts and formulas of the language $\mathcal{L}$ for $\mathcal{BCDL}_{\mathcal{K}}$ are defined as follows:

$$\begin{array}{rcl} C, D & ::= & \bot \mid A \mid C \sqcap D \mid C \sqcup D \mid C \rightarrow D \mid \exists R.C \mid \forall R.C \\ H & ::= & (c, d) : R \mid c : C \mid C \sqsubseteq D \end{array}$$

where $c, d \in \texttt{NI} \cup \texttt{VAR}$, $A \in \texttt{NC}$ and $R \in \texttt{NR}$. Basically, formulas of $\mathcal{BCDL}_{\mathcal{K}}$ are defined exactly as in the language for $\mathcal{KALC}$, but variables may appear in place of individual names. The introduction of variables in the language for $\mathcal{BCDL}_{\mathcal{K}}$ is needed in order to model the input parameters of services in our formalization.

The constructive interpretation of $\mathcal{BCDL}_{\mathcal{K}}$ is based on the notion of *information term* [Ferrari *et al.*(2010a)]. Intuitively, an information term α for a closed formula K is a structured object that provides a justification for the validity of K in a classical model, in the spirit of the *BHK interpretation* of logical connectives [Troelstra(1999)]. Information terms are inductively defined on the structure of the closed formulas, starting from the constant symbol tt associated to atomic formulas. The meaning and the correct reading of an information term is provided by the related formula. For instance, the truthness of an existential formula $c : \exists R.C$ in a classical model $\mathcal{M}$ can be explained by its information term (d, α), that explicitly provides the witness d such that $(c^{\mathcal{M}}, d^{\mathcal{M}}) \in R^{\mathcal{M}}$ and $d^{\mathcal{M}} \in C^{\mathcal{M}}$; moreover, the information term α recursively explains why $d^{\mathcal{M}} \in C^{\mathcal{M}}$. Given $\mathcal{N} \subseteq \texttt{NI}$ and a closed formula K of $\mathcal{L}_{\mathcal{N}}$, we define the set of information terms $\mathrm{IT}_{\mathcal{N}}(K)$ by induction on K as follows.

$$\begin{array}{l} \mathrm{IT}_{\mathcal{N}}(K) \;=\; \{\texttt{tt}\}, \text{ if } K \text{ is a simple formula} \\ \mathrm{IT}_{\mathcal{N}}(c : C_1 \sqcap C_2) \;=\; \{\, (\alpha, \beta) \mid \alpha \in \mathrm{IT}_{\mathcal{N}}(c : C_1) \text{ and } \beta \in \mathrm{IT}_{\mathcal{N}}(c : C_2) \,\} \\ \mathrm{IT}_{\mathcal{N}}(c : C_1 \sqcup C_2) \;=\; \{\, (k, \alpha) \mid k \in \{1, 2\} \text{ and } \alpha \in \mathrm{IT}_{\mathcal{N}}(c : C_k) \,\} \\ \mathrm{IT}_{\mathcal{N}}(c : C \rightarrow D) \;=\; \{\, \phi : \mathrm{IT}_{\mathcal{N}}(c : C) \rightarrow \mathrm{IT}_{\mathcal{N}}(c : D) \,\} \\ \mathrm{IT}_{\mathcal{N}}(c : \exists R.C) \;=\; \{\, (d, \alpha) \mid d \in \mathcal{N} \text{ and } \alpha \in \mathrm{IT}_{\mathcal{N}}(d : C) \,\} \\ \mathrm{IT}_{\mathcal{N}}(c : \forall R.C) \;=\; \{\, \phi : \mathcal{N} \rightarrow \bigcup_{d \in \mathcal{N}} \mathrm{IT}_{\mathcal{N}}(d : C) \mid \phi(d) \in \mathrm{IT}_{\mathcal{N}}(d : C) \,\} \\ \mathrm{IT}_{\mathcal{N}}(C \sqsubseteq D) \;=\; \{\, \phi : \bigcup_{d \in \mathcal{N}} \mathrm{IT}_{\mathcal{N}}(d : C) \rightarrow \bigcup_{d \in \mathcal{N}} \mathrm{IT}_{\mathcal{N}}(d : D) \,\} \end{array}$$

Let $\mathcal{M}$ be a model for $\mathcal{L}_{\mathcal{N}}$, K a closed formula of $\mathcal{L}_{\mathcal{N}}$ and $\eta \in \mathrm{IT}_{\mathcal{N}}(K)$. We define the *realizability relation* $\mathcal{M} \blacktriangleright \langle \eta \rangle\, K$ by induction on the structure of K.

$$\begin{array}{l} \mathcal{M} \blacktriangleright \langle \texttt{tt} \rangle\, K \;\text{ iff }\; \mathcal{M} \models K \\ \mathcal{M} \blacktriangleright \langle (\alpha, \beta) \rangle\, c : C_1 \sqcap C_2 \;\text{ iff }\; \mathcal{M} \blacktriangleright \langle \alpha \rangle\, c : C_1 \text{ and } \mathcal{M} \blacktriangleright \langle \beta \rangle\, c : C_2 \\ \mathcal{M} \blacktriangleright \langle (k, \alpha) \rangle\, c : C_1 \sqcup C_2 \;\text{ iff }\; \mathcal{M} \blacktriangleright \langle \alpha \rangle\, c : C_k \\ \mathcal{M} \blacktriangleright \langle \phi \rangle\, c : C \rightarrow D \;\text{ iff }\; \text{for every } \alpha \in \mathrm{IT}_{\mathcal{N}}(c : C) \text{ if } \mathcal{M} \blacktriangleright \langle \alpha \rangle\, c : C \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{then } \mathcal{M} \blacktriangleright \langle \phi(\alpha) \rangle\, d : D \end{array}$$

$\mathcal{M} \blacktriangleright \langle (d, \alpha) \rangle\, c : \exists R.C$ iff $\mathcal{M} \models (c, d) : R$ and $\mathcal{M} \blacktriangleright \langle \alpha \rangle\, d : C$

$\mathcal{M} \blacktriangleright \langle \phi \rangle\, c : \forall R.C$ iff $\mathcal{M} \models c : \forall R.C$ and, for every $d \in \mathcal{N}$,
$\mathcal{M} \models (c, d) : R$ implies $\mathcal{M} \blacktriangleright \langle \phi(d) \rangle\, d : C$

$\mathcal{M} \blacktriangleright \langle \phi \rangle\, C \sqsubseteq D$ iff $\mathcal{M} \models C \sqsubseteq D$ and, for every $d \in \mathcal{N}$ and $\alpha \in \mathrm{IT}_{\mathcal{N}}(d : C)$,
$\mathcal{M} \blacktriangleright \langle \alpha \rangle\, d : C$ implies $\mathcal{M} \blacktriangleright \langle \phi(\alpha) \rangle\, d : D$

If Γ is a finite set of closed formulas of $\mathcal{L}_{\mathcal{N}}$, $\mathrm{IT}_{\mathcal{N}}(\Gamma)$ denotes the set of functions $\overline{\eta}$ mapping each $K \in \Gamma$ in an element $\overline{\eta}(K) \in \mathrm{IT}_{\mathcal{N}}(K)$. $\mathcal{M} \blacktriangleright \langle \overline{\eta} \rangle\, \Gamma$ iff, $\mathcal{M} \blacktriangleright \langle \overline{\eta}(K) \rangle\, K$ for every $K \in \Gamma$.

Now, we introduce the example we refer to throughout this chapter.

Example 5 (Theory definition)
Our example is a reinterpretation in our context of the "purchase and delivery service" example of [Traverso and Pistore(2004)]. The example presents a system composed by three agents: a User, a Shipper and a Producer agent. The Shipper and the Producer provide the User with services to request and obtain offers for the delivery and the purchase of a product: the goal of the example is to combine the services of the two agents in order to provide the User with a single service to request the production and shipping of a product. We begin by defining the theory T_{PS} that models our system.

$\texttt{AcceptedRequest} \sqsubseteq \texttt{Request}$
$\texttt{RefusedRequest} \sqsubseteq \texttt{Request} \sqcap \neg\texttt{AcceptedRequest}$

$\texttt{ProduceRequest} \sqsubseteq \texttt{Request}$
$\texttt{AcceptedProduceRequest} \sqsubseteq \texttt{ProduceRequest} \sqcap \texttt{AcceptedRequest}$

$\texttt{ShippingRequest} \sqsubseteq \texttt{Request}$
$\texttt{AcceptedShippingRequest} \sqsubseteq \texttt{ShippingRequest} \sqcap \texttt{AcceptedRequest}$

$\texttt{ProduceOffer} \sqsubseteq \texttt{Offer}$
$\texttt{ShippingOffer} \sqsubseteq \texttt{Offer}$

The theory states that a request can be classified as accepted or refused by one of the two agents: we further characterize offers, requests and accepted requests by the agent to which they refer. To relate requests to offers and to the information that they convey, we include in T_{PS} the following axioms:

$\texttt{Offer} \sqsubseteq \forall \texttt{hasCost.Price}$
$\texttt{Request} \sqsubseteq \forall \texttt{hasOffer.Offer}$

$\texttt{ShippingRequest} \sqsubseteq \forall \texttt{hasDestination.Location}$
$\texttt{ProduceRequest} \sqsubseteq \forall \texttt{hasProduct.Product}$

In other words, every offer in `Offer` specifies its `Price` by the role `hasCost`; requests relate to their offers by the role `hasOffer`; finally, a `ShippingRequest` contains information about the `Location` to where to ship by the role `hasDestination` and a `ProduceRequest` describes the `Product` to buy by the role `hasProduct`.

Given a finite set of individual names $\mathcal{N}$, we assume to have a suitable $\eta \in \text{IT}_{\mathcal{N}}(\mathsf{T}_{PS})$ justifying the validity of T_{PS} with respect to elements of $\mathcal{N}$. Note that T_{PS} only represents a TBox, thus information terms of its subsumptions are functions mapping information terms of the included concept in those of the including concept. If we assume to store assertions of an ABox over $\mathcal{N}$ in some kind of database (e.g., a relational database or the data part of a logic program), the functions for each of these information terms can be implemented as query prototypes (to be instantiated with individuals of $\mathcal{N}$) over the database. $\diamond$

The compatibility between information terms and classical semantics is explained by the following proposition.

Proposition 5.1
Let K be a closed formula of $\mathcal{L}_{\mathcal{N}}$ and let $\eta \in \text{IT}_{\mathcal{N}}(K)$. For every model $\mathcal{M}$, if $\mathcal{M} \blacktriangleright \langle \eta \rangle K$, then $\mathcal{M} \models K$. $\square$

The proposition directly follows by induction on the formula K.

The previous definition of realizability allow us to define a constructive logical consequence relation in $\mathcal{BCDL}_{\mathcal{K}}$:

Definition 5.2 (Constructive consequence)
Let $\Gamma \cup \{K\}$ be a set of closed formulas of $\mathcal{L}_{\mathcal{N}}$. We say that K is a *constructive consequence* *of Γ, and we write $\Gamma \overset{c}{\models} K$, iff, for every $\overline{\gamma} \in \text{IT}_{\mathcal{N}}(\Gamma)$, there exists $\eta \in \text{IT}_{\mathcal{N}}(K)$ such that, for every model $\mathcal{M}$ for $\mathcal{L}_{\mathcal{N}}$, $\mathcal{M} \blacktriangleright \langle \overline{\gamma} \rangle \Gamma$ implies $\mathcal{M} \blacktriangleright \langle \eta \rangle K$.*

Thus, the relation $\Gamma \overset{c}{\models} K$ implicitly defines a map $\Phi_{\mathcal{N}}$ from $\text{IT}_{\mathcal{N}}(\Gamma)$ to $\text{IT}_{\mathcal{N}}(K)$ such that, for every model $\mathcal{M}$, $\mathcal{M} \blacktriangleright \langle \overline{\gamma} \rangle \Gamma$ implies $\mathcal{M} \blacktriangleright \langle \Phi_{\mathcal{N}}(\overline{\gamma}) \rangle K$. Note that $\Phi_{\mathcal{N}}$ is independent from the choice of the models. We show in the following how such map can be defined as a computable function with respect to proofs of a natural deduction calculus for $\mathcal{BCDL}_{\mathcal{K}}$.

5.2 Natural deduction calculus $\mathcal{ND}_{\mathcal{K}}$

In this section we introduce the natural deduction calculus $\mathcal{ND}_{\mathcal{K}}$ for $\mathcal{BCDL}_{\mathcal{K}}$ whose rules are shown in Figure 5.2. We refer to [Troelstra and Schwichtenberg(1996), Prawitz(1965)] for a detailed presentation of natural deduction calculi and their notation: the calculus we present is adapted from the similar one provided for $\mathcal{BCDL}$ in [Bozzato *et al.*(2007), Ferrari *et al.*(2010a)].

Let us briefly summarize some of the conventions we use in our calculus. We provide *introduction* and *elimination* rules for each of the logical constants in $\mathcal{L}$: while the first ones allow to introduce the corresponding concept constructor to the conclusion, the second ones remove the constructor from the main formula in the premise. The rules for $\neg I$, $\sqcup E$, $\rightarrow I$, $\exists E$ and $\forall I$ allow us to *discharge* some of the assumptions, which means that the conclusion does not depend on the specified assumption: we put between square brackets the discharged assumptions. The

rules for $\exists E$ and $\forall I$ need a side condition on the rule parameter to guarantee their correctness.

A *proof* in $\mathcal{ND}_{\mathcal{K}}$ is a finite tree whose root is the *proof conclusion* and whose leaves are the *proof assumptions*. Each internal node in the proof tree is the consequent of a rule of $\mathcal{ND}_{\mathcal{K}}$, that is applied to subproofs which roots consist of the children of the node. The *depth* $d(\pi)$ of a proof π is defined as the depth of its proof tree, formally:

$$d(\pi) = \begin{cases} 0, \text{ if } \pi \text{ consists of an assumption introduction} \\ max\{d(\pi') \mid \pi' \text{ is an immediate subproof of } \pi\} + 1, \text{ otherwise} \end{cases}$$

We denote with $\pi :: \Gamma \vdash K$ the fact that π is a proof of $\Gamma \vdash K$, that is a proof of a formula K with undischarged assumptions in a set of formulas Γ. We write $\Gamma \mid_{\overline{\mathcal{BCDL}_{\mathcal{K}}}} K$ to denote the fact that there exists a proof $\pi :: \Gamma \vdash K$ in $\mathcal{ND}_{\mathcal{K}}$.

5.2.1 Soundness for $\mathcal{BCDL}_{\mathcal{K}}$

In the following we show that $\mathcal{ND}_{\mathcal{K}}$ is sound with respect to the information term semantics of $\mathcal{BCDL}_{\mathcal{K}}$.

Given a finite subset $\mathcal{N}$ of $\mathtt{NI}$, an $\mathcal{N}$*-substitution* σ is a map $\sigma : \mathtt{VAR} \to \mathcal{N}$. We extend σ to $\mathcal{L}_{\mathcal{N}}$ as usual: if $c \in \mathcal{N}$, $\sigma c = c$; for a formula K of $\mathcal{L}_{\mathcal{N}}$, σK denotes the closed formula of $\mathcal{L}_{\mathcal{N}}$ obtained by replacing every variable x occurring in K with $\sigma(x)$; given a set of formulas Γ, $\sigma\Gamma$ is the set of σK such that $K \in \Gamma$. If $c \in \mathcal{N}$, $\sigma[c/p]$ is the $\mathcal{N}$-substitution σ' such that $\sigma'(p) = c$ and $\sigma'(x) = \sigma(x)$ for $x \neq p$. A $\mathcal{N}$-substitution σ is a *closing substitution* for a set of formulas Γ if $\sigma\Gamma$ is a set of closed formulas.

The proof of the soundness of $\mathcal{ND}_{\mathcal{K}}$ with respect to $\mathcal{BCDL}_{\mathcal{K}}$ is based on the definition of the operator $\Phi^{\pi}_{\mathcal{N}}$. As we show in the following sections, this operator also provide the foundation for the computational interpretation of $\mathcal{ND}_{\mathcal{K}}$ proofs and it will be essential in the definition of service compositions.

Let $\pi :: \Gamma \vdash K$ be a proof of $\mathcal{ND}_{\mathcal{K}}$ on $\mathcal{L}_{\mathcal{N}}$ and σ a closing $\mathcal{N}$-substitution. We define a computable function:

$$\Phi^{\pi}_{\sigma,\mathcal{N}} : \mathrm{IT}_{\mathcal{N}}(\sigma\Gamma) \to \mathrm{IT}_{\mathcal{N}}(\sigma K)$$

by induction on the depth $d(\pi)$ of the proof. If $d(\pi) = 0$, then π only consists of the introduction of the assumption K and $\Phi^{\pi}_{\sigma,\mathcal{N}}$ is the identity function on $\mathrm{IT}_{\mathcal{N}}(\sigma K)$. If K is a simple formula, $\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) = \{\mathtt{tt}\}$. If $d(\pi) > 0$, the function is defined by cases on the last rule r applied in π:

- $r = \bot E$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \mathrm{IT}_{\mathcal{N}}(\sigma\Gamma) \to \mathrm{IT}_{\mathcal{N}}(\sigma K)$ and $\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) = \eta^{+}$, where η^{+} is an element of $\mathrm{IT}_{\mathcal{N}}(\sigma K)$[1].

[1]We note that by the definition of information terms, there exists at least one element in $\mathrm{IT}_{\mathcal{N}}(\sigma K)$ for any $K \in \mathcal{L}_{\mathcal{N}}$, thus it is always possible to find a η^{+} satisfying this condition.

$$\frac{\begin{array}{cc}\Gamma_1 & \Gamma_2\\ \vdots\,\pi_1 & \vdots\,\pi_2\\ t:C & t:\neg C\end{array}}{t:\bot}\ \bot I \qquad \frac{\begin{array}{c}\Gamma\\ \vdots\,\pi'\\ t:\bot\end{array}}{K}\ \bot E \qquad \frac{\begin{array}{c}\Gamma,[t:C]\\ \vdots\,\pi'\\ t:\bot\end{array}}{t:\neg C}\ \neg I \qquad \frac{\begin{array}{cc}C\sqsubseteq D & \Gamma'\\ & \vdots\,\pi'\\ & t:C\end{array}}{t:D}\ \sqsubseteq E$$

$$\frac{\begin{array}{cc}\Gamma_1 & \Gamma_2\\ \vdots\,\pi_1 & \vdots\,\pi_2\\ t:C_1 & t:C_2\end{array}}{t:C_1\sqcap C_2}\ \sqcap I \qquad \frac{\begin{array}{c}\Gamma\\ \vdots\,\pi'\\ t:C_1\sqcap C_2\end{array}}{t:C_k}\ \sqcap E_k \quad k\in\{1,2\}$$

$$\frac{\begin{array}{c}\Gamma\\ \vdots\,\pi'\\ t:C_k\end{array}}{t:C_1\sqcup C_2}\ \sqcup I_k \quad k\in\{1,2\} \qquad \frac{\begin{array}{ccc}\Gamma_1 & \Gamma_2,[t:C_1] & \Gamma_3,[t:C_2]\\ \vdots\,\pi_1 & \vdots\,\pi_2 & \vdots\,\pi_3\\ t:C_1\sqcup C_2 & K & K\end{array}}{K}\ \sqcup E$$

$$\frac{\begin{array}{c}\Gamma,[t:C]\\ \vdots\,\pi'\\ t:D\end{array}}{t:C\to D}\ \to I \qquad \frac{\begin{array}{cc}\Gamma_1 & \Gamma_2\\ \vdots\,\pi_1 & \vdots\,\pi_2\\ t:C & t:C\to D\end{array}}{t:D}\ \to E$$

$$\frac{\begin{array}{cc}\Gamma' & \\ \vdots\,\pi' & \\ (t,u):R & u:C\end{array}}{t:\exists R.C}\ \exists I \qquad \frac{\begin{array}{ccc}\Gamma_1 & \Gamma_2,[(t,p):R,\ p:C] & \\ \vdots\,\pi_1 & \vdots\,\pi_2 & \\ t:\exists R.C & K & \end{array}}{K}\ \exists E$$

where $p\in$ `VAR` does not occur in $\Gamma_2\cup\{K\}$ and $p\neq t$

$$\frac{\begin{array}{c}\Gamma,[(t,p):R]\\ \vdots\,\pi'\\ p:C\end{array}}{t:\forall R.C}\ \forall I$$

where $p\in$ `VAR`, p does not occur in Γ and $p\neq t$

$$\frac{\begin{array}{cc}(s,t):R & \Gamma'\\ & \vdots\,\pi'\\ & s:\forall R.C\end{array}}{t:C}\ \forall E \qquad \frac{\begin{array}{c}\Gamma\\ \vdots\,\pi'\\ t:\forall R.\neg\neg C\end{array}}{t:\neg\neg\forall R.C}\ \text{KUR}$$

Figure 5.1: The rules of the calculus $\mathcal{ND}_{\mathcal{K}}$

– $r = \sqcap I$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \to \text{IT}_{\mathcal{N}}(\sigma(t : C_1 \sqcap C_2))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2) \;=\; (\, \Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1),\, \Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2)\,)$$

– $r = \sqcap E_k$ with $k \in \{1, 2\}$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \to \text{IT}_{\mathcal{N}}(\sigma(t : C_k))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) \;=\; \mathit{Pro}_k(\, \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma})\,)$$

where Pro_k is the k-projection function.

– $r = \sqcup I_k$ with $k \in \{1, 2\}$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \to \text{IT}_{\mathcal{N}}(\sigma(t : C_1 \sqcup C_2))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) \;=\; (\, k,\, \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma})\,)$$

– $r = \sqcup E$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_3) \to \text{IT}_{\mathcal{N}}(\sigma K)$, and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2, \overline{\gamma}_3) \;=\; \begin{cases} \Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2, \alpha), & \text{if } \Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) \;=\; (1, \alpha) \\ \Phi^{\pi_3}_{\sigma,\mathcal{N}}(\overline{\gamma}_3, \beta), & \text{if } \Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) \;=\; (2, \beta) \end{cases}$$

– $r = \to I$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \to \text{IT}_{\mathcal{N}}(\sigma(t : C \to D))$ and

$$\left[\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma})\right](\alpha) \;=\; \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma},\, \alpha), \text{ for every } \alpha \in \text{IT}_{\mathcal{N}}(\sigma(t : C))$$

– $r = \to E$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \to \text{IT}_{\mathcal{N}}(\sigma(t : D))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2) \;=\; \left[\Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2)\right](\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1))$$

– $r = \exists I$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \{\mathtt{tt}\} \times \text{IT}_{\mathcal{N}}(\sigma\Gamma') \to \text{IT}_{\mathcal{N}}(\sigma(t : \exists R.C))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\mathtt{tt}, \overline{\gamma}') \;=\; (\, \sigma u,\, \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}')\,)$$

– $r = \exists E$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \to \text{IT}_{\mathcal{N}}(\sigma K)$. Let $\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) = (c, \alpha)$[2], and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2) \;=\; \Phi^{\pi_2}_{\sigma[c/p],\mathcal{N}}(\overline{\gamma}_2,\, \mathtt{tt},\, \alpha)$$

– $r = \forall I$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \to \text{IT}_{\mathcal{N}}(\sigma(t : \forall R.C))$[3]. Then $\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma})$ is the function defined as follows: for every $c \in \mathcal{N}$

$$\left[\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma})\right](c) \;=\; \Phi^{\pi'}_{\sigma[c/p],\mathcal{N}}(\overline{\gamma},\, \mathtt{tt})$$

[2] By the side condition on p, $(\sigma[c/p])\Gamma_2 = \sigma\Gamma_2$ and $(\sigma[c/p])K = \sigma K$.

[3] Let $c \in \mathcal{N}$, by the side condition on p, $(\sigma[c/p])\Gamma = \sigma\Gamma$ and $(\sigma[c/p])t : \forall R.C = \sigma t : \forall R.C$.

– $r = \forall E$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \{\mathtt{tt}\} \times \text{IT}_{\mathcal{N}}(\sigma\Gamma') \rightarrow \text{IT}_{\mathcal{N}}(\sigma(t : C))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\mathtt{tt}, \overline{\gamma}') \;=\; \left[\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}')\right](\sigma t)$$

– $r = \sqsubseteq E$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(C \sqsubseteq D) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma') \rightarrow \text{IT}_{\mathcal{N}}(\sigma(t : D))$. For $\phi \in \text{IT}_{\mathcal{N}}(C \sqsubseteq D)$, we define

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\phi, \overline{\gamma}') \;=\; \phi(\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}'))$$

With the following lemmas we show the soundness of $\mathcal{ND}_{\mathcal{K}}$ with respect to the $\mathcal{ALC}$-validity and realizability relations.

Lemma 5.3
Let $\Gamma \cup \{K\} \subseteq \mathcal{L}_{\mathcal{N}}$ and let $\pi :: \Gamma \vdash K$ be a proof of $\mathcal{ND}_{\mathcal{K}}$ on $\mathcal{L}_{\mathcal{N}}$. For every model $\mathcal{M}$, if $\mathcal{M} \models \Gamma$ then $\mathcal{M} \models K$. □

The lemma can be easily proved by showing the relations of $\mathcal{ND}_{\mathcal{K}}$ with a suitable natural deduction calculus for $\mathcal{ALC}$.

Lemma 5.4
Let $\Gamma \cup \{K\} \subseteq \mathcal{L}_{\mathcal{N}}$, let σ be a closing $\mathcal{N}$-substitution and let $\pi :: \Gamma \vdash K$ be a proof of $\mathcal{ND}_{\mathcal{K}}$ on $\mathcal{L}_{\mathcal{N}}$. Then, for every $\overline{\gamma} \in \text{IT}_{\mathcal{N}}(\sigma\Gamma)$ and for every model $\mathcal{M}$ for $\mathcal{L}_{\mathcal{N}}$, $\mathcal{M} \blacktriangleright \langle\overline{\gamma}\rangle\, \sigma\Gamma$ implies $\mathcal{M} \blacktriangleright \langle\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma})\rangle\, \sigma K$.

Proof: We show the lemma by induction on the depth of π. If π only consists of an assumption introduction, then $\Gamma = \{K\}$ and $\overline{\gamma} = \gamma \in \text{IT}_{\mathcal{N}}(\sigma K)$. By definition, $\Phi^{\pi}_{\sigma,\mathcal{N}}$ is the identity function, then the assertion directly follows. If $d(\pi) > 0$, then let r be the last rule applied in π: we prove the assertion by cases on the rule r.

If $r = \sqcap I$, by induction hypothesis over π_1 and π_2:

$$\mathcal{M} \blacktriangleright \langle\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1)\rangle\, \sigma(t : C_1) \qquad \mathcal{M} \blacktriangleright \langle\Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2)\rangle\, \sigma(t : C_2)$$

Thus, by definition:

$$\mathcal{M} \blacktriangleright \langle(\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1), \Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2))\rangle\, \sigma(t : C_1 \sqcap C_2)$$

that is $\mathcal{M} \blacktriangleright \langle\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2)\rangle\, \sigma(t : C_1 \sqcap C_2)$.

If $r = \sqcap E_k$, by induction hypothesis on π':

$$\mathcal{M} \blacktriangleright \langle\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma})\rangle\, \sigma(t : C_1 \sqcap C_2)$$

with $\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}) = (\alpha, \beta)$. By definition it holds that $\mathcal{M} \blacktriangleright \langle\alpha\rangle\, \sigma(t : C_1)$ and $\mathcal{M} \blacktriangleright \langle\beta\rangle\, \sigma(t : C_2)$. It follows that:

$$\mathcal{M} \blacktriangleright \langle Pro_k(\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}))\rangle\, \sigma(t : C_k)$$

that is $\mathcal{M} \blacktriangleright \langle\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma})\rangle\, \sigma(t : C_k)$, with $k \in \{1, 2\}$.

If $r = \sqcup I_k$, by induction hypothesis on π':

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}) \rangle \, \sigma(t : C_k)$$

with $k \in \{1, 2\}$. By definition, it holds:

$$\mathcal{M} \blacktriangleright \langle (i, \, \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma})) \rangle \, \sigma(t : C_1 \sqcup C_2)$$

that is $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) \rangle \, \sigma(t : C_1 \sqcup C_2)$.

If $r = \sqcup E$, by induction hypothesis on π_1:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) \rangle \, \sigma(t : C_1 \sqcup C_2)$$

with $\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) = (k, \alpha)$ and $k \in \{1, 2\}$. Suppose $k = 1$: by definition, it holds $\mathcal{M} \blacktriangleright \langle \alpha \rangle \, \sigma(t : C_1)$. By induction hypothesis on π_2, it holds:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2, \, \alpha) \rangle \, \sigma K$$

The case with $k = 2$ is similar: thus, in both cases $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2, \overline{\gamma}_3) \rangle \, \sigma K$.

If $r = \!\!\rightarrow I$, suppose that α is a generic element of $\text{IT}_{\mathcal{N}}(\sigma(t : C))$ such that $\mathcal{M} \blacktriangleright \langle \alpha \rangle \, \sigma(t : C)$. By induction hypothesis on π':

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}, \alpha) \rangle \, \sigma(t : D)$$

Given that, by definition, ϕ is defined as the function such that $\phi(\alpha) = \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}, \alpha)$ and α has been chosen arbitrarily, by definition we have:

$$\mathcal{M} \blacktriangleright \langle \phi \rangle \, \sigma(t : C \rightarrow D)$$

that is $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) \rangle \, \sigma(t : C \rightarrow D)$.

If $r = \!\!\rightarrow E$, by induction hypothesis on π_1 and π_2:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) \rangle \, \sigma(t : C) \qquad \mathcal{M} \blacktriangleright \langle \Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2) \rangle \, \sigma(t : C \rightarrow D)$$

Then, by definition:

$$\mathcal{M} \blacktriangleright \langle [\Phi^{\pi_2}_{\sigma,\mathcal{N}}(\overline{\gamma}_2)](\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1)) \rangle \, \sigma(t : D)$$

that is $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2) \rangle \, \sigma(t : D)$.

If $r = \exists I$, by induction hypothesis, $\mathcal{M} \blacktriangleright \langle \mathtt{tt} \rangle \, \sigma((t, u) : R)$ hence we have $\mathcal{M} \models \sigma((t, u) : R)$. On the other hand, by induction hypothesis on π', we have:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}') \rangle \, \sigma(u : C)$$

Then, by definition:

$$\mathcal{M} \blacktriangleright \langle (\sigma u, \, \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}')) \rangle \, \sigma(t : \exists R.C)$$

thus $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\mathtt{tt}, \overline{\gamma}') \rangle\, \sigma(t : \exists R.C)$.

If $r = \exists E$, by induction hypothesis on π_1 it holds:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) \rangle\, \sigma(t : \exists R.C)$$

with $\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1) = (c, \alpha)$. By definition, it holds $\mathcal{M} \blacktriangleright \langle \alpha \rangle\, \sigma(c : C)$ and $\mathcal{M} \models \sigma((t, c) : R)$. Let us consider the proof π_2 and the closed $\mathcal{N}$-substitution $\sigma' = \sigma[c/p]$. Then, by the assumptions on the parameter p, we have that $\sigma'\Gamma = \sigma\Gamma$ and $\sigma' K = \sigma K$. Hence:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi_2}_{\sigma',\mathcal{N}}(\overline{\gamma}_2, \mathtt{tt}, \alpha) \rangle\, \sigma' K$$

and $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \overline{\gamma}_2) \rangle\, \sigma K$.

If $r = \forall I$, let us consider the set:

$$Y = \{c \in \mathcal{N} \mid c^{\mathcal{M}} \in \mathcal{D}^{\mathcal{M}} \text{ and } (\sigma t^{\mathcal{M}}, c^{\mathcal{M}}) \in R^{\mathcal{M}}\}$$

and, for any $c \in Y$, the closed $\mathcal{N}$-substitution $\sigma_c = \sigma[c/p]$. By the assumptions on the parameter p, for any $c \in Y$, $\sigma_c\Gamma = \sigma\Gamma$ and $\sigma_c(t : \forall R.C) = \sigma(t : \forall R.C)$. By induction hypothesis on π', for any $c \in Y$ it holds:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi'}_{\sigma_c,\mathcal{N}}(\overline{\gamma}, \mathtt{tt}) \rangle\, \sigma_c(p : C)$$

We note that $\Phi^{\pi'}_{\sigma_c,\mathcal{N}}(\overline{\gamma}, \mathtt{tt}) = [\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma})](c)$. Now, the assertion is proved if we can show that $\mathcal{M} \models \sigma(t : \forall R.C)$: given that by Proposition 5.1 it holds $\mathcal{M} \models \sigma\Gamma$, by Lemma 5.3 we have that $\mathcal{M} \models \sigma(t : \forall R.C)$.

If $r = \forall E$, by induction hypothesis:

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}') \rangle\, \sigma(t : \forall R.C) \qquad \mathcal{M} \blacktriangleright \langle \mathtt{tt} \rangle\, \sigma((s, t) : R)$$

By Proposition 5.1, $\mathcal{M} \models \sigma((s, t) : R)$ thus it holds:

$$\mathcal{M} \blacktriangleright \langle [\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}')](\sigma t) \rangle\, \sigma(t : C)$$

hence $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\mathtt{tt}, \overline{\gamma}') \rangle\, \sigma(t : C)$.

If $r = \sqsubseteq E$, by induction hypothesis:

$$\mathcal{M} \blacktriangleright \langle \phi \rangle\, C \sqsubseteq D$$

with $\phi \in \text{IT}_{\mathcal{N}}(C \sqsubseteq D)$. Moreover, by induction hypothesis on π',

$$\mathcal{M} \blacktriangleright \langle \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}') \rangle\, \sigma(t : C)$$

By definition, thus it holds that:

$$\mathcal{M} \blacktriangleright \langle \phi(\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}')) \rangle\, \sigma(t : D)$$

that is $\mathcal{M} \blacktriangleright \langle \Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) \rangle\, \sigma(t : D)$.

If $r = \bot E$, by induction hypothesis on π', we would have that $\mathcal{M} \models \bot$, which is obviously an absurd. Thus we deduce that it can not exist $\overline{\gamma} \in \text{IT}_{\mathcal{N}}(\sigma\Gamma)$ such that $\mathcal{M} \blacktriangleright \langle\overline{\gamma}\rangle\, \sigma\Gamma$.

If $r = \bot I$, the assertion holds by the observations of the previous case, since it does not exist $\mathcal{M}$ such that $\mathcal{M} \models \sigma K$ and $\mathcal{M} \models \sigma\neg K$.

If $r = \neg I$, we have that $\mathcal{M} \blacktriangleright \langle\alpha\rangle\, \sigma t : C$ can not hold, since this would imply $\mathcal{M} \blacktriangleright \langle\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}', \alpha)\rangle\, \sigma(t : \bot)$ and hence $\mathcal{M} \models \sigma(t : \bot)$ by Proposition 5.1.

If $r = \text{KUR}$, by induction hypothesis

$$\mathcal{M} \blacktriangleright \langle\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}')\rangle\, \sigma(t : \forall R.\neg\neg C)$$

By semantics definition, it holds that $\mathcal{M} \models \sigma(t : \forall R.\neg\neg C)$. In the classical semantics for $\mathcal{ALC}$, one can show that this is equivalent to $\mathcal{M} \models \sigma(t : \neg\neg\forall R.C)$. This implies that $\mathcal{M} \blacktriangleright \langle\texttt{tt}\rangle\, \sigma(t : \neg\neg\forall R.C)$ and the assertion holds. □

From the previous lemmas, if $\Phi^{\pi}_{\mathcal{N}} = \Phi^{\pi}_{\sigma,\mathcal{N}}$ for any $\mathcal{N}$-substitution σ, we get:

Theorem 5.5
Let $\Gamma \cup \{K\}$ be a set of closed formulas with $\Gamma \cup \{K\} \subseteq \mathcal{L}_{\mathcal{N}}$ and let $\pi :: \Gamma \vdash K$ be a proof of $\mathcal{ND}_{\mathcal{K}}$ on $\mathcal{L}_{\mathcal{N}}$. Then:

(i) $\Gamma \models K$;

(ii) For every $\overline{\gamma} \in \text{IT}_{\mathcal{N}}(\Gamma)$ and for every model $\mathcal{M}$ for $\mathcal{L}_{\mathcal{N}}$, $\mathcal{M} \blacktriangleright \langle\overline{\gamma}\rangle\, \Gamma$ implies $\mathcal{M} \blacktriangleright \langle\Phi^{\pi}_{\mathcal{N}}(\overline{\gamma})\rangle\, K$. □

As a direct consequence and by the definition of the constructive consequence relation, the Soundness theorem can be stated as follows:

Theorem 5.6 (Soundness)
$\Gamma \underset{\mathcal{BCDL}_{\mathcal{K}}}{\vdash} K$ *implies* $\Gamma \overset{c}{\models} K$. □

We remark that the proof of the previous result by the definition of the function $\Phi^{\pi}_{\mathcal{N}}$ provide us with an effective method to extract information from proofs of $\mathcal{ND}_{\mathcal{K}}$. Indeed, according to the *proofs-as-programs* paradigm, proofs are seen as programs to solve the problem represented by the proved formula: in other words, they represent functions transforming the information terms for premises in information terms for the consequences.

We also note that we can easily generalize our semantics for concept conjunctions and disjunctions to the n-ary case: formally, given $\mathcal{N} \subseteq \texttt{NI}$ and a closed formula $K \in \mathcal{L}_{\mathcal{N}}$, we can extend the definition of information term $\text{IT}_{\mathcal{N}}(K)$ as follows:

$$\text{IT}_{\mathcal{N}}(c : C_1 \sqcap \cdots \sqcap C_n) \;=\; \{\, (\alpha_1, \ldots, \alpha_n) \mid \alpha_k \in \text{IT}_{\mathcal{N}}(c : C_k) \text{ for } k \in \{1, \ldots, n\}\}$$

$$\text{IT}_{\mathcal{N}}(c : C_1 \sqcup \cdots \sqcup C_n) \;=\; \{\, (k, \alpha) \mid k \in \{1, \ldots, n\} \text{ and } \alpha \in \text{IT}_{\mathcal{N}}(c : C_k)\,\}$$

Given a model $\mathcal{M}$ of $\mathcal{L}_{\mathcal{N}}$, we can thus extend the definition of realizability relation for the above cases:

$$\mathcal{M} \blacktriangleright \langle(\alpha_1, \ldots, \alpha_n)\rangle\, c : C_1 \sqcap \cdots \sqcap C_n \;\text{ iff }\; \mathcal{M} \blacktriangleright \langle\alpha_k\rangle\, c : C_k \text{ for every } k \in \{1, \ldots, n\}$$

$$\mathcal{M} \blacktriangleright \langle(k, \alpha)\rangle\, c : C_1 \sqcup \cdots \sqcup C_n \;\text{ iff }\; \mathcal{M} \blacktriangleright \langle\alpha\rangle\, c : C_k$$

$$\frac{\begin{array}{ccc}\Gamma_1 & & \Gamma_n \\ \vdots\,\pi_1 & & \vdots\,\pi_n \\ t : C_1 & \ldots & t : C_n\end{array}}{t : C_1 \sqcap \cdots \sqcap C_n}\,\sqcap I \qquad \frac{\begin{array}{c}\Gamma \\ \vdots\,\pi' \\ t : C_1 \sqcap \cdots \sqcap C_n\end{array}}{t : C_k}\,\sqcap E_k \quad k \in \{1, \ldots, n\}$$

$$\frac{\begin{array}{c}\Gamma \\ \vdots\,\pi' \\ t : C_k\end{array}}{t : C_1 \sqcup \cdots \sqcup C_n}\,\sqcup I_k \quad k \in \{1, \ldots, n\} \qquad \frac{\begin{array}{ccccc}\Gamma' & & \Gamma_1, [t : C_1] & & \Gamma_n, [t : C_n] \\ \vdots\,\pi' & & \vdots\,\pi_1 & & \vdots\,\pi_n \\ t : C_1 \sqcup \cdots \sqcup C_n & & K & \ldots & K\end{array}}{K}\,\sqcup E$$

Figure 5.2: Rules of $\mathcal{ND}_{\mathcal{K}}$ for n-ary concept constructors

According to these definitions, we can also extend $\mathcal{ND}_{\mathcal{K}}$ to treat the cases of such n-ary constructors: the new rules are presented in Figure 5.2. We extend the definition of the function $\Phi^{\pi}_{\sigma,\mathcal{N}}$ to consider such rules. Given $\pi :: \Gamma \vdash K$ a proof of $\mathcal{ND}_{\mathcal{K}}$ on $\mathcal{L}_{\mathcal{N}}$ and σ a closing $\mathcal{N}$-substitution, if $d(\pi) > 0$, we define $\Phi^{\pi}_{\sigma,\mathcal{N}}$ on the last rule r applied in π:

– $r = \sqcap I$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \cdots \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_n) \rightarrow \text{IT}_{\mathcal{N}}(\sigma(t : C_1 \sqcap \cdots \sqcap C_n))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}_1, \ldots, \overline{\gamma}_n) \;=\; (\,\Phi^{\pi_1}_{\sigma,\mathcal{N}}(\overline{\gamma}_1), \ldots, \Phi^{\pi_n}_{\sigma,\mathcal{N}}(\overline{\gamma}_n)\,)$$

– $r = \sqcap E_k$ with $k \in \{1, \ldots, n\}$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \rightarrow \text{IT}_{\mathcal{N}}(\sigma(t : C_k))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) \;=\; \mathit{Pro}_k(\,\Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma})\,)$$

where Pro_k is the k-projection function.

– $r = \sqcup I_k$ with $k \in \{1, \ldots, n\}$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \rightarrow \text{IT}_{\mathcal{N}}(\sigma(t : C_1 \sqcup \cdots \sqcup C_n))$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}) \;=\; (\,k,\, \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma})\,)$$

– $r = \sqcup E$. Then, $\Phi^{\pi}_{\sigma,\mathcal{N}} : \text{IT}_{\mathcal{N}}(\sigma\Gamma') \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \cdots \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_n) \rightarrow \text{IT}_{\mathcal{N}}(\sigma K)$ and

$$\Phi^{\pi}_{\sigma,\mathcal{N}}(\overline{\gamma}', \overline{\gamma}_1, \ldots, \overline{\gamma}_n) \;=\; \Phi^{\pi_k}_{\sigma,\mathcal{N}}(\overline{\gamma}_k, \alpha), \quad \text{if } \Phi^{\pi'}_{\sigma,\mathcal{N}}(\overline{\gamma}') \;=\; (k, \alpha)$$

In all of the other cases, the definition of the function is not modified.

It is easy to prove that such n-ary extensions to the semantics and the natural deduction calculus preserve the results of Lemma 5.4 and thus the soundness of the extended calculus. In particular, this generalization of concept constructors to n-ary concepts is useful in the definition of service composition rules provided in the following sections.

5.2.2 Soundness for $\mathcal{KALC}$

Given a $\mathcal{KALC}$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$ and $\alpha \in P$, an *α-substitution* is a function $\sigma : \mathtt{VAR} \to \mathtt{NI} \cup \mathcal{D}^\alpha$. Note that, by the monotonicity property, we obtain that each α-substitution is also a β-substitution for every $\beta \in P$ such that $\alpha \leq \beta$. We extend the definition of α-substitution to formulas and sets of formulas as previously defined for $\mathcal{N}$-substitutions.

Theorem 5.7
Let $\Gamma \cup \{K\} \subseteq \mathcal{L}_\mathcal{N}$ and let $\pi :: \Gamma \vdash K$ be a proof of $\mathcal{ND}_\mathcal{K}$ on $\mathcal{L}_\mathcal{N}$. Then, for every $\mathcal{KALC}$-model $\underline{K} = \langle P, \leq, \rho, \iota \rangle$, for every $\alpha \in P$ and every α-substitution σ, $\alpha \Vdash \sigma\Gamma$ implies $\alpha \Vdash \sigma K$.

Proof: We prove the assertion by induction on the depth of the proof π. If $d(\pi) = 0$, then π only consists of an assumption introduction and the assertion immediately holds. If $d(\pi) > 0$, the proof goes by cases on the last rule r applied in π.

Let $r = \sqcap I$. Then by induction hypothesis, we have that $\Gamma_1 \overset{\mathrm{k}}{\models} t : C_1$ and $\Gamma_2 \overset{\mathrm{k}}{\models} t : C_2$, with $\Gamma = \Gamma_1 \cup \Gamma_2$. Then if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$ then $\alpha \Vdash \sigma(t : C_1)$ and $\alpha \Vdash \sigma(t : C_2)$ in $\underline{K}$, thus $\alpha \Vdash \sigma(t : C_1 \sqcap C_2)$ in $\underline{K}$ and the assertion holds.

Let $r = \sqcap E$. Then by induction hypothesis on π', we have that $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$ implies $\alpha \Vdash \sigma(t : C_1 \sqcap C_2)$ in $\underline{K}$. By definition of the semantics, $\alpha \Vdash \sigma(t : C_1)$ and $\alpha \Vdash \sigma(t : C_2)$: thus, $\alpha \Vdash \sigma(t : C_k)$ in $\underline{K}$ for $k \in \{1, 2\}$.

Let $r = \sqcup I$. Then by induction hypothesis on π' we have that $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$ implies $\alpha \Vdash \sigma(t : C_k)$ in $\underline{K}$ with $k \in \{1, 2\}$. By definition, this implies that $\alpha \Vdash \sigma(t : C_1 \sqcup C_2)$ in $\underline{K}$.

Let $r = \sqcup E$. Hence $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$: by induction hypothesis on π_1, $\alpha \Vdash \sigma(t : C_1 \sqcup C_2)$ in $\underline{K}$. Let us assume that $\alpha \Vdash \sigma(t : C_1)$ in $\underline{K}$: then by induction hypothesis on π_2, $\alpha \Vdash \sigma K$ in $\underline{K}$. The case in which $\alpha \Vdash \sigma(t : C_2)$ in $\underline{K}$ is similar: thus, in both cases $\alpha \Vdash \sigma K$ in $\underline{K}$ and the assertion is proved.

Let $r = \Longrightarrow I$. Let β be any element of P such that $\alpha \leq \beta$. By monotonicity property, $\beta \Vdash \sigma\Gamma$ and, by induction hypothesis on π', if $\beta \Vdash \sigma t : C$ then $\beta \Vdash \sigma t : D$, hence the assertion.

Let $r = \Longrightarrow E$. Hence $\Gamma = \Gamma_1 \cup \Gamma_2$: by induction hypothesis, if $\alpha \Vdash \sigma\Gamma$, we get that $\alpha \Vdash \sigma(t : C \to D)$ and that $\alpha \Vdash \sigma(t : C)$ in $\underline{K}$. Thus, by semantics definition, $\alpha \Vdash \sigma(t : D)$ in $\underline{K}$.

Let $r = \exists I$. Then $\Gamma = \Gamma' \cup \{(t, s) : R\}$: $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$, by induction hypothesis, implies that $\alpha \Vdash \sigma((t, u) : R)$ and $\alpha \Vdash \sigma(u : A)$ in $\underline{K}$. Thus, by definition, it holds that $\alpha \Vdash \sigma(t : \exists R.A)$ in $\underline{K}$.

Let $r = \exists E$. Then $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \{(t, p) : R, p : C\}$ and if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$, by induction hypothesis on π_1, we get that $\alpha \Vdash \sigma(t : \exists R.C)$ in $\underline{K}$. This implies that there exists $d \in \mathcal{D}^\alpha$ such that $\alpha \Vdash \sigma((t, d) : R)$ and $\alpha \Vdash \sigma(d : C)$ in $\underline{K}$. Let us consider a new α-substitution $\sigma' = \sigma[d/p]$. Since p does not occur in Γ_2, $\alpha \Vdash \sigma'\Gamma_2$ in $\underline{K}$ and by induction hypothesis we get that $\alpha \Vdash \sigma' K$ in $\underline{K}$. Since p does not occur in K, this implies the assertion.

Let $r = \forall I$. Hence $\Gamma = \Gamma' \cup \{(t,p) : R\}$: if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$, then, by induction hypothesis on π', $\alpha \Vdash \sigma(p : C)$ in $\underline{K}$. Since p does not occur in Γ, this happens for every α-substitution σ' such that $\sigma'(p) = d$, where $d \in \mathcal{D}^\alpha$ and $\alpha \Vdash \sigma((t,d) : R)$ in $\underline{K}$. By the monotonicity property, this holds for every $\beta \in P$ such that $\alpha \leq \beta$: thus by definition $\alpha \Vdash \sigma(t : \forall R.C)$ and the assertion holds.

Let $r = \forall E$. Then $\Gamma = \Gamma' \cup \{(s,t) : R\}$: if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$, then by induction hypothesis on π' we have that $\alpha \Vdash \sigma(s : \forall R.A)$ and $\alpha \Vdash \sigma((s,t) : R)$ in $\underline{K}$. Thus, by semantics definition, $\alpha \Vdash \sigma(s : C)$ in $\underline{K}$.

Let $r = \bot I$. Hence $\Gamma = \Gamma_1 \cup \Gamma_2$: if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$, by induction hypothesis on π_1 and π_2, this would imply that $\alpha \Vdash \sigma(t : C)$ and $\alpha \Vdash \sigma(t : \neg A)$ in $\underline{K}$ which is an absurd. This implies that $\underline{K}, \alpha \Vdash \sigma\Gamma$ can not hold and this verifies the assertion.

Let $r = \bot E$. Then by induction hypothesis on π' we have that if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$ then $\alpha \Vdash \sigma(t : \bot)$ in $\underline{K}$. Since, for any $t \in \mathtt{NI} \cup \mathcal{D}^\alpha$, $\alpha \nVdash \sigma(t : \bot)$ in $\underline{K}$ we get that $\alpha \Vdash \sigma\Gamma$ can not hold in $\underline{K}$. Thus $\alpha \Vdash \sigma K$ in $\underline{K}$ for every formula K of $\mathcal{L}_\mathcal{N}$.

Let $r = \neg I$. Let $\beta \in P$ be any element such that $\alpha \leq \beta$. If $\beta \Vdash \sigma\Gamma$ then $\beta \nVdash \sigma(t : C)$ in $\underline{K}$, otherwise by induction hypothesis we would have $\beta \Vdash \sigma(t : \bot)$. Thus, $\beta \nVdash \sigma(t : C)$, which implies that, if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$, then for every $\beta \in P$ such that $\alpha \leq \beta$ it holds $\beta \nVdash \sigma(t : C)$. By definition of the semantics, it holds that $\alpha \Vdash \sigma(t : \neg C)$ in $\underline{K}$.

Let $r = \sqsubseteq E$. Hence $\Gamma = \Gamma' \cup \{C \sqsubseteq D\}$ and by induction hypothesis, if $\alpha \Vdash \sigma\Gamma$ in $\underline{K}$, then we have that $\alpha \Vdash C \sqsubseteq D$ and $\alpha \Vdash \sigma(t : C)$ in $\underline{K}$. Thus, by definition of the semantics, $\alpha \Vdash \sigma(t : D)$ in $\underline{K}$.

Let $r = \mathrm{KUR}$. Then by induction hypothesis, if $\alpha \Vdash \sigma\Gamma$ it holds that $\alpha \Vdash \sigma(t : \forall R.\neg\neg C)$. By definition of the semantics, and given that Kur is a $\mathcal{KALC}$-valid schema (e.g., see Example 3), we obtain that $\alpha \Vdash \sigma(t : \neg\neg\forall R.C)$. □

We obtain that $\mathcal{ND}_\mathcal{K}$ is also sound with respect to the Kripke semantics of $\mathcal{KALC}$:

Theorem 5.8 (Soundness)
$\Gamma \mid\!\!\frac{}{\mathcal{BCDL}_\mathcal{K}} K$ *implies* $\Gamma \overset{\mathrm{k}}{\models} K$. □

To conclude the discussion on the soundness of $\mathcal{ND}_\mathcal{K}$, we note that we can enrich our calculus with the following rule, that is provable to be sound with respect to the information terms semantics of $\mathcal{BCDL}_\mathcal{K}$:

$$\dfrac{\begin{array}{c}\Gamma \\ \vdots\ \pi' \\ t : \neg\neg A\end{array}}{t : A}\ \mathrm{At} \qquad \text{where } A \in \mathtt{NC}$$

While this rule is sound with respect to $\mathcal{BCDL}_\mathcal{K}$ and it is included in the calculus for $\mathcal{BCDL}$, in general it is not sound with respect to the Kripke semantics of $\mathcal{KALC}$.

5.3 Service specifications

We can now introduce the basic definitions for the description of systems and for the specification of services operating on them.

A *service specification (over $\mathcal{L}_\mathcal{N}$)* is an expression of the form

$$\mathsf{s}(x) :: P \Rightarrow Q$$

where:

- s is a label that identifies the service;
- x is the input parameter of the service (to be instantiated with an individual name from $\mathcal{N}$);
- P and Q are concepts over $\mathcal{L}_\mathcal{N}$.

P is called the service *pre-condition*, denoted with $\mathrm{Pre}(\mathsf{s})$, and Q the service *post-condition*, denoted with $\mathrm{Post}(\mathsf{s})$.

Given a service specification $\mathsf{s}(x) :: P \Rightarrow Q$ over $\mathcal{L}_\mathcal{N}$ we call *service implementation* a function

$$\Phi_s : \bigcup_{t \in \mathcal{N}} \mathrm{IT}_\mathcal{N}(t : P) \rightarrow \bigcup_{t \in \mathcal{N}} \mathrm{IT}_\mathcal{N}(t : Q)$$

We denote with the pair $(\mathsf{s}(x) :: P \Rightarrow Q, \Phi_s)$ (or simply with (s, Φ_s)) a *service definition* over $\mathcal{L}_\mathcal{N}$.

Essentially, a service definition corresponds to an effective Web service. The service specification provides the formal description of the behavior of the service in terms of pre- and post- conditions. The function Φ_s represents a formal description of the service implementation: that is, it represents the input/output function that the service provides.

The notion of correctness of implementations with respect to their service specifications is modeled as follows.

Definition 5.9 (Uniform solvability)
Given a language $\mathcal{L}_\mathcal{N}$, a service definition $(\mathsf{s}(x) :: P \Rightarrow Q, \Phi_s)$ over $\mathcal{L}_\mathcal{N}$ and a model $\mathcal{M}$ for $\mathcal{L}_\mathcal{N}$, Φ_s uniformly solves $\mathsf{s}(x) :: P \Rightarrow Q$ *in $\mathcal{M}$ iff, for every individual name $t \in \mathcal{N}$ and every $\alpha \in \mathrm{IT}_\mathcal{N}(t : P)$ such that $\mathcal{M} \blacktriangleright \langle\alpha\rangle\, t : P$, $\mathcal{M} \blacktriangleright \langle\Phi_s(\alpha)\rangle\, t : Q$.*

Example 6 (Service specification)
We can now model the services provided by the Producer and Shipper agents.

$$\begin{array}{l}\mathsf{DoProduceRequest}(\mathtt{req}) :: \\ \quad \mathtt{ProduceRequest} \sqcap \exists \mathtt{hasProduct}.\mathtt{Product} \\ \quad \Rightarrow \mathtt{RefusedRequest} \sqcup (\, \mathtt{AcceptedProduceRequest} \sqcap \\ \qquad \exists \mathtt{hasOffer}.(\, \mathtt{ProduceOffer} \sqcap \exists \mathtt{hasCost}.\mathtt{Price}\,)\,)\end{array}$$

$$\begin{aligned}&\mathsf{DoShippingRequest}(\mathtt{req}) ::\\ &\quad \mathtt{ShippingRequest} \sqcap \exists \mathtt{hasDestination}.\mathtt{Location}\\ &\quad \Rightarrow \mathtt{RefusedRequest} \sqcup (\,\mathtt{AcceptedShippingRequest} \sqcap\\ &\qquad \exists \mathtt{hasOffer}.(\,\mathtt{ShippingOffer} \sqcap \exists \mathtt{hasCost}.\mathtt{Price}\,)\,)\end{aligned}$$

The service described by DoProduceRequest takes as input a request req specifying the required product and must classify it according to the service post-condition: namely, the service can answer with a refusal to the request (by classifying req in RefusedRequest) or it can accept the request and produce an offer with a price specified by the hasCost role. The DoShippingRequest service works in a similar way: it takes as input the destination where to ship the product and either refuses the request or it accepts the request providing an offer with the associated price.

In our setting, service implementations correspond to functions mapping information terms for the pre-condition into information terms for the post-condition. These functions formalize the behavior of the effective implementation of the Web services. In particular, consider the implementation Φ_{DPR} of the DoProduceRequest service. Let req_1 be the individual name representing a request. The input of Φ_{DPR} is any information term

$$\alpha \in \text{IT}_{\mathcal{N}}(\mathtt{req\_1} : \text{Pre}(\mathsf{DoProduceRequest}))$$

req_1 can be seen as a reference to a database record providing the information required by the service precondition and α can be seen as a structured representation of such information. Let us suppose that

$$\alpha = (\mathtt{tt}, (\mathtt{book\_1}, \mathtt{tt}))$$

This information term means that req_1 is a product request with associated product book_1. Now, let

$$\beta = \Phi_{\text{DPR}}(\alpha) \in \text{IT}(\mathtt{req\_1} : \text{Post}(\mathsf{DoProduceRequest}))$$

If $\beta = (1, \mathtt{tt})$, this classify req_1 as refused. Otherwise β could be

$$(2, (\mathtt{tt}, (\mathtt{off\_1}, (\mathtt{tt}, (\mathtt{price\_1}, \mathtt{tt})))))$$

which classifies req_1 as accepted and specifies that there is an offer off_1 with associated price price_1 for the requested product. The implementation Φ_{DSR} of DoShippingRequest acts in a similar way.

To conclude, we remark that the intended model $\mathcal{M}$ we use to evaluate the correctness of the system is implicitly defined by the knowledge base of the system. Indeed,

$$\mathcal{M} \blacktriangleright \langle \alpha \rangle\, \mathtt{req\_1} : \text{Pre}(\mathsf{DoProduceRequest})$$

if and only if in our system req_1 effectively codify a request and book_1 is classified as a product. In this case, since Φ_{DPR} uniformly solves the service specification, we know that

$$\mathcal{M} \blacktriangleright \langle \beta \rangle\, \mathtt{req\_1} : \text{Post}(\mathsf{DoProduceRequest})$$

This trivially corresponds to the fact that, looking at its knowledge base, the Producer can generate its offer. ◇

The problem of service composition amounts to build a new service from a family of implemented services. We formalize this problem in the context of an *environment*, that is a structure

$$\mathbf{E} = \langle \mathcal{L}_{\mathcal{N}}, \mathbf{T}, \eta, \mathcal{S}_1, \ldots, \mathcal{S}_n \rangle$$

where:

– $\mathbf{T}$ is a theory over the language $\mathcal{L}_{\mathcal{N}}$;

– $\eta \in \text{IT}_{\mathcal{N}}(\mathbf{T})$;

– for every $i \in \{1, \ldots, n\}$, $\mathcal{S}_i = (s_i, \Phi_i)$ is a service definition in $\mathcal{L}_{\mathcal{N}}$.

Given a model $\mathcal{M}$ for $\mathcal{L}_{\mathcal{N}}$ we say that $\mathcal{M}$ is a model for $\mathbf{E}$ iff $\mathcal{M} \blacktriangleright \langle \eta \rangle \mathbf{T}$ and for every $i \in \{1, \ldots, n\}$, Φ_i uniformly solves s_i in $\mathcal{M}$. A service specification s' is *solvable* in $\mathbf{E}$ if there exists an implementation Φ' of s' such that, for every model $\mathcal{M}$ of $\mathcal{L}_{\mathcal{N}}$, if $\mathcal{M}$ is a model for $\mathbf{E}$ then Φ' uniformly solves s' in $\mathcal{M}$.

Example 7 (Composition problem definition)
Given the previous specifications, we are now ready to state the composition problem. We want to combine the services DoProduceRequest and DoShippingRequest to provide the User with a single service to request both the production and the delivery of an object. To do this, we define a third service that composes the offers from the two agents:

ProcessOffers(`req`) ::
$\quad$ `AcceptedProduceRequest` $\sqcap$ $\exists$`hasOffer.(` `ProduceOffer` $\sqcap$ $\exists$`hasCost.Price` `)` $\sqcap$
$\quad$ `AcceptedShippingRequest` $\sqcap$ $\exists$`hasOffer.(` `ShippingOffer` $\sqcap$ $\exists$`hasCost.Price` `)`
$\quad$ $\Rightarrow$ `AcceptedRequest` $\sqcap$ $\exists$`hasOffer.(` `Offer` $\sqcap$ $\exists$`hasCost.Price` `)`

Let Φ_{PO} be the implementation of ProcessOffers. We define the environment

$$\mathbf{E}_{PS} = \langle \mathcal{L}_{\mathcal{N}}, \mathsf{T}_{PS}, \eta, S_1, S_2, S_3 \rangle$$

where:

$$S_1 = (\mathsf{DoProduceRequest}, \Phi_{\text{DPR}})$$
$$S_2 = (\mathsf{DoShippingRequest}, \Phi_{\text{DSR}})$$
$$S_3 = (\mathsf{ProcessOffers}, \Phi_{\text{PO}})$$

The problem can be now reduced to the definition of a suitable service specification that is solvable in such environment. ◇

Now, the main point of service composition is to effectively build the implementation of the service specification starting from the environment. This problem can be solved in two ways: the first solution consists in the definition of a composition

language which allows the user to build up a new service starting from the environment. The second is given by providing a method to automatically build up the new service implementation.

The formalization of the composition problem in the framework of a constructive logic allows to use the proof-theoretical properties of the logical system to support the composition problem. In the following we only concentrate on the definition of a composition language. As for the problem of automatic service composition, it can be seen as a reformulation of the *program-synthesis* problem, that is a problem which has a long tradition in the constructive logics context. The program-synthesis problem can be related to the idea of information extraction from proofs which has already been studied in the framework of $\mathcal{BCDL}$ in [Bozzato *et al.*(2007), Ferrari *et al.*(2010a)].

5.4 Composition calculus

The composition calculus we describe in this section is mainly inspired by *PAP* [Miglioli *et al.*(1991)], a calculus which support program synthesis from proofs of a constructive logical system. Our calculus allows to manually compose services guaranteeing the correctness of the composed service. The main advantage of our formalization is that service composition can be supported by an appropriate proof-system. This tool can be used to check the correctness of rule applications and to automatically build the proofs of the applicability conditions.

A *composition* over an environment $\mathbf{E} = \langle \mathcal{L}_{\mathcal{N}}, \mathbf{T}, \eta, \mathcal{S}_1, \ldots, \mathcal{S}_n \rangle$ is defined as:

$$\frac{\mathsf{s}(x) :: P \Rightarrow Q}{\begin{array}{c}\Pi_1 : \mathsf{s_1}(x) :: P_1 \Rightarrow Q_1 \\ \cdots \\ \Pi_n : \mathsf{s_n}(x) :: P_n \Rightarrow Q_n\end{array}} r$$

where:

- $\mathsf{s}(x) :: P \Rightarrow Q$ is a service specification over $\mathbf{E}$;
- r is one of the rules of the composition calculus $\mathcal{SC}$;
- For every $i \in \{1, \ldots, n\}$, $\Pi_i : \mathsf{s_i}(x) :: P_i \Rightarrow Q_i$ is a service composition over $\mathbf{E}$ that meets the applicability conditions of r.

The rules of the composition calculus $\mathcal{SC}$ and their *computational interpretation* (CI) are given in Figure 5.3. In the rules, the service specification $\mathsf{s}(x) :: P \Rightarrow Q$ is called the *main sequent* of the rule and represents the specification of the service to be composed. The service specifications $\mathsf{s_i}(x) :: P_i \Rightarrow Q_i$ are called *subsequents* of the rule and represent the services involved in the composition. The sequents must satisfy the *applicability conditions* (AC) of the rule. These conditions describe the role of the subsequents in the composition of the main sequent: in order to verify

$$\frac{\mathsf{s}(x) :: A \Rightarrow B}{\mathsf{s_1}(x) :: A_1 \Rightarrow B_1 \quad \cdots \quad \mathsf{s_n}(x) :: A_n \Rightarrow B_n}\,AND$$

AC $\begin{cases} (a_k) & \mathbf{T}, x : A \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : A_k,\ k = 1, \ldots, n \\ (b) & \mathbf{T}, x : B_1 \sqcap \ldots \sqcap B_n \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : B \end{cases}$

CI $\Phi_s(\alpha) = \Phi_b(\Phi_{s_1}(\Phi_{a_1}(\alpha)), \ldots, \Phi_{s_n}(\Phi_{a_n}(\alpha)))$

$$\frac{\mathsf{s}(x) :: A \Rightarrow B}{\mathsf{s_1}(x) :: A_1 \Rightarrow B_1 \quad \cdots \quad \mathsf{s_n}(x) :: A_n \Rightarrow B_n}\,CASE$$

AC $\begin{cases} (a) & \mathbf{T}, x : A \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : A_1 \sqcup \ldots \sqcup A_n \\ (b_k) & \mathbf{T}, x : B_k \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : B,\ k = 1, \ldots, n \end{cases}$

CI $\Phi_s(\alpha) = \Phi_{b_k}(\Phi_{s_k}(\alpha_k))$ with $(k, \alpha_k) = \Phi_a(\alpha)$

$$\frac{\mathsf{s}(x) :: A \Rightarrow B}{\mathsf{s_1}(x) :: A_1 \Rightarrow B_1 \quad \cdots \quad \mathsf{s_n}(x) :: A_n \Rightarrow B_n}\,SEQ$$

AC $\begin{cases} (b_1) & \mathbf{T}, x : A \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : A_1 \\ (b_k) & \mathbf{T}, x : B_{k-1} \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : A_k,\ k = 2, \ldots, n \\ (c) & \mathbf{T}, x : B_n \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : B \end{cases}$

CI $\Phi_s(\alpha) = \Phi_c(\,\Phi_{s_n} \cdot \Phi_{b_n} \cdot \ldots \cdot \Phi_{s_1} \cdot \Phi_{b_1}(\alpha)\,)$

$\mathsf{s}(x) :: A \Rightarrow B$ AX

AC $(a)\ \mathbf{T}, x : A \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : B$ CI $\Phi_s(\alpha) = \Phi_a(\alpha)$

$\mathsf{s}(x) :: A \Rightarrow B$ ENV

with (s, Φ_s) a service defined in $\mathbf{E}$

Figure 5.3: The rules of calculus $\mathcal{SC}$

the correctness of compositions, the proof checker must verify the truth of such conditions.

The composition rules have both a logical and a computational reading. The computational interpretation (CI) of rules allow to associate with a service composition its service implementation, defined as follows: given a service composition Π with main sequent $\mathsf{s}(x) :: P \Rightarrow Q$, we define the function

$$\Phi_\mathsf{s} : \bigcup_{t \in \mathcal{N}} \mathrm{IT}_\mathcal{N}(t : P) \to \bigcup_{t \in \mathcal{N}} \mathrm{IT}_\mathcal{N}(t : Q)$$

associated with s. The function is inductively defined on the last rule r applied in Π. Here we assume the following conventions: given a subsequent s' of the rule r, we denote with $\Phi_{s'}$ its computed function; given the applicability condition $(a)\ \Gamma \vdash_{\overline{\mathcal{BCDL}_\mathcal{K}}} x : A$ of the rule r we denote with Φ_a the operator corresponding to the proof $\pi :: \Gamma \vdash x : A$ defined according to Section 5.2.

Inspecting the rules of Figure 5.3 we can understand them as logical rules and control structures as follows:

– The AND rule represents a $\sqcap$ introduction on the right hand side of the specification sequents: the services composed by this rule are seen as a parallel execution of the sub services.

– The CASE rule represents a $\sqcup$ elimination on the left hand side of the specification sequent: the services composed by this rule are seen as in a case construct, in which the applicability condition determines the executed sub-service.

– The SEQ rule represents a composition given as a sequential execution of the sub-services and a composition of proofs under the logical reading.

– The AX rule states that the system can infer specifications provable under a suitable calculus for $\mathcal{BCDL}_{\mathcal{K}}$.

– The ENV rule allows to use the specifications given in the environment **E**.

Let us complete our example with a sample service composition.

Example 8 (Service Composition)
Given the environment $\mathbf{E}_{PS}$ defined in Example 7 and the rules of $\mathcal{SC}$, we can define a new service ProduceAndShip as the composition Π of the stated specifications: the composition is presented in Figure 5.4. The behavior of this service is defined as follows: using the DoRequest service (service composition Π_1), it first invokes the DoProduceRequest and the DoShippingRequest services to query the Producer and the Shipper over the combined request `req`. The answer of the two is then combined by PresentOffer (service composition Π_2): by a case construct, this sub-service either responds that the request `req` has been classified as refused, or it accepts the request and generate the combined price using the ProcessOffers service.

Now, let us discuss how the composite service computes information terms by explaining a sample execution. Let `req_2` be both a `ProduceRequest` and a `ShippingRequest` with associated product `book_1` and destination `my_home`. Then, a call of ProduceAndShip over `req_2` has as input information term

$$\alpha_1 = (\mathtt{tt}, (\mathtt{tt}, ((\mathtt{book\_1}, \mathtt{tt}), (\mathtt{my\_home}, \mathtt{tt}))))$$

Following the composition, the execution of ProduceAndShip starts with the sequence construct and the first invoked service is DoRequest which process the information term α_1. DoRequest consists of a parallel call to the services of the Producer and the Shipper. The first request is executed as a call to DoProduceRequest. According to the conditions of the AND rule, we have a proof:

$$\pi_1 :: \mathsf{T}_{PS}, x : \mathrm{Pre}(\mathsf{DoRequest}) \vdash_{\mathcal{BCDL}_{\mathcal{K}}} x : \mathrm{Pre}(\mathsf{DoProduceRequest})$$

The corresponding operator $\Phi^{\pi_1}_{\mathcal{N}}$ allows us to extract from α_1 the information term:

$$(\mathtt{tt}, (\mathtt{book\_1}, \mathtt{tt})) \in \mathrm{IT}_{\mathcal{N}}(\mathtt{req\_2} : \mathrm{Pre}(\mathsf{DoProduceRequest}))$$

Let us suppose that the Producer accepts the request and produces an offer `p_off` with an associated price. The offer is codified in the information term:

$$\alpha_2 = \Phi_{DPR}((\mathtt{tt}, (\mathtt{book\_1}, \mathtt{tt}))$$

ProduceAndShip(req) ::
ProduceRequest ⊓ ShippingRequest⊓
∃hasProduct.Product ⊓ ∃hasDestination.Location ⇒
RefusedRequest ⊔ (AcceptedRequest ⊓ ∃hasOffer.(Offer ⊓ ∃hasCost.Price))
——— SEQ
Π_1 : DoRequest(req) ::
ProduceRequest ⊓ ShippingRequest⊓
∃hasProduct.Product ⊓ ∃hasDestination.Location ⇒
RefusedRequest ⊔ ((AcceptedProduceRequest ⊓
∃hasOffer.(ProduceOffer ⊓ ∃hasCost.Price))
⊓ (AcceptedShippingRequest ⊓
∃hasOffer.(ShippingOffer ⊓ ∃hasCost.Price)))
——— AND
DoProduceRequest(req) :: ENV
ProduceRequest ⊓ ∃hasProduct.Product ⇒
RefusedRequest ⊔ (AcceptedProduceRequest ⊓
∃hasOffer.(ProduceOffer ⊓ ∃hasCost.Price))
DoShippingRequest(req) :: ENV
ShippingRequest ⊓ ∃hasDestination.Location ⇒
RefusedRequest ⊔ (AcceptedShippingRequest ⊓
∃hasOffer.(ShippingOffer ⊓ ∃hasCost.Price))
Π_2 : PresentOffer(req) ::
RefusedRequest ⊔ ((AcceptedProduceRequest ⊓
∃hasOffer.(ProduceOffer ⊓ ∃hasCost.Price))
⊓ (AcceptedShippingRequest ⊓
∃hasOffer.(ShippingOffer ⊓ ∃hasCost.Price))) ⇒
RefusedRequest ⊔ (AcceptedRequest ⊓
∃hasOffer.(Offer ⊓ ∃hasCost.Price))
——— CASE
RefuseRequest(req) :: AX
RefusedRequest ⇒ RefusedRequest
ProcessOffers(req) :: ENV
AcceptedProduceRequest ⊓
∃hasOffer.(ProduceOffer ⊓ ∃hasCost.Price) ⊓
AcceptedShippingRequest ⊓
∃hasOffer.(ShippingOffer ⊓ ∃hasCost.Price) ⇒
AcceptedRequest ⊓ ∃hasOffer.(Offer ⊓ ∃hasCost.Price)

Figure 5.4: Composition of the ProduceAndShip service

Let us assume that α_2 has the following form:

$$\alpha_2 = (2, (\texttt{tt}, (\texttt{p\_off}, (\texttt{tt}, (\texttt{p\_off\_price}, \texttt{tt})))))$$

The request to the Shipper consists in a call to DoShippingRequest with input information term

$$(\texttt{tt}, (\texttt{my\_home}, \texttt{tt}))$$

Also in this case this information term is generated from the operator associated with an applicability rule. As above, if the Shipper accepts the request with an offer `s_off` and its price, then the output information term is:

$$\alpha_3 = (2, (\texttt{tt}, (\texttt{s\_off}, (\texttt{tt}, (\texttt{s\_off\_price}, \texttt{tt})))))$$

Now the applicability conditions of the AND composition rule, in particular the proof:

$$\begin{array}{rl} \pi_2 :: & \mathsf{T}_{PS}, x : \mathrm{Post}(\mathsf{DoProduceRequest}) \sqcap \mathrm{Post}(\mathsf{DoShippingRequest}) \\ & \vdash_{\overline{\mathcal{BCDL}_{\mathcal{K}}}} x : \mathrm{Post}(\mathsf{DoRequest}) \end{array}$$

allows us to combine α_2 and α_3 to get an $\alpha_4 \in \mathrm{IT}_{\mathcal{N}}(\texttt{req\_2} : \mathrm{Post}(\mathsf{DoRequest}))$ as follows:

$$\begin{array}{l} \alpha_4 = (2, ((\texttt{tt}, (\texttt{p\_off}, (\texttt{tt}, (\texttt{p\_off\_price}, \texttt{tt})))), \\ \qquad\quad (\texttt{tt}, (\texttt{s\_off}, (\texttt{tt}, (\texttt{s\_off\_price}, \texttt{tt})))))) \\ \qquad) \end{array}$$

Proceeding in the sequence, the previous responses are combined by means of a call to PresentOffer with input information term α_4. By the AC of the CASE construct, as the request has been accepted by both agents, we enter in the second of the cases and we call ProcessOffers with input information term:

$$\begin{array}{l} \alpha_5 = (\ (\texttt{tt}, (\texttt{p\_off}, (\texttt{tt}, (\texttt{p\_off\_price}, \texttt{tt})))), \\ \qquad (\texttt{tt}, (\texttt{s\_off}, (\texttt{tt}, (\texttt{s\_off\_price}, \texttt{tt}))))\ ) \end{array}$$

The service combines the offers producing a composite offer `ps_off` with its associated price `ps_off_price` modeled by the information term:

$$(\texttt{tt}, (\texttt{ps\_off}, (\texttt{tt}, (\texttt{ps\_off\_price}, \texttt{tt}))))$$

Finally the output of PresentOffer and ProduceAndShip is:

$$(2, (\texttt{tt}, (\texttt{ps\_off}, (\texttt{tt}, (\texttt{ps\_off\_price}, \texttt{tt})))))$$

This object states that the request has been accepted and it contains both the object representing the composite offer (`ps_off`) and its composite price (`ps_off_price`). $\diamond$

To conclude this section we state the result asserting the soundness of the rules with respect to uniform solvability. Note that the proof of this theorem relies on the generalized semantics and rules presented at the end of Section 5.2.1.

Theorem 5.10
Let $\mathbf{E} = \langle \mathcal{L}_{\mathcal{N}}, \mathbf{T}, \eta, \mathcal{S}_1, \ldots, \mathcal{S}_n \rangle$ *be an environment and let* $\mathsf{s}(x) :: P \Rightarrow Q$ *be the main sequent of a composition* Π *over* $\mathbf{E}$. *For every model* $\mathcal{M}$ *for* $\mathcal{L}$, *if* $\mathcal{M}$ *is a model for* $\mathbf{E}$ *then the function* Φ_s *extracted from* Π *uniformly solves* s *in* $\mathcal{M}$.

Proof: Let $\mathcal{M}$ be a model for $\mathbf{E}$ and let $t \in \mathcal{N}$ and $\alpha \in \mathrm{IT}_{\mathcal{N}}(t : P)$ such that $\mathcal{M} \blacktriangleright \langle \alpha \rangle\, t : P$. We prove the assertion by induction on the last rule r applied in the composition Π.

If $r = \mathrm{AX}$, the composition Π only consists of an axiom introduction. By the applicability condition (a) and Theorem 5.5, we have that $\mathcal{M} \blacktriangleright \langle \Phi_a(\alpha) \rangle\, t : B$: since by definition $\Phi_a(\alpha) = \Phi_s(\alpha)$, this implies that Φ_s uniformly solves s in $\mathcal{M}$.

If $r = \mathrm{ENV}$, the assertion directly holds since, by definition, the service (s, Φ_s) is defined in $\mathbf{E}$. Since $\mathcal{M}$ is a model for $\mathbf{E}$, Φ_s uniformly solves s in $\mathcal{M}$.

If $r = \mathrm{AND}$, by the applicability conditions (a_k) for $k \in \{1, \ldots, n\}$ and Theorem 5.5, since $\mathcal{M} \blacktriangleright \langle \alpha \rangle\, t : A$, then

$$\mathcal{M} \blacktriangleright \langle \Phi_{a_k}(\alpha) \rangle\, t : A_k$$

Let $\beta_k = \Phi_{a_k}(\alpha)$. Since $\mathcal{M}$ is a model for $\mathbf{E}$, this implies that

$$\mathcal{M} \blacktriangleright \langle \Phi_{s_k}(\beta_k) \rangle\, t : B_k$$

for every $k \in \{1, \ldots, n\}$. Now, let $\gamma_k = \Phi_{s_k}(\beta_k)$. By semantics definition it holds that

$$\mathcal{M} \blacktriangleright \langle (\gamma_1, \ldots, \gamma_n) \rangle\, t : B_1 \sqcap \ldots \sqcap B_n$$

By (b) and Theorem 5.5, we have

$$\mathcal{M} \blacktriangleright \langle \Phi_b((\gamma_1, \ldots, \gamma_n)) \rangle\, t : B$$

and the assertion is proved.

If $r = \mathrm{CASE}$, by the applicability condition (a) and Theorem 5.5, we have that

$$\mathcal{M} \blacktriangleright \langle \Phi_a(\alpha) \rangle\, t : A_1 \sqcup \ldots \sqcup A_n$$

Let $\Phi_a(\alpha) = (k, \alpha_k)$, with $k \in \{1, \ldots, n\}$. By semantics definition, this implies that:

$$\mathcal{M} \blacktriangleright \langle \alpha_k \rangle\, t : A_k$$

Thus, since $\mathcal{M}$ is a model for $\mathbf{E}$, it holds that

$$\mathcal{M} \blacktriangleright \langle \Phi_{s_k}(\alpha_k) \rangle\, t : B_k$$

Let $\gamma_k = \Phi_{s_k}(\alpha_k)$, then by (b_k) and Theorem 5.5,

$$\mathcal{M} \blacktriangleright \langle \Phi_{b_k}(\gamma_k) \rangle\, t : B$$

The assertion is proved by the definition of Φ_s.

If $r = \text{SEQ}$, by the applicability condition (b_1) and Theorem 5.5

$$\mathcal{M} \blacktriangleright \langle \Phi_{b_1}(\alpha) \rangle \, t : A_1$$

Let $\beta_1 = \Phi_{b_1}(\alpha)$. Since $\mathcal{M}$ is a model for $\mathbf{E}$, we have that

$$\mathcal{M} \blacktriangleright \langle \Phi_{s_1}(\beta_1) \rangle \, t : B_1$$

Let $\gamma_1 = \Phi_{s_1}(\beta_1)$. By the application rules (b_k) and Theorem 5.5, for every $k \in \{2, \ldots, n\}$ we have

$$\mathcal{M} \blacktriangleright \langle \gamma_{k-1} \rangle \, t : B_{k-1} \text{ implies } \mathcal{M} \blacktriangleright \langle \Phi_{b_k}(\gamma_{k-1}) \rangle \, t : A_k$$

Moreover, given that $\mathcal{M}$ is a model for $\mathbf{E}$, for every $k \in \{2, \ldots, n\}$ we have

$$\mathcal{M} \blacktriangleright \langle \beta_k \rangle \, t : A_k \text{ implies } \mathcal{M} \blacktriangleright \langle \Phi_{s_k}(\beta_k) \rangle \, t : B_k$$

with $\beta_k = \Phi_{b_k}(\gamma_{k-1})$. We obtain that, since $\mathcal{M} \blacktriangleright \langle \beta_1 \rangle \, t : A_1$ then

$$\mathcal{M} \blacktriangleright \langle \gamma_n \rangle \, t : B_n$$

Finally, by (c) and Theorem 5.5, it holds

$$\mathcal{M} \blacktriangleright \langle \Phi_c(\gamma_n) \rangle \, t : B$$

and, by the definition of Φ_s, the assertion is verified. □

To conclude the chapter, we discuss some of the features of our approach with respect to the known proposals for Semantic Web services composition and we give some final notes on the limits and possible extensions of our calculus.

Our proposal can be classified together with the approaches which base the composition on a pure atomic view of the services, as provided by the *service profile* of OWL-S [Martin *et al.*(2007)]. An approach based on services profiles, which also highlights the relationships between compositions and software synthesis, is presented in [Matskin and Rao(2002)]. The proposal uses the *Structural Synthesis Program (SSP)* method [Matskin and Tyugu(2001)] to extract compositions. We notice that SSP is based on the implicative part of the intuitionistic propositional calculus and it can only define sequential or conditional compositions in general. Another kind of composition depending on the formulas of service profiles can be found in the related approaches for action formalisms and planning over description logics as in [Baader *et al.*(2005), Calvanese *et al.*(2007a), Drescher and Thielscher(2007), Miličić(2007)]. However, even classical planning techniques mostly generate sequential compositions achieving a goal.

On the other hand, other approaches concentrate on the *process model* of the services, that is the representation of the flow of interactions that composes the services. Among these proposals we can include approaches based on different formalizations for the process model of OWL-S service descriptions, such as translations to Petri Nets [Narayanan and McIlraith(2002)] and to state transition systems

[Traverso and Pistore(2004)]. We notice, however, that many of the latter proposals do not base their compositions on a logic representation of pre- and post- conditions or on the function mapping inputs to outputs. Moreover, whenever a composition of sub services is defined, this does not explicitly depend on the condition stated in a declarative description of the composite service.

To compare our approach with the ones cited above, we remark that $\mathcal{SC}$ assures that the composite service specification directly follows from composition proof. The correctness of compositions can be checked directly by verifying the applicability conditions of the rules used in the composition: moreover, our rules directly represent common control structures, thus allowing to represent complex compositions. We remark that, even if we simply detailed manual composition of services, by implementing $\mathcal{SC}$ we would obtain a method for automatic composition. For an actual implementation of our calculus we mainly need an implementation of the information terms semantics of $\mathcal{BCDL}_{\mathcal{K}}$ and its calculus, which can be obtained by the study of its relations with $\mathcal{KALC}$ and $\mathcal{BCDL}$.

Note that, in accord with the original formulation of PAP [Miglioli *et al.*(1991)], $\mathcal{SC}$ can be partly understood as a goal directed sequent calculus: however, $\mathcal{SC}$ is not directly implementable as such since the proofs of the applicability conditions are given in $\mathcal{ND}_{\mathcal{K}}$, a forward and not goal oriented natural deduction calculus. Moreover, the services definable with $\mathcal{SC}$ are more limited than the programs synthesized in the original PAP: indeed, there is no way to model complex data structures as *ADTs* (which however seem to have little interest in the context of Web services) and the proposed rules can not model induction and recursion. In the case of induction, while in PAP different induction schemas can be defined on the base of the properties of the modeled data (e.g. the usual induction schema on natural numbers), in $\mathcal{SC}$ we can consider to introduce induction by weaker induction principles, as schemas based on descending chains. However, compared with the first order formalization of PAP, it is not clear how to formalize a suitable induction schema given the lack of free variables in services specifications.

6

Conclusions

Motivated by the need of computable methods for reasoning in constructive description logics, in this work we presented the decidable logic $\mathcal{KALC}$. We introduced its Kripke-style semantics and studied its properties also with respect to the classical semantics of $\mathcal{ALC}$. We then proved its decidability by formulating the sound, complete and terminating tableau calculus $\mathcal{T}_{\mathcal{K}}$. The proof of its termination give rise to an effective tableau procedure for the decision of the constructive reformulation of usual inference problems for description logics.

In the following, we studied the relations between $\mathcal{KALC}$ and other significant logics: in particular, we described the relations with $\mathcal{KALC}^{\infty}$, a variant of $\mathcal{KALC}$ admitting infinite posets, first-order intuitionistic logic **QInt** and the multimodal intuitionistic logic $\mathbf{FS}_n$.

Finally, we proposed an application for an information terms semantics for $\mathcal{KALC}$ in the context of Semantic Web services composition. By formulating our proposal for a composition calculus, we also introduced the natural deduction calculus $\mathcal{ND}_{\mathcal{K}}$: we proved that this calculus is sound with respect to $\mathcal{KALC}$ and its information terms counterpart $\mathcal{BCDL}_{\mathcal{K}}$ and we explained how to obtain a computational interpretation of its proofs.

To conclude our work, we discuss some of the possible directions in which our results can be extended and completed. First of all, we aim at completing the study of the relations between $\mathcal{KALC}$ and our previous proposal $\mathcal{KALC}^{\infty}$: in particular, we conjecture that $\mathcal{KALC} \subseteq \mathcal{KALC}^{\infty}$. As pointed out in Chapter 4, if this is true we can conclude that $\mathcal{KALC} = \mathcal{KALC}^{\infty}$: this would imply that this logic meets the finite model property, meaning that also $\mathcal{KALC}^{\infty}$ is decidable.

Another needed investigation concerns the complexity of the proposed decision procedure. In particular we conjecture that, in the case of empty TBoxes, the complexity of the tableau procedure has to be in PSPACE and thus comparable to the complexity of classical tableau procedures for $\mathcal{ALC}$. To prove such inclusion we need to show that, for every branch of a proof tree, the number of rules applications is polynomial in the size of the initial set of formulas. The

question is to find a suitable upper bound for the number of individuals generated in each branch such to provide a limit to the applications of rules for dup-formulas. As noted in the related discussion in Chapter 3, another way to obtain a PSPACE tableau procedure could be to adopt a suitable variant of the trace technique [Schmidt-Schauß and Smolka(1991)] and prove the completeness of such strategy.

After such studies on complexity, another possible direction regards the development of implementations for the tableau calculus $\mathcal{T}_{\mathcal{K}}$ and the composition calculus $\mathcal{SC}$. With respect to our tableaux calculus, we can also investigate its possible extension with optimization techniques typically proposed in the context of tableaux calculi for intuitionistic and constructive logics [Ferrari *et al.*(2009), Ferrari *et al.*(2010b)].

We also plan to further investigate the relations between $\mathcal{KALC}$ and $\mathcal{BCDL}$, our constructive description logic based on information terms semantics. By this study we could extend our decidable tableau procedure also to the alternative constructive semantics of $\mathcal{BCDL}$. A possible starting point in this direction consists in the expansion and study of the natural deduction calculus $\mathcal{ND}_{\mathcal{K}}$ proposed in Chapter 5 for $\mathcal{BCDL}_{\mathcal{K}}$.

Finally, we aim to extend the decision procedure to treat general TBoxes and more complex role definitions as transitive and inverse role relations. In order to achieve this, we should adopt loop-checking mechanisms such as blocking and its variants [Horrocks *et al.*(2000)], commonly used in classical description logics tableau algorithms.

Bibliography

[Almukdad and Nelson(1984)] Almukdad, A. and Nelson, D. (1984). Constructible falsity and inexact predicates. *J. of Symbolic Logic*, **49**(1), 231–233.

[Avellone *et al.*(1999)] Avellone, A., Ferrari, M., and Miglioli, P. (1999). Duplication-free tableau calculi and related cut-free sequent calculi for the interpolable propositional intermediate logics. *Logic J. of the IGPL*, **7**(4), 447–480.

[Baader and Nutt(2003)] Baader, F. and Nutt, W. (2003). Basic description logics. In [Baader *et al.*(2003a)], pages 43–95.

[Baader *et al.*(2003a)] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003a). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[Baader *et al.*(2003b)] Baader, F., Küsters, R., and Wolter, F. (2003b). Extensions to description logics. In [Baader *et al.*(2003a)], pages 219–261.

[Baader *et al.*(2005)] Baader, F., Lutz, C., Miličić, M., Sattler, U., and Wolter, F. (2005). A description logic based approach to reasoning about web services. In *WSS2005 - Web Service Semantics Workshop (WWW2005)*.

[Bozzato and Ferrari(2010a)] Bozzato, L. and Ferrari, M. (2010a). A Note on Semantic Web Services Specification and Composition in Constructive Description Logics. *Computing Research Repository*, **cs.AI/1007.2364**.

[Bozzato and Ferrari(2010b)] Bozzato, L. and Ferrari, M. (2010b). Composition of semantic web services in a constructive description logic. In *RR2010 - Web Reasoning and Rule Systems*, volume 6333 of *Lecture Notes in Computer Science*, pages 223–226. Springer.

[Bozzato *et al.*(2007)] Bozzato, L., Ferrari, M., Fiorentini, C., and Fiorino, G. (2007). A constructive semantics for $\mathcal{ALC}$. In [Calvanese *et al.*(2007b)], pages 219–226.

[Bozzato *et al.*(2009a)] Bozzato, L., Ferrari, M., and Villa, P. (2009a). Actions over a constructive semantics for description logics. *Fundamenta Informaticae*, **96**(3), 253–269.

[Bozzato *et al.*(2009b)] Bozzato, L., Ferrari, M., and Villa, P. (2009b). A note on constructive semantics for description logics. In M. Gavanelli and F. Riguzzi, editors, *CILC09 - Convegno Italiano di Logica Computazionale*.

[Bozzato *et al.*(2010)] Bozzato, L., Ferrari, M., Fiorentini, C., and Fiorino, G. (2010). A decidable constructive description logic. In *JELIA 2010 - Logics in Artificial Intelligence*, volume 6341 of *Lecture Notes in Computer Science*, pages 51–63. Springer.

[Brachman(1979)] Brachman, R. (1979). On the epistemological status of semantic networks. In N. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press.

[Brachman and Levesque(1984)] Brachman, R. and Levesque, H. (1984). The tractability of subsumption in frame-based description languages. In R. Brachman, editor, *AAAI-84 - National Conference on Artificial Intelligence*, pages 34–37. AAAI Press.

[Calvanese *et al.*(2007a)] Calvanese, D., Giacomo, G. D., Lenzerini, M., and Rosati, R. (2007a). Actions and programs over description logic ontologies. In [Calvanese *et al.*(2007b)], pages 29–40.

[Calvanese *et al.*(2007b)] Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Tessaris, S., and Turhan, A., editors (2007b). *DL2007 - 20th International Workshop on Description Logics*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Chagrov and Zakharyaschev(1997)] Chagrov, A. and Zakharyaschev, M. (1997). *Modal Logic*. Oxford University Press.

[de Paiva(2005)] de Paiva, V. (2005). Constructive description logics: what, why and how. Technical report, Xerox Parc. Presented at *Context Representation and Reasoning 2006*.

[Drescher and Thielscher(2007)] Drescher, C. and Thielscher, M. (2007). Integrating action calculi and description logics. In *KI 2007: Advances in Artificial Intelligence*, volume 4667 of *Lecture Notes in Computer Science*, pages 68–83. Springer.

[Ferrari *et al.*(2009)] Ferrari, M., Fiorentini, C., and Fiorino, G. (2009). Towards the use of simplification rules in intuitionistic tableaux. In M. Gavanelli and F. Riguzzi, editors, *CILC09 - Convegno Italiano di Logica Computazionale*.

[Ferrari *et al.*(2010a)] Ferrari, M., Fiorentini, C., and Fiorino, G. (2010a). $\mathcal{BCDL}$: basic constructive description logic. *J. of Automated Reasoning*, **44**(4), 371–399.

[Ferrari *et al.*(2010b)] Ferrari, M., Fiorentini, C., and Fiorino, G. (2010b). fCube: an efficient prover for intuitionistic propositional logic. In C. G. Fermüller and A. Voronkov, editors, *LPAR2010 - Logic for Programming, Artificial Intelligence*

and Reasoning, volume 6397 of *Lecture Notes in Computer Science*, pages 294–301. Springer.

[Fischer Servi(1981)] Fischer Servi, G. (1981). Completeness for non normal intuitionistic modal logics. *Note di Matematica*, **1**, 203–212.

[Fischer Servi(1984)] Fischer Servi, G. (1984). Axiomatizations for some intuitionistic modal logics. *Rend. Sem. Mat. Univers. Polit. Torino*, **42**, 179–194.

[Fitting(1983)] Fitting, M. (1983). *Proof Methods for Modal and Intuitionistic Logics*. Reidel.

[Gabbay(1981)] Gabbay, D. (1981). *Semantical Investigations in Heyting's Intuitionistic Logic*. Reidel.

[Gabbay *et al.*(2003)] Gabbay, D., Kurucz, A., Wolter, F., and Zakharyaschev, M. (2003). *Many-dimensional modal logics: theory and applications*. Studies in logic and the foundations of mathematics. North-Holland.

[Goad(1980)] Goad, C. (1980). Proofs as description of computation. In W. Bibel and R. A. Kowalski, editors, *CADE 1980 - Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 39–52. Springer.

[Hofmann(2005)] Hofmann, M. (2005). Proof-theoretic approach to description-logic. In *LICS 2005 - IEEE Symposium on Logic in Computer Science*, pages 229–237. IEEE Computer Society.

[Horrocks *et al.*(2000)] Horrocks, I., Sattler, U., and Tobies, S. (2000). Practical reasoning for very expressive description logics. *Logic J. of the IGPL*, **8**(3), 239–264.

[Howard(1980)] Howard, W. (1980). The formulae-as-types notion of construction. In J. Seldin and J. Hindley, editors, *To H.B. Curry: Essay on Combinatory Logic, Lambda-calculus and Formalism*. Academic Press.

[Kamide(2010)] Kamide, N. (2010). Paraconsistent description logics revisited. In V. Haarslev, D. Toman, and G. Weddell, editors, *DL2010 - International Workshop on Description Logics*, volume 573 of *CEUR Workshop Proceedings*, pages 197–208. CEUR-WS.org.

[Kaneiwa(2005)] Kaneiwa, K. (2005). Negations in description logic – contraries, contradictories, and subcontraries. In F. Dau, M.-L. Mugnier, and G. Stumme, editors, *ICCS 2005 - International Conference on Conceptual Structures*, pages 66–79. Kassel University Press.

[Kleene(1952)] Kleene, S. (1952). *Introduction to Metamathematics*. Van Nostrand.

[Martin *et al.*(2007)] Martin, D. L., Burstein, M. H., McDermott, D. V., McIlraith, S. A., Paolucci, M., Sycara, K. P., McGuinness, D. L., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to web services with OWL-S. *World Wide Web*, **10**(3), 243–277.

[Matskin and Rao(2002)] Matskin, M. and Rao, J. (2002). Value-added web services composition using automatic program synthesis. In *WES2002 - Web Services, E-Business, and the Semantic Web (CAiSE 2002)*, pages 213–224.

[Matskin and Tyugu(2001)] Matskin, M. and Tyugu, E. (2001). Strategies of structural synthesis of programs and its extensions. *Computers and Artificial Intelligence*, **20**(1).

[Mendler and Scheele(2009)] Mendler, M. and Scheele, S. (2009). Towards a type system for semantic streams. In *SR2009 - Stream Reasoning Workshop (ESWC 2009)*, volume 466 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Mendler and Scheele(2010)] Mendler, M. and Scheele, S. (2010). Towards constructive DL for abstraction and refinement. *J. of Automated Reasoning*, **44**(3), 207–243.

[Miglioli and Ornaghi(1981)] Miglioli, P. and Ornaghi, M. (1981). A logically justified model of computation I. *Fundamenta Informaticae*, **IV**(1), 151–172.

[Miglioli *et al.*(1991)] Miglioli, P., Moscato, U., and Ornaghi, M. (1991). Program specification and synthesis in constructive formal systems. In *LOPSTR 1991 - Logic Program Synthesis and Transformation*, pages 13–26. Springer.

[Miličić(2007)] Miličić, M. (2007). Planning in action formalisms based on DLs: First results. In [Calvanese *et al.*(2007b)], pages 112–122.

[Minsky(1981)] Minsky, M. (1981). A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*, pages 95–128. The MIT Press.

[Narayanan and McIlraith(2002)] Narayanan, S. and McIlraith, S. A. (2002). Simulation, verification and automated composition of web services. In *WWW2002 - World Wide Web Conference*, pages 77–88.

[Odintsov and Wansing(2003)] Odintsov, S. and Wansing, H. (2003). Inconsistency-tolerant description logic. Motivation and basic systems. In V. Hendricks and J. Malinowski, editors, *Trends in Logic. 50 Years of Studia Logica*, pages 301–335. Kluwer Academic Publishers.

[Odintsov and Wansing(2008)] Odintsov, S. and Wansing, H. (2008). Inconsistency-tolerant description logic. Part II: A tableau algorithm for $\mathcal{CALC}^C$. *J. of Applied Logic*, **6**(3), 343–360.

[Patel-Schneider *et al.*(2009)] Patel-Schneider, P. F., Motik, B., and Grau, B. C. (2009). OWL 2 Web Ontology Language direct semantics. W3C Recommendation, W3C. `http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/`.

[Prawitz(1965)] Prawitz, D. (1965). *Natural Deduction*. Almquist and Winksell.

[Sattler *et al.*(2003)] Sattler, U., Calvanese, D., and Molitor, R. (2003). Relationships with other formalisms. In [Baader *et al.*(2003a)], pages 137–177.

[Schmidt-Schauß and Smolka(1991)] Schmidt-Schauß, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, **48**(1), 1–26.

[Simpson(1994)] Simpson, A. (1994). *The Proof Theory and Semantics of Intuitionistic Modal Logic*. Ph.D. thesis, University of Edinburgh.

[Smorynski(1973)] Smorynski, C. (1973). Applications of Kripke models. In A. Troelstra, editor, *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*, pages 324–391. Springer.

[Smullyan(1968)] Smullyan, R. (1968). *First-Order Logic*. Springer.

[Statman(1979)] Statman, R. (1979). Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, **9**, 67–72.

[Tobies(2001)] Tobies, S. (2001). Complexity results and practical algorithms for logics in knowledge representation. *Computing Research Repository*, **cs.LO/0106031**.

[Traverso and Pistore(2004)] Traverso, P. and Pistore, M. (2004). Automated composition of semantic web services into executable processes. In *ISWC 2004 - International Semantic Web Conference*, pages 380–394.

[Troelstra(1973a)] Troelstra, A. (1973a). Intuitionistic formal systems. In [Troelstra(1973b)], pages 1–96.

[Troelstra(1973b)] Troelstra, A., editor (1973b). *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer.

[Troelstra(1977)] Troelstra, A. (1977). Aspects of constructive mathematics. In J. Barwise, editor, *Handbook of Mathematical Logic*. North-Holland.

[Troelstra and Schwichtenberg(1996)] Troelstra, A. and Schwichtenberg, H. (1996). *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

[Troelstra(1999)] Troelstra, A. S. (1999). From constructivism to computer science. *Theoretical Computer Science*, **211**(1-2), 233–252.

[van Harmelen and McGuinness(2004)] van Harmelen, F. and McGuinness, D. L. (2004). OWL Web Ontology Language overview. W3C Recommendation, W3C. `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.

[Villa(2010)] Villa, P. (2010). *Semantics foundations for constructive description logics*. Ph.D. thesis, DICOM – Università degli Studi dell'Insubria.

Index