Fork algebras como herramienta de razonamiento entre especificaciones heterogéneas

López Pombo, Carlos Gustavo

2007

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de computación

Fork algebras como herramienta de razonamiento entre especificaciones heterogéneas

Tesis presentada para optar al título de Doctor de la Universidad de Buenos Aires en el área ciencias de la computación

Carlos Gustavo Lopez Pombo

Director: Marcelo Fabián Frias

Lugar de trabajo: Departamento de computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires.

Buenos Aires, 2007

Fork algebras como herramienta de razonamiento entre especificaciones heterogéneas

Resumen: Las lógicas han sido usadas como sistemas formales para especificar sistemas de *software*. Más aun, las especificaciones lógicas, por ser formales, contribuyen en la aplicación de métodos técnicas correctas de verificación. Diversos formalismos han sido desarrollados para lidiar con estos aspectos y muchos de ellos son eficientes en describir ciertas características de los sistemas de *software*. Por ejemplo, la lógica temporal, proposicional y de primer order, consigue describir la forma en la que los sistemas de *software* evolucionan en el tiempo. La lógica dinámica también permite la especificación de sistemas pero lo hace describiendo cómo los programas transforman el estado del sistema. Estos son sólo algunos ejemplos de cómo una lógica particular permite la especificación de determinados comportamientos de un sistema. La pregunta interesante acerca de este hecho es: Existe un lenguaje ideal para especificar el comportamiento de un sistema?

A pesar de que no vamos a concentrarnos en responder esta pregunta, creemos que ese lenguaje debe tener una sintaxis clara y una semántica fácil de entender, con el propósito de facilitar la comprensión de especificaciones y la aplicación de métodos formales. Entre las propuestas que recopilamos, las instituciones se imponen como un formalismo para razonar entre lógicas, y una institución "universal", que permita razonar entre las lógicas "interesantes" sería la respuesta a nuestra pregunta.

En esta tesis mostraremos que una definición adecuada de la institución de las *fork algebras* es util para razonar entre diversas lógicas proposicionales y de primer orden que aparecen frecuentemente en la especificación de *software.*

Palabras clave: Instituciones, especificaciones heterogéneas, algebras de fork, métodos relacionales en ciencias de la computación

Fork algebras as a tool for reasoning across heterogeneous specifications

Abstract: Logics have often been used as formal systems suitable for the specification of software artifacts. Moreover, logical specifications, being formal, might contribute towards the application of sound verification techniques. Several formalisms have been developed to cope with these aspects and most of them are effective in describing some particular characteristics of software systems. For example, propositional and first-order temporal logics succeed in describing the way software systems evolve along time. Dynamic logic also allows system specification but it does so by describing how programs transform the system state. These are only some examples of how a particular logic allows to specify a certain system behavior. The interesting question about this fact is: Is there an ideal language to specify the behavior of a system?

Although we will not concentrate on answering this question, we believe such language must have a clear syntax and an easy to understand semantics with the purpose of facilitating the comprehension of specifications and the application of formal methods. Among the proposals we surveyed, institutions prevail as a formalism to reason across logics, and a sort of "universal" institution, allowing us to reason across all "interesting" logics would be the answer to our question.

We will show in this thesis that an adequately defined institution of fork algebras is useful for reasoning across many propositional and first-order logics ubiquitous in software specification.

keywords: Institutions, heterogeneous specifications, fork algebra, relational methods in computer science

Contents

List of Figures

List of Tables

Prologue

Angesichts von Hindernissen mag die kürzeste Linie zwischen zwei Punkten die krumme sein.

Eugen Berthold Friedrich Brecht
"Das Leben des Galilei", 1938-1939.

If there are obstacles, the shortest line between two points may be a crooked line.

Eugen Berthold Friedrich Brecht
"The life of Galileo", 1938-1939.

It is probably a paradox to start this work speaking about its end, but in some sense, writing this thesis is not a starting point but the end of the path that lead to it. I refuse to understand this path as a step in the stair of the social structure of science. Instead, I prefer to think about this path as a part of a learning process which will never stop, and this thesis as the systematized result of traversing that path.

For me, it is not possible to conceive what I do as a single and isolated product developed by me (and a few colleagues), in a well-delimited space and time. I understand that what I do, and what I am, are almost the same; not because of my life being only computer science, but rather because my work is influenced by all other aspects of my life. Thus, this work is a joint work with all those persons that, in a way or another, made it possible.

The author wants to thank: Marcelo F. Frias, not only for being my technical advisor, but for his generosity and friendship. My colleagues, students and administrative personnel of the Department of Computer Science of the *Facultad de Ciencias Exactas y Naturales* of the *Universidad de*

Buenos Aires. To Sam Owre, Dexter Kozen, Peter Jipsen, Michael Winter, Paulo A. S. Veloso, Jorge Petrucio Viana, and the RelMiCS (Relational Methods in Computer Science) community for their technical support and kindness. And finally, to Isabel Méndez-Díaz and Alejandro Ríos for being so helpful at the end of this path.

The author wants to dedicate this work to: Angela and Juán Carlos (my loving parents), and Ana (my beautiful and comprehensive companion), for their permanent support. To Yair (my ten years old godson), for materializing the word love. To my big family, specially to Josefina, Dorohe, Ruben, Silvana, Marisa and Claudio, for their responsibility in what I am. To Caco, Daniela, Luana, Temis, Juán, Cristina, Mirta, Julia, Esteban, Gabriel and Gustavo, my family by choice, my partners, the activists of the *Frente José Martí* and the *Corriente universitaria Julio Antonio Mella*, for building the future with their own hands, and, in a very special way, my enormous list of beloved friends.

CHAPTER 1

Introduction and motivations

It is not a novel idea to use logic to specify software systems, with the aim of serving as a formal language which allows the application of software verification techniques. Several languages have been developed to cope with these aspects and most of them are effective (at least) in describing some particular aspects of software systems. For example the propositional temporal logics LTL [**Eme90**] and TL (and their first-order versions, FOLTL and FOTL, respectively) [**MP95**] succeed in describing the way a software system evolves in time along its executions. The propositional dynamic logic PDL (and its first-order version FODL) [**HKT00**] allow to specify systems by characterizing the way actions (and complex programs built from them) transform the system state. These are only two examples of how a particular logic allows to specify a certain system behavior.

Consider the situation where we are interested in specifying a software system and to this end we build separate theories, written in different logics, reflecting its characteristics. We believe this approach has the advantage of supporting modular specifications based on the aspects under consideration. At first sight this approach seems to have a nontrivial disadvantage. Using different languages to describe software systems implies that the possibility of applying formal methods to verify system properties is restricted to each of the separate aspects of the system, not regarding on the interaction among these aspects. Moreover, properties emerging from the interconnection of logics can fall in two situations. On one hand there could be properties which model desired behaviors of the system that can not be proved from the analysis of the separate parts of the specification. On the other hand, these emerging properties might reveal undesired behaviors, or even inconsistencies, of the specification as a whole which would not be detected by observing the separate parts.

Several approaches have been developed to deal with this inter-logic gap produced by the utilization of different formalisms in the specification of software. Henriksen and Thiagarajan proposed in [**HT99**] a formalism to deal with dynamic and temporal properties. To this end they defined the *dynamic linear temporal logic* (DLTL for short) in which formulas are traditional LTL-formulas with the exception of the *until* operator (U) which is strengthened with a test free PDL-program. The result is a logic that is able to express all LTL-formulas and most PDL-formulas.

From other point of view, there is a vast amount of research on reasoning across different logics within the framework of *institutions*. Institutions were introduced by Goguen and Burstall in [**GB84**] and continued in [**GB92**]. Institutions have very rich metalogical foundations and provide a general framework for analyzing the interconnection among logics.

In §2.1 and §2.2 we will return to these two approaches.

Our approach, which falls in the field of institutions, can be described in terms of the **A**$r_{\mathbf{g}}$*entum* project (see Figure 1.1 for a graphical description of **A**$r_{\mathbf{g}}$*entum*). **A**$r_{\mathbf{g}}$*entum* is a CASE tool with relational foundations. Rather than using a single monolithic language for software specification, it uses different logics for modeling different views of systems. Thus, a system specification is a collection of theories coming from different logics. Using the interpretability results for these logics, the theories are translated to a uniform (regarding the language) relational specification. Once a relational specification is obtained, different tools such as bounded model checkers, sat-solvers or theorem provers can be applied in order to verify the relational specification.

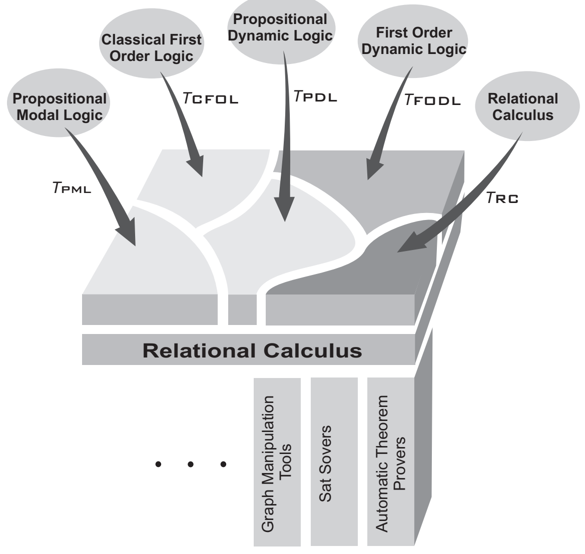

Propositional Dynamic Logic
Classical First Order Logic
First Order Dynamic Logic
Propositional Modal Logic
Relational Calculus
T_{CFOL}
T_{PDL}
T_{FODL}
T_{PML}
T_{RC}
Relational Calculus
. . .
Graph Manipulation Tools
Sat Sovers
Automatic Theorem Provers

FIGURE 1.1. Graphical description of the **A**$r_{\mathbf{g}}$*entum* project.

The spheres located at the top of the figure stand for specifications of different aspects of a system using different logics. The arrows originating at the spheres map logical specifications to a relational specification (located in the box targeted by the arrows). The homogeneous specification can then be analyzed using tools (the lower boxes) which can be plugged into **A**$r_{\mathbf{g}}$*entum*.

The relational language we propose to be used as target language is the fork algebras, presented by Haeberer and Veloso in [**HV91**]. From our point of view, the language of fork algebra has three basic advantages:

(1) a clear syntax,
(2) an easy to understand semantics and
(3) the possibility to apply automatic and semiautomatic tools to verify fork-algebraic specifications.

Our proposal is based on existing work on algebraization of logics. Before going into the concrete results used to interpret logics into fork algebras we present some historical remarks on the development of this vast field of research.

1. Algebraization of logic: historical remarks

Algebraization of logic has been studied for a long time. We point the interested reader to [**Mad91**] for a detailed historical dissertation on the topic. We will only point out some interesting milestones on the field. The oldest most known attempt to assimilate logic reasoning to a calculus associated to an algebraic structure[1] is the one due to George Boole [**Boo47**] in which an algebraization of propositional logic is done by resorting to an equational calculus. Later work due to Charles Sanders Peirce [**Pei70**] is an outstanding effort to create an algebra in which logic reasoning can be carried out. Peirce's work was deeply influenced by De Morgan's "fourth memoir" [**dM64**], where he sketched the theory of dyadic relations under the name "the logic of relations". This effort gave birth to the algebra of binary relations, originally, as an attempt to obtain an algebraization of first-order predicate logic. It was in [**Pei83a**] where Peirce gave to the *algebras of binary relations*, at that time, under the name "the logic of relatives" its final shape. After that, Peirce's system for the algebras of binary relations was extensively developed by Schröder in [**Sch95**].

In [**Tar41**], Tarski calls our attention to the fact that there was almost no research being carried out in the field until Russell and Whitehead [**RW13**] included the algebras of binary relations in the whole of logic. They did it by putting it in the center of their logical system and going further in some new concepts connected with the notion of relation. It was in [**Tar41**] where Tarski committed himself to the development of the *calculus of relations.* In the first place Tarski presents the *elementary theory of binary relations*, as a logical formalization of the algebra of binary relations; thus the *calculus of relations* is obtained from the elementary theory of binary relations by restricting the language to sentences obeying a particular criteria. To the end of this work Tarski stated five questions related to the calculus of relations; from our point of view, three of them are of particular interest:

- Is every model of the calculus of relations isomorphic to an algebra of binary relations?

[1]Massimo Mugnai, in his Italian translation of [**Boo47**] drives our attention to a Spanish philosopher called Raimondo Lullo (1235–1315) who, according to him, seems to be the first to attempt to construct a calculus for logical deduction. It is also claimed that Raimondo Lullo gave a schematic logical proof of the existence of God. See Figure A.1.

- Is it true that every formula that is valid in every algebra of binary relations is provable in the calculus of relations?
- Is it true that every formula of the elementary theory of binary relations can be transformed into an equivalent formula of the calculus of relations?

It was Lyndon in [**Lyn50**] who gave a negative answer to the first two questions by showing a single construction, a finite, non-simple and non-trivial algebra of relations that is not representable as an algebra of binary relations. After that, it was Monk in [**Mon64**] who proved that the class of the algebras of binary relations can not be finitely axiomatized. The third question was answered by Korselt (the proof appears in [**Löw15**]), by proving the equipolence of the relational calculus and the three variable fragment of the dyadic first-order predicate logic.

In [**Tho52**][2] Thompson presented the cylindric algebras. In the case of cylindric algebras, the mission of algebrizing first-order predicate logic was finally accomplished resulting in a complete but non-finitely axiomatizable calculus.

The fork algebras, presented by Armando Haeberer and Paulo A. S. Veloso in [**HV91**], are an extension of the relation algebras, obtained from them by adding a new operator called *fork* (typically represented as "∇"). They arose in the search for a formalism suitable for software specification and verification. In [**Fri02**, Chapter 3, pp. 20] Frias gave a detailed discussion on the evolution of fork algebras and called our attention to the concepts which were responsible of such evolution.

Extensions of fork algebras have been used for program representation and derivation [**FBH98**]. This class of algebras have some particularly attractive features:

- they are isomorphic to algebras whose domain is a set of binary relations (Frias et al. in [**FBH97**] and Gyuris in [**Gyu97**]),
- finite equational calculus (Frias et al. in [**FHV97**]),
- expressive power capable of providing an interpretation language for many logics. Given a logic $\mathcal{L}$, an interpretation is a relational algebraization of $\mathcal{L}$. This is done by resorting to a semantics-preserving mapping $T_{\mathcal{L}} : \mathit{Formulas}_{\mathcal{L}} \to \mathit{RelDes}(X)$ for some set of relational variables X, translating $\mathcal{L}$-formulas to relational terms. Let $\Gamma \cup \{\alpha\} \subseteq \mathit{Formulas}_{\mathcal{L}}$, then the property of being semantic-preserving is characterized by the following condition:
  $\Gamma \models_{\mathcal{L}} \alpha \iff \{\, T_{\mathcal{L}}(\gamma) = 1 \mid \gamma \in \Gamma \,\} \vdash_{\mathsf{CFAU}} T_{\mathcal{L}}(\alpha) = 1$:
  - interpretability of first-order predicate logic (FOL) in fork algebra [**Fri02**],
  - interpretability of PDL in fork algebra [**FO98**],
  - interpretability of first-order dynamic logic (FODL) in fork algebra [**FBM02**],
  - interpretability of LTL, TL [**FL03**] and their first-order versions in fork algebra [**FL06**],
  - interpretability of propositional dynamic linear temporal logic (DLTL) in fork algebra [**FGLR05**].

[2]See also [**TT52**].

2. Related work

2.1. On the development of an *ad-hoc* language for reasoning across logics. As we mentioned before (see [**HT99**]), Henriksen and Thiagarajan proposed the use of propositional dynamic linear temporal logic DLTL as a logic in which it is possible to reason across dynamic and temporal properties. They do so by strengthening the LTL until operator, U, indexing it with PDL-programs. In DLTL, the formula $\alpha \mathsf{U}^{\pi} \beta$ is satisfied in a model M and a trace σ, by τ, a finite prefix of σ, if there exists an extension of τ with τ' such that $\tau\tau'$ is a finite prefix of σ, τ' is a valid execution of the PDL-program π and α is satisfied by every state of τ' until a point in which β is satisfied.

In this work they not only present this logic, but also give a complete axiomatization for the logic obtained by extending the axiomatization of PDL presented by Segerberg in [**Seg77**]. The completeness proof is developed on the basis of the arguments presented by Kozen and Parikh in [**KP81**] to prove the completeness of this PDL axiomatization.

In some sense, if we generalize this approach for inter-logic reasoning, the result is a logic involving syntax, and consequently axioms, to deal with all the relevant aspects we want to include in the specification of the system. From our point of view it leads to a complex language with an obscure semantics and an axiomatization that soon turns to be very difficult to use in order to prove properties of specifications.

2.2. On the institution based approach for reasoning across logics. Institutions were introduced by Goguen and Burstall in [**GB84**]. They present institutions as a framework where it is possible to abstract the model theory of a logical system by "gluing together" all possible signatures, sentences and models. They also prove that under some precise conditions it is possible to use a theorem prover for one institution to prove properties of another by using the notion of institution morphism, also presented in [**GB84**], but extensively developed in [**GB92**]. An institution morphism $\Phi : \mathsf{I} \to \mathsf{I}'$ not only allows us to "borrow" a theorem prover for I' to be used to prove properties of I, but also provides a formal framework to constrain a theory in I' with constrains from I. In order to be precise, constraining a theory of one institution with constrains from another requires a less restrictive construction called semi-morphism [**Tar96**].

In [**Mes92a, MOM01**], Meseguer and Martí-Orliet presented rewriting logic as a unified language to model certain system behavior. Rewriting logic is proposed as a universal logical and semantic framework for computation, showing that different models of computation can be interpreted in a semantics preserving way in rewriting logic. On the theoretical side, our approach shares with rewriting logic the intention of providing a universal specification formalism.

The main, and significant, difference is that while rewriting logic is close to models of computation (and thus more appropriate for dealing with operational formalisms) fork algebras are better suited for handling declarative specification languages.

One of the main advantages of integrating partial specifications is the emergence of properties that occurs as a consequence of the synchronization of system behaviors which are only visible in a global setting. The problem of emergence of properties was widely discussed by Fiadeiro in [**Fia96**]. There he used the concept of *channel*, presented by Fiadeiro and Maibaum in [**FM92**], to compose LTL partial specifications producing a synchronization of the shared symbols. In this sense, channels allow the emergence of properties which are not consequence of the partial descriptions but are yet revealed by their interaction through them. Fiadeiro's treatment of emerging properties is extremely important in our work but we go one step further, we use it in a more complex setting allowing the synchronization of symbols shared by partial specifications written in different logics.

In [**GB84, GB92**] the authors introduce two particular kinds of institutions, *duplex* and *multiplex institutions*. These concepts are used to allow the homogenization of specifications with constraints written in different languages. Although this approach is correct and very useful, it requires the institutions used to express constraints to be related by morphisms to the institutions used to specify the system. This condition establishes limits on the possibilities of using these constructions for a more general interlogic reasoning because we want to consider the scenario where logics are not related by morphisms. An example of this situation is the case of two partial specification of a system, one of them written in PDL and the other in LTL. None of them can act as a constraining theory of the other because there is no possible translation between them.

Sernadas et al. developed in [**SSC97a, SSC97b**] a method for composing partial specifications by forcing some synchronization on the behavior of the system. In [**SSC97a**] they explore two kinds of synchronizations, at the level of consequence systems (synchronization on formulas) and at the level of satisfaction systems (synchronization on models). On the side of synchronization of consequence systems they consider the addition of new mixed rules (with premises from one logic and conclusions from the other) to the proof rules of each constituent logic. These mixed rules allow one to switch from one proof system to another. As we will show in Chapter 6, our approach is similar to the latter kind of synchronization except that we obtain the synchronization on models for free as a direct consequence of using a common language to carry on the homogenization of partial specifications.

In [**Mes89**], Meseguer developed the categorical formalization of logical system by complementing the model theoretic view of a logic (institutions) with its deductive view (entailment system and proof calculus). In this work Meseguer also introduced the notion of institution representation maps. Institution morphisms and representation maps (co-morphisms) were extensively studied by Tarlecki in [**Tar96**], who, at the end of this work, writes:

> "... this suggests that we should strive at a development of a convenient to use proof theory (with support tools!) for a sufficiently rich "universal" institution, and then reuse it for other institutions linked to it by institution representations."

thus exposing what, from his point of view, is an important problem to solve.

We believe that extensions of fork algebras provide a sufficiently rich "universal" institution because of its expressive power capable of interpreting a large menu of different logics. Of course this observation heavily depends on the software designer needs. Following this guideline we believe the most influential works are [**Mes92a, MOM01**], due to Meseguer and Martí-Orliet, and [**Dia02, DF02**], due to Diaconescu and Futatsugi. Those mentioned in second term are specially appealing to us because of their theoretical beauty and at the same time its practical anchor to software development. In this case the Grothendiek institution [**Dia02**] plays the rôle of Tarlecki's "universal" institution, which is obtained by performing a Grothendiek construction [**BW90, Cro93**] (or "flattening" [**TBG91**]) on the CafeOBJ cube [**DF02**]. The CafeOBJ cube is formed by eight different logics and twelve projections between them. These logics can be classified into two well-defined classes of specification languages, a class formed by four variants of algebraic languages and a class formed by four variants of rewriting logic languages. Although we agree on the approach of using heterogeneous languages, we believe other logics like LTL (propositional and first order), PDL, FODL and many others provide a more natural language for software specification.

The key point of our work is the formalization of the foundations of $\mathbf{A}r_{\mathbf{g}}$*entum*. This is carried out by formalizing the logical system (institution) of fork algebras, on top of which we build an heterogeneous specification language, much like CafeOBJ, whose constituent specific logical languages, also presented as institutions, have a particular and well defined rôle in software design. Then, the existing interpretability results for these logics, to extensions of fork algebras, are formalized by means of representation maps between institutions thus, giving the possibility of using fork algebras for interpreting the whole description of the system.

The use of fork algebras as a common algebraic framework for interpreting several logical systems also provides the possibility of introducing cross-logic constrains, to which we call *design decisions*, in the form of algebraic theories, in order to obtain a good integration of partial specifications. In this thesis we discuss several methodological approaches for gluing partial specifications depending on the logics on which they are written.

The thesis is organized as follows. Chapter 2 presents fork algebras within the framework of universal algebra. This is done by reviewing its historical development. Chapter 3 presents the basic definitions of category theory and general logics we will need to formalize the $\mathbf{A}r_{\mathbf{g}}$*entum* foundations. Chapter 4 presents the logical system of fork algebras within the framework of general logics. In Chapter 5 we review an interpretability result and show how it is formalized in the framework of general logics. This formalization provides a witness on how existing interpretability results can be reformulated in this different framework. Chapter 6 presents how the use of fork algebras as a common algebraic language can be used to produce different kinds of synchronization between partial specifications written in different logics. To do this we show, in a general way, how separate specifications can be glued together by means of the addition of algebraic theories expressing design decisions. In Chapter 7 we present a small case-study, developed from its conceptual description, and going throw all the steps, to a complete algebraic description of the system. Chapter 8 discuss the existing validation and verification tools for fork algebras; and finally, in Chapter 9 we draw some conclusions and identify future lines of research.

CHAPTER 2

Relation algebras, fork algebras and ω-closure fork algebras

In this chapter we will present some preliminary definitions and results on the field of algebra. We assume that the reader is familiar with elementary concepts on set theory and first-order predicate logic; for a reference on these fields the interested reader is pointed to [**Bar77**] and [**End72**]. We also assume that the reader is familiar with some basic concepts in the field of universal algebra and point the interested reader to [**BS81**].

The chapter is organized as follows. The first part (§1) is dedicated to provide all the basic definitions from the field of universal algebra that will be needed throughout the rest of the thesis. In §2 we present relation algebra by summarizing the historical development of the field. Finally, in §3, we present fork algebra and its extension to ω-closure fork algebra with urelements.

1. Universal Algebra

In this section we will briefly summarize some fundamental definitions from the field of universal algebra that will be useful in the development of this chapter. We begin by introducing the notion of *algebra*.

Definition 2.1. Let $\{A_k\}_{k\in\mathcal{K}}$ be non-empty family of non-empty sets and $n \in \mathbb{N}$. Then, for $\{k_1, \ldots, k_n\} \subseteq \mathcal{K}$ we define:

$$A_{k_1} \times \ldots \times A_{k_n} = \{ \langle a_1, \ldots, a_n\rangle \mid a_1 \in A_{i_1} \wedge \ldots a_n \in A_{i_n} \} \ .$$

An *n-ary operation* (or *function*) on $A_{k_1} \times \ldots \times A_{k_n}$ is any function f from $A_{k_1} \times \ldots \times A_{k_n}$ to A_k, with $k \in \mathcal{K}$, (denoted $f : A_{k_1} \times \ldots \times A_{k_n} \to A_k$); the sequence $[A_{k_1}, \ldots, A_{k_n}, A_k]$ is said to be the *arity* of f.

Definition 2.2. A *language* (or *similarity type*) is an ordered pair $\langle\mathcal{A}, \mathcal{F}\rangle$, where $\mathcal{A}$ in an indexed set of sort names and $\mathcal{F}$ an indexed set of function symbols $\mathcal{F}$ such that each $f \in \mathcal{F}$ is assigned a sequence of elements of $\mathcal{A}$. This sequence will be called the arity of f, and f will be said to be an n-ary function symbol whenever then length of its arity is $n + 1$.

Let $\langle \mathcal{A}, \mathcal{F} \rangle$ be a similarity type, then, the function *arity* retrieves the arity of the function symbols of $\mathcal{F}$.

The previous definitions can be substantially simplified when $\mathcal{A}$ is a singleton set. Notice that the definition of similarity type is given only in terms of an indexed set of function symbols, and the arity of a function is just a natural number because there exists a unique set on which the functions are defined.

DEFINITION 2.3. Let $\mathcal{T} = \langle \mathcal{A}, \mathcal{F} \rangle$ be a language. Then an *algebra* $\mathfrak{A}$ of similarity type $\mathcal{T}$ is a structure $\langle \mathcal{A}, \{f_i^{A_{i_1},\ldots,A_{i_n},A_i}\}_{i \in \mathcal{I} \wedge \{A_{i_1},\ldots,A_{i_n},A_i\} \subseteq \mathcal{A}} \rangle$ such that:

$$f_i : A_{i_1} \times \ldots \times A_{i_n} \to A_i \text{ for all } i \in \mathcal{I}.$$

Algebras of similarity type $\langle \mathcal{A}, \mathcal{F} \rangle$ are also known as *many-sorted* or *mono-sorted* when $\mathcal{A}$ contains more than one sort, or $\mathcal{A}$ is a singleton set, respectively. From now one, we will restrict ourselves to the case of mono-sorted algebras, unless it is clearly stated.

Given a mono-sorted algebra $\mathfrak{A} = \langle A, \{f_i\}_{i \in \mathcal{I}} \rangle$, A will be called the *universe* of $\mathfrak{A}$ and $\{f_i\}_{i \in \mathcal{I}}$ will be called the *operations* (or *fundamental operations*) of $\mathfrak{A}$. If $\mathit{arity}(f_i) = 0$, f_i will be called a *constant*.

DEFINITION 2.4. An algebra $\mathfrak{A}$ with universe A is *non-trivial* if $|A| \geq 2$.

DEFINITION 2.5. Let $\mathfrak{A} = \langle A, \{f_i\}_{i \in \mathcal{I}} \rangle$ and $\mathfrak{B} = \langle B, \{g_i\}_{i \in \mathcal{I}} \rangle$ be two algebras of the same similarity type, $\mathfrak{B}$ is a *subalgebra* of $\mathfrak{A}$ if

- $B \subseteq A$,
- $\mathit{arity}(f_i) = \mathit{arity}(g_i)$ for all $i \in \mathcal{I}$,
- $g_i(b_1, \ldots, b_{\mathit{arity}(g_i)}) = f_i(b_1, \ldots, b_{\mathit{arity}(f_i)})$ for all $b_1, \ldots, b_{\mathit{arity}(g_i)} \in B$.

DEFINITION 2.6. Let $\mathfrak{A} = \langle A, \{f_i\}_{i \in \mathcal{I}} \rangle$ and $\mathfrak{B} = \langle B, \{g_i\}_{i \in \mathcal{I}} \rangle$ two algebras of the same similarity type, a function $h : A \to B$ is a *homomorphism* from $\mathfrak{A}$ to $\mathfrak{B}$ if

$$h(f_i(a_1, \ldots, a_{\mathit{arity}(f_i)})) = g_i(h(a_1), \ldots, h(a_{\mathit{arity}(g_i)})) \text{ for all } i \in \mathcal{I}.$$

$\mathfrak{B}$ is a *homomorphic image* of $\mathfrak{A}$ if there exists a surjective homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$. A bijective homomorphism is called *isomorphism*.

DEFINITION 2.7. Let $\mathfrak{A}_1 = \langle A_1, \{f_i^{\mathfrak{A}_1}\}_{i \in \mathcal{I}} \rangle$ and $\mathfrak{A}_2 = \langle A_2, \{f_i^{\mathfrak{A}_2}\}_{i \in \mathcal{I}}$ be two algebras of the same similarity type. Then the direct product of $\mathfrak{A}_1$ and $\mathfrak{A}_2$, denoted by $\mathfrak{A}_1 \times \mathfrak{A}_2$ is set to be the algebra $\langle A_1 \times A_2, \{f_i^{\mathfrak{A}_1 \times \mathfrak{A}_2}\}_{i \in \mathcal{I}} \rangle$ such that for all $i \in \mathcal{I}$, $\{a_j^1\}_{1 \leq j \leq \mathit{arity}(f_i^{\mathfrak{A}_1})} \subseteq A_1$ and $\{a_j^2\}_{1 \leq j \leq \mathit{arity}(f_i^{\mathfrak{A}_2})} \subseteq A_2$:

$$f_i^{\mathfrak{A}_1 \times \mathfrak{A}_2}(\langle a_1^1, a_1^2 \rangle, \ldots, \langle a^1_{\mathit{arity}(f_i^{\mathfrak{A}_1})}, a^2_{\mathit{arity}(f_i^{\mathfrak{A}_2})} \rangle) = \\ \langle f^{\mathfrak{A}_1}(a_1^1, \ldots, a^1_{\mathit{arity}(f^{\mathfrak{A}_1})}), f_i^{\mathfrak{A}_2}(a_1^2, \ldots, a^2_{\mathit{arity}(f_i^{\mathfrak{A}_2})}) \rangle .$$

When dealing with many-sorted algebras, the function $\mathbf{Rd}_T$ takes reducts to the similarity type T; to take reduct of a class of algebras of similarity type $\langle \mathcal{A}, \mathcal{F} \rangle$, to a similarity type $\langle \mathcal{A}', \mathcal{F}' \rangle$ means to forget all the domains of

$\mathcal{A}$ not mentioned in $\mathcal{A}'$ and the functions of $\mathcal{F}$ not mentioned in $\mathcal{F}'$. Notice that this definition requires $\mathcal{A}'$ to be included in $\mathcal{A}$, and $\mathcal{F}'$ to be included in $\mathcal{F}$. Once again, the definition is simpler when dealing with mono-sorted algebras. The operator **S** closes a class of algebras of the same similarity type under subalgebras, **P** under direct product, **I** under isomorphisms and **H** under homomorphisms.

A class of algebras closed under subalgebras, direct products and homomorphisms is a *variety.* In [**Bir35**], Birkhoff proved that varieties can always be axiomatized by a (possibly infinite) set of equations.

THEOREM 2.1. ([**Bir35**])
Let K be a class of algebras of the same similarity type. Then K is an equational class if and only if K is a variety. ■

DEFINITION 2.8. Given an algebra $\mathfrak{A}$ and a class of algebras K, $\mathfrak{A}$ is *representable* in K if there exists $\mathfrak{B} \in K$ such that $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$. This notion generalizes as follows: a class of algebras K_1 is representable in a class of algebras K_2 if every member of K_1 is representable in K_2.

Before continuing with this presentation we will present the concept of lattice and some definitions and results that will be useful in order to characterize some properties of algebraic structures.

Lattices can be defined from two different points of view. On one hand lattices can be viewed as algebras, and on the other hand they can be defined from the observation of their geometric properties. We will first give the geometric definition.

DEFINITION 2.9. Let A be a set, a binary relation $\leq \subseteq A \times A$ is a partial order on A if for all $a, b, c \in A$ it satisfies:

- $a \leq a$ (reflexivity),
- if $a \leq b$ and $b \leq a$, then $a = b$ (antisymmetry),
- if $a \leq b$ and $b \leq c$, then $a \leq c$ (transitivity).

DEFINITION 2.10. Let P be a set and $\leq \subseteq P \times P$, $\langle P, \leq \rangle$ is said to be a *partially ordered set* (poset for short).

DEFINITION 2.11. Let $\langle P, \leq \rangle$ be a poset and $A \subseteq P$. An element $p \in P$ is an *upper bound* for A if for all $a \in A$, $a \leq p$. An element $p \in P$ is the *least upper bound* (or supremum) of A (denoted as *sup* A) if p is an upper bound of A, and if for every $a \in A$, $a \leq b$, then $p \leq b$. Dually, we can define *lower bound* and *greatest lower bound* (or infimum) of A (denoted as $inf\ A$).

DEFINITION 2.12. A poset $\langle P, \leq \rangle$ is *complete* if for every $A \subseteq P$ both *sup* A and $inf\ A$ exist (in P).

DEFINITION 2.13. A poset $\langle L, \leq \rangle$ is a *lattice* if for every elements $a, b \in L$, $sup\ \{a, b\}, inf\ \{a, b\} \in L$.

THEOREM 2.2. ([**BS81**, Chapter I, §4, Theorem 4.2])
If $\langle P, \leq \rangle$ is a complete poset, then $\langle P, \leq \rangle$ is a complete lattice.

DEFINITION 2.14. A binary relation $R \subseteq A \times A$ is an *equivalence relation* if for all $a, b, c \in A$ the following three properties are satisfied:

- $a\ R\ a$ (reflexivity),

- If $a\ R\ b$, then $b\ R\ a$ (symmetry),
- If $a\ R\ b$ and $b\ R\ c$, then $a\ R\ c$ (transitivity).

The set of all equivalence relations on a set A will be denoted as $Eq(A)$.

THEOREM 2.3. ([**BS81**, Chapter I, §4, Theorem 4.5])
$\langle Eq(A), \subseteq\rangle$ is a complete lattice. ■

DEFINITION 2.15. Let $\mathfrak{A} = \langle A, \{f_i\}_{i\in\mathcal{I}}\rangle$ be an algebra and $E \in Eq(A)$. Then E is a congruence on $\mathfrak{A}$ if E satisfies the following *compatibility property*:

$$\text{For each } i \in \mathcal{I}, \text{ if } a_j\ E\ b_j \text{ for all } 1 \le j \le arity(f_i), \text{ then } f_i(a_1, \ldots, a_{arity(f_i)})\ E\ f_i(b_1, \ldots, b_{arity(f_i)}).$$

The set of all congruences on $\mathfrak{A}$ is denoted as $Con\ \mathfrak{A}$.

THEOREM 2.4. ([**BS81**, Chapter II, §5, Theorem 5.3])
Let $\mathfrak{A}$ be an algebra, then $\langle Con\ \mathfrak{A}, \subseteq\rangle$ is a complete sub-lattice of $\langle Eq(A), \subseteq\rangle$, the lattice of equivalence relations on A. ■

DEFINITION 2.16. Let $\mathfrak{A} = \langle A, \{f_i\}_{i\in\mathcal{I}}\rangle$, then the congruence lattice of $\mathfrak{A}$, denoted as **Con** $\mathfrak{A}$ is the lattice whose universe is $Con\ \mathfrak{A}$, and meets and joins are calculated as in $\langle Eq(A), \wedge, \vee\rangle$.

DEFINITION 2.17. An algebra $\mathfrak{A}$ is *simple* if **Con** $\mathfrak{A} = \{\mathsf{U}, \mathsf{I}\}$.

Intuitively, these two congruences are the universal relation and the identity respectively.

From the algebraic point of view a lattice is defined as follows.

DEFINITION 2.18. An algebraic structure $\langle L, \cdot, +\rangle$ is a *lattice* if for all $x, y, z \in L$, the following identities are satisfied:

- Commutative laws:
  - $x\cdot y = y\cdot x$,
  - $x+y = y+x$,
- Associative laws:
  - $x\cdot(y\cdot z) = (x\cdot y)\cdot z$,
  - $x+(y+z) = (x+y)+z$,
- Idempotent laws:
  - $x\cdot x = x$,
  - $x+x = x$,
- Absorption laws:
  - $x = x\cdot(x+y)$,
  - $x = x+(x\cdot y)$.

The following theorem, which is a well-known result in the field of universal algebra, shows the equivalence between the geometric and algebraic definitions of lattice.

THEOREM 2.5. *The following two properties hold:*

- *If $\langle L, \le\rangle$ is a lattice, then $\langle L, sup, inf\rangle$ is a lattice, and*
- *If $\langle L, \cdot, +\rangle$ is a lattice, then $\langle L, \sqsubseteq\rangle$ (where for all $a, b \in L$, $a \sqsubseteq b$ if and only if $a+b = b$), is lattice.*

■

2. Boolean algebras and relation algebras

In 1847 George Boole [**Boo47**] presented his algebraization of the propositional logic, referred by Boole at that time as *the algebra of logic*. We will not get attached to Boole's presentation of the calculus, in order to give a more modern definition of Boolean algebra. For a general reference in the field of Boolean algebra, the interested reader is pointed to [**Hal63, Sik64**].

Definition 2.19. ([**HMT71**, §1.1, pp. 159])
A *Boolean algebra* is an algebraic structure $\langle A, +, \cdot, ^{-}, 0, 1\rangle$ in which $+$ and $\cdot$ are binary operations, $^{-}$ is a unary operation and 0 and 1 are distinguished elements, satisfying the following identities for all $x, y, z \in A$:

$$x+y = y+x \tag{Ax. 1}$$

$$x\cdot y = y\cdot x \tag{Ax. 2}$$

$$x+(y\cdot z) = (x+y)\cdot(x+z) \tag{Ax. 3}$$

$$x\cdot(y+z) = (x\cdot y)+(x\cdot z) \tag{Ax. 4}$$

$$x+0 = x \tag{Ax. 5}$$

$$x\cdot 1 = x \tag{Ax. 6}$$

$$x+\overline{x} = 1 \tag{Ax. 7}$$

$$x\cdot\overline{x} = 0 \tag{Ax. 8}$$

The class of all Boolean algebras will be denoted as BA.

Given $\mathfrak{A} \in \mathsf{BA}$, $x \leq_{\mathfrak{A}} y$ is the partial ordering on the elements of $|\mathfrak{A}|$ (see Theorem 2.5).

Definition 2.20. An algebraic structure $\langle A, \cup, \cap, ^{-}, \emptyset, E\rangle$ where $A \subseteq 2^U$ for some set U; $\cup$, $\cap$, $^{-}$, $\emptyset$ have their standard set theoretical meaning, $E \in A$ and $\bigcup_{r\in A} r \subseteq E$, such that A is closed under the operations, is called a *set Boolean algebra.*[1]

Definition 2.21. Let $\mathfrak{A} \in \mathsf{BA}$, a non-zero a in $|\mathfrak{A}|$ is said to be an atomic element if it satisfies either of the following properties:

- $b \leq a$ implies $b = a$ or $b = 0$,
- for all b, $a\cdot b = 0$ or $a\cdot b = a$.

Definition 2.22. Let $\mathfrak{A} \in \mathsf{BA}$, a class π of atomic elements of $|\mathfrak{A}|$ is said to be an *atomic basis* if every non-zero element b of $|\mathfrak{A}|$ satisfy that there exists $\pi' \subseteq \pi$ such that:

$$b = \Sigma_{a\in\pi'} a\ .$$

Definition 2.23. A class π of atomic elements is said to be a *complete atomic system* if $b = 0$ is the only element such that for all $a \in \pi$, $b\cdot a = 0$.

[1]Let U be a set, 2^U denotes the set of subsets of U.

THEOREM 2.6. ([**Sto36**, §4, Theorem 6])
Let $\mathfrak{A} \in \mathsf{BA}$. Then if a and b are atomic elements of $|\mathfrak{A}|$, then $a = b$ or $a \cdot b = 0$. ■

THEOREM 2.7. ([**Sto36**, §4, Theorem 7])
Let $\mathfrak{A} \in \mathsf{BA}$. Then a complete atomic system in a Boolean algebra contains every atomic element in $|\mathfrak{A}|$. ■

DEFINITION 2.24. Let $\mathfrak{A}$ be an algebra. Then $\mathfrak{A}$ is *atomistic* if there exists a complete atomic system for $|\mathfrak{A}|$.

If $\mathfrak{A} \in \mathsf{BA}$, by $At(\mathfrak{A})$ we denote the set of atoms of $\mathfrak{A}$.

Peirce's *logic of relatives* can be defined in terms of a class of algebras. To this purpose we introduce the notion of *algebras of binary relations*.

DEFINITION 2.25. An *algebra of binary relations* is an algebraic structure $\langle A, \cup, \cap, ^-, \emptyset, E, \circ, ^\smile, Id\rangle$ in which A is a set of binary relations on a set U, $\cup$, $\cap$ and $\circ$ are binary operations, $^-$ and $^\smile$ are unary operations and $\emptyset$, E and Id are distinguished elements of A satisfying:

- A is closed under $\cup$ (i.e. set union),
- A is closed under $\cap$ (i.e. set intersection),
- A is closed under $^-$ (i.e. set complement with respect to E),
- $\emptyset \in A$ is the empty relation on the set U,
- $E \in A$ and $\bigcup_{r \in A} r \subseteq E$,
- A is closed under $\circ$, defined as follows

$$x \circ y = \{ \langle a, b\rangle \in U \times U \mid (\exists c)(\langle a, c\rangle \in x \wedge \langle c, b\rangle \in y) \} ,$$

- A is closed under $^\smile$, defined as follows

$$\breve{x} = \{ \langle a, b\rangle \in U \times U \mid \langle b, a\rangle \in x \} ,$$

- $Id \in A$ is the identity relation on the set U.

From now on, and for the sake of clarifying concepts, we will refer to the algebras of binary relations as *proper relation algebras* (PRA for short) following the name used by Jonsson and Tarski in [**JT51, JT52**].

DEFINITION 2.26. A proper relation algebra $\langle A, \cup, \cap, ^-, \emptyset, E, \circ, ^\smile, Id\rangle$ on a set U is said to be *full* if $A = 2^{U \times U}$.

Notice that if a proper relation algebra is full, then it is simple. The proof of this property can be found in [**JT52**, Theorem 4.10].

The following three lemmas about full proper relation algebras will be useful in Chapter 5.

LEMMA 2.1. *Let $\mathfrak{A}, \mathfrak{B} \in \mathsf{PRA}$ such that both $\mathfrak{A}$ and $\mathfrak{B}$ are full, $\mathfrak{B}$ is non-trivial, and δ be a homomorphism of the form $\delta : |\mathfrak{A}| \to |\mathfrak{B}|$.*

Then if $a \in At(\mathfrak{A})$, $\delta(a) \neq \emptyset$.

PROOF. Suppose $a \in At(\mathfrak{A})$ such that $\delta(a) = \emptyset$.

$$\begin{array}{ll} & U_{\mathfrak{B}} \times U_{\mathfrak{B}} \\ = & \delta(U_{\mathfrak{A}} \times U_{\mathfrak{A}}) \\ & \quad \text{[because } \delta \text{ is a homomorphism]} \\ = & \delta((U_{\mathfrak{A}} \times U_{\mathfrak{A}}) \circ a \circ (U_{\mathfrak{A}} \times U_{\mathfrak{A}})) \\ & \quad \text{[because } \mathfrak{A} \text{ is simple and [}\textbf{JT52}\text{, Theorem 4.10]]} \\ = & \delta(U_{\mathfrak{A}} \times U_{\mathfrak{A}}) \circ \delta(a) \circ \delta(U_{\mathfrak{A}} \times U_{\mathfrak{A}}) \\ & \quad \text{[because } \delta \text{ is a homomorphism]} \\ = & U_{\mathfrak{B}} \times U_{\mathfrak{B}} \circ \delta(a) \circ U_{\mathfrak{B}} \times U_{\mathfrak{B}} \\ & \quad \text{[because } \delta \text{ is a homomorphism]} \\ = & U_{\mathfrak{B}} \times U_{\mathfrak{B}} \circ \emptyset \circ U_{\mathfrak{B}} \times U_{\mathfrak{B}} \\ & \quad \text{[by hypothesis]} \\ = & \emptyset \end{array}$$

Which is a contradiction because $\mathfrak{B}$ is non-trivial. ∎

LEMMA 2.2. *Let $\mathfrak{A}, \mathfrak{B} \in$ PRA such that both $\mathfrak{A}$ and $\mathfrak{B}$ are full, $\mathfrak{B}$ is non-trivial, and δ be a homomorphism of the form $\delta : |\mathfrak{A}| \to |\mathfrak{B}|$.*
Then if $a \in At(\mathfrak{A})$, then $\delta(a) \in At(\mathfrak{B})$.

PROOF. As δ is a homomorphism, for all $r \in |\mathfrak{A}|$, $\delta(r^{\diamond}) = \delta(r)^{\diamond}$. If we consider the case $a \in At(\mathfrak{A})$ we obtain that $\delta(a) = \delta(a^{\diamond}) = \delta(a)^{\diamond}$. Thus, by Lemma 2.1, $\delta(a)$ must be an atom. ∎

LEMMA 2.3. *Let $\mathfrak{A}, \mathfrak{B} \in$ PRA such that both $\mathfrak{A}$ and $\mathfrak{B}$ are full, $\mathfrak{B}$ is non-trivial, and δ be a homomorphism of the form $\delta : |\mathfrak{A}| \to |\mathfrak{B}|$.*
Then δ is a bijection.

PROOF. We first prove that δ is injective. Let $r, s \in |\mathfrak{A}|$ such that $\delta(r) = \delta(s)$. As $\mathfrak{A}$ is atomistic, then $\delta(r) = \delta(\cup_{a \in At(\mathfrak{A}) \wedge a \leq r} a) = \cup_{a \in At(\mathfrak{A}) \wedge a \leq r} \delta(a)$. By applying the same reasoning to s, we obtain that $\cup_{a \in At(\mathfrak{A}) \wedge a \leq r} \delta(a) = \cup_{a \in At(\mathfrak{A}) \wedge a \leq s} \delta(a)$. Now, as δ is a homomorphism, if $a, b \in At(\mathfrak{A})$, then $\delta(a) = \delta(b)$ implies $a = b$. Thus, we obtain that $r = s$.

The proof of the surjectivity of δ follows from the fact that $\delta(U_{\mathfrak{A}} \times U_{\mathfrak{A}}) = U_{\mathfrak{B}} \times U_{\mathfrak{B}}$. Then, $\delta(\cup_{a \in At(\mathfrak{A}) \wedge a \leq U_{\mathfrak{A}} \times U_{\mathfrak{A}}} a) = \cup_{a \in At(\mathfrak{B}) \wedge a \leq U_{\mathfrak{B}} \times U_{\mathfrak{B}}} a$. Now, as δ is a homomorphism, $\cup_{a \in At(\mathfrak{A}) \wedge a \leq U_{\mathfrak{A}} \times U_{\mathfrak{A}}} \delta(a) = \cup_{a \in At(\mathfrak{B}) \wedge a \leq U_{\mathfrak{B}} \times U_{\mathfrak{B}}} a$. By Lemma 2.2, $At(\mathfrak{B}) = \{ \delta(a) \mid a \in At(\mathfrak{A}) \}$. Thus, recalling on the fact that any $r \in |\mathfrak{B}|$ satisfies that $r = \cup_{a \in At(\mathfrak{B}) \wedge a \leq r} a$, there exists $s \in |\mathfrak{A}|$ such that $s = \cup_{a \in At(\mathfrak{A}) \wedge \delta(a) \leq r} a$ and $\delta(s) = r$. ∎

In [**Tar41**], Tarski presented the *elementary theory of binary relations* (ETBR for short) as a calculus whose intended models are the proper relation algebras.

DEFINITION 2.27. Let $\mathcal{R}$ be a set of relation variables, then the set of relation designations is the smallest set $RelDes(\mathcal{R})$ such that:

- $\mathcal{R} \cup \{ 1, 0, 1' \} \subseteq RelDes(\mathcal{R})$,
- If $r, s \in RelDes(\mathcal{R})$, then $\{ r+s, r \cdot s, \overline{r}, r ; s, \breve{r} \} \subseteq RelDes(\mathcal{R})$.

Let $\mathcal{I}$ be a set of individual variables and $\mathcal{R}$ be a set of relation variables, then the set of atomic formulas is the smallest sets $AtForm(\mathcal{I}, \mathcal{R})$ such that:

- If $x, y \in \mathcal{I}$ and $r \in RelDes(\mathcal{R})$, then $x \; r \; y \in AtForm(\mathcal{I}, \mathcal{R})$,

- If $r, s \in \mathit{RelDes}(\mathcal{R})$, then $r = s \in \mathit{AtForm}(\mathcal{I}, \mathcal{R})$.

Then, the set of formulas of ETBR is the smallest set $\mathit{ETBRForm}(\mathcal{I}, \mathcal{R})$

- $\mathit{AtForm}(\mathcal{I}, \mathcal{R}) \subseteq \mathit{ETBRForm}(\mathcal{I}, \mathcal{R})$,
- If $f, g \in \mathit{ETBRForm}(\mathcal{I}, \mathcal{R})$ and $x \in \mathcal{I}$, then $\{ \neg f, f \vee g, \exists x(f) \} \subseteq \mathit{ETBRForm}(\mathcal{I}, \mathcal{R})$.

The rest of the propositional operators such as conjunction ($\wedge$), and implication ($\Longrightarrow$), are defined as always in terms of the negation ($\neg$) and disjunction ($\vee$) operators as $\alpha \wedge \beta \equiv \neg(\neg\alpha \vee \neg\beta)$ and $\alpha \Longrightarrow \beta \equiv \neg\alpha \vee \beta$ respectively. The universal quantifier ($\forall$) is defined in terms of the existential quantifier as $\forall x(\alpha) \equiv \neg\exists x(\neg\alpha)$.

Definition 2.28. ([**Tar41**, pp. 75–76])
Let $\mathcal{I}$ be a set of individual variables and $\mathcal{R}$ be a set of relation variables, then ETBR is defined as follows:

- Formulas: $\mathit{ETBRForm}(\mathcal{I}, \mathcal{R})$,
- Axioms[2]:
  - Axioms for the propositional operators,
  - Axioms for the relational operators:

$$\begin{array}{l}
\forall x, y(x\ 1\ y)\ , \\
\forall x, y(x\ r{\cdot}s\ y \Longrightarrow x\ r\ y \wedge x\ s\ y)\ , \\
\forall x, y(\neg x\ 0\ y)\ , \\
\forall x, y(x\ r{+}s\ y \Longrightarrow x\ r\ y \vee x\ s\ y)\ , \\
\forall x, y(x\ \overline{r}\ y \Longrightarrow \neg x\ r\ y)\ , \\
\forall x(x\ 1'\ x)\ , \\
\forall x, y, z(x\ r\ y \wedge y\ 1'\ z \Longrightarrow x\ r\ z)\ , \\
\forall x, y(x\ \breve{r}\ y \Longleftrightarrow y\ r\ x)\ , \\
\forall x, y(x\ r{;}s\ y \Longrightarrow \exists z(x\ r\ z \wedge z\ s\ y))\ , \\
r = s \Longleftrightarrow \forall x, y(x\ r\ y \Longleftrightarrow x\ s\ y)\ .
\end{array}$$

- Inference rules:
  - Inference rules for the propositional operators (if required),
  - Modus ponens:
  $$\frac{\alpha \Longrightarrow \beta \qquad \alpha}{\beta}$$
  - Generalization rule:
  $$\frac{\alpha}{\forall x(\alpha)}$$

As we mentioned in a previous paragraph, Tarski presented ETBR with the aim of giving a calculus for the class PRA. Later on, in [**JT52**, Theorem 4.10], Jonsson and Tarski proved that it was not true, because ETBR

[2]In Tarski's presentation of ETBR [**Tar41**], two other concepts are included, a binary operator called relative addition ($\dagger$) and a constant symbol for the diversity ($0'$). This two new concepts acts as relative, or Peircean, counterpart of the Boolean concept of addition ($+$) and empty relation (0).

Relative addition and the diversity are characterized by the following two axioms:

$$\begin{array}{l}
\forall x, y(x\ r \dagger s\ y \Longrightarrow \forall z(x\ r\ z \wedge z\ s\ y)) \\
\forall x, y(x\ 0'\ y \Longleftrightarrow \neg(x\ 1'\ y))
\end{array}$$

forces models to be simple. As we will see later on in this work, there are more important reasons why this relationship does not hold.

Even when there is no doubt that this calculus is appropriate to reason about properties where the only variables occurring in the formula are relation variables, it is easy to see that the only way to build a proof is by using the last axiom of ETBR (i.e. definition of $=$ in terms of a formula on individual variables) and then by reasoning as in FOL. This was Tarski's motivation to present, also in [**Tar41**], the *calculus of relations* (CR for short) as a calculus in which it should be possible to prove properties of relations but without the need of using elements from outside the language. To this purpose he confined himself to formulas in which no individuals variables appear. Thus formulas are either atomic formulas of the form $r = s$ or a formula built from atomic formulas using propositional connectives.

The following definitions present CR.

Definition 2.29. Let $\mathcal{R}$ be a set of relation variables, then the set of formulas of CR is the smallest set $\mathit{CRForm}(\mathcal{R})$ such that:

- If $r, s \in \mathit{RelDes}(\mathcal{R})$, then $r = s \in \mathit{CRForm}(\mathcal{R})$,
- If $f, g \in \mathit{CRForm}(\mathcal{R})$, then $\{\neg f, f \vee g\} \subseteq \mathit{CRForm}(\mathcal{R})$.

As we mentioned before, the remaining propositional connectives are defined in terms of the negation ($\neg$) and disjunction ($\vee$) connectives.

Definition 2.30. ([**Tar41**, pp. 76–77])
Let $\mathcal{R}$ be a set of relation variables, then CR is defined as follows:

- Formulas: $\mathit{CRForm}(\mathcal{R})$,
- Axioms[3]:
  - Axioms for the propositional operators,
  - Axioms for the relational operators:

$$\begin{array}{l}
(r = s \wedge r = t) \Longrightarrow s = t \ , \\
r = s \Longrightarrow (r{+}t = s{+}t \wedge r{\cdot}t = s{\cdot}t) \ , \\
r{+}s = s{+}r \wedge r{\cdot}s = s{\cdot}r \ , \\
r{+}(s{\cdot}t) = (r{+}s){\cdot}(r{+}t) \wedge r{\cdot}(s{+}t) = (r{\cdot}s){+}(r{\cdot}t) \ , \\
r + 0 = r \wedge r{\cdot}1 = r \ , \\
r{+}\overline{r} = 1 \wedge r{\cdot}\overline{r} = 0 \ , \\
\overline{1} = 0 \ , \\
\breve{\breve{r}} = r \ , \\
(r;s)\breve{} = \breve{s};\breve{r} \ , \\
r;(s;t) = (r;s);t \ , \\
r;1' = r \ , \\
r;1 = 1 \vee 1;\overline{r} = 1 \ , \\
(r;s){\cdot}\breve{t} = 0 \Longrightarrow (s;t){\cdot}\breve{r} = 0 \ .
\end{array}$$

- Inference rules:

[3]As we mentioned before Tarski presented CR incorporating axioms in order to characterize the relative addition ($\dagger$) and the diversity ($0'$). Tarski's axioms for these two operators are:

$$r \dagger s = \overline{\overline{r};\overline{s}}$$
$$0' = \overline{1'}$$

- Inference rules for the propositional operators (if required),
- Modus ponens:

$$\frac{\alpha \Longrightarrow \beta \qquad \alpha}{\beta}$$

- Inference rules for $=$:

$$\frac{}{r = r}$$

$$\frac{s = r}{r = s}$$

$$\frac{s = t \qquad r = s}{r = t}$$

$$\frac{r_1 = s_1 \qquad \cdots \qquad r_k = s_k}{E(r_1, \ldots, r_k) = E(s_1, \ldots, s_k)}$$

In [**Tar41**, pp. 87], Tarski argued, as a conclusion of the development of some proofs in CR, that it could be presented as a purely equational calculus in order to avoid the inclusion of axioms and inference rules for the sentential calculus. And it was in [**JT52**], where Jonsson and Tarski proved that axiom $r;1 = 1 \vee 1;\overline{r} = 1$ forces the models to be simple, property that is not necessarily satisfied by the proper relation algebras, so that is why they moved to a definition of CR in which this axiom is not present.

Now we present CR but as a purely equational calculus. This is the presentation of CR we will used from now on.

Definition 2.31. Let $\mathcal{R}$ be a set of relation variables, then the set of formulas of CR is the smallest set $\mathit{CRForm}(\mathcal{R})$ such that:

- If $r, s \in \mathit{RelDes}(\mathcal{R})$, then $r = s \in \mathit{CRForm}(\mathcal{R})$,

Definition 2.32. ([**JT52**, Definition 4.1])
Let $\mathcal{R}$ be a set of relation variables, then CR is defined as follows:

- Formulas: $\mathit{CRForm}(\mathcal{R})$,
- Axioms:
  - Axioms (1) – (8) of Definition 2.19 for the Boolean operators
  - The following axioms for the relational operators[4]: for all $r, s, t \in A$

$$r;(s;t) = (r;s);t \tag{Ax. 9}$$

$$(r+s);t = (r;t)+(s;t) \tag{Ax. 10}$$

$$(r+s)^{\smile} = \breve{r}+\breve{s} \tag{Ax. 11}$$

$$\breve{\breve{r}} = r \tag{Ax. 12}$$

$$r;1' = r \tag{Ax. 13}$$

[4]Axiom (15), or Dedekind formula, is equivalent to

$$r;s\cdot t = 0 \text{ iff } t;\breve{s}\cdot r = 0 \text{ iff } \breve{r};t\cdot s = 0$$

known as *cycle rule.*

$$(r;s)\breve{} = \breve{s};\breve{r} \tag{Ax. 14}$$

$$(r;s)\cdot t \leq (r\cdot(t;\breve{s}));(s\cdot(\breve{r};t)) \tag{Ax. 15}$$

- Inference rules: the inference rules presented in Definition 2.30 for =.

Definition 2.33. The class of *relation algebras* (RA for short) is the class of the models of the identities provable in CR.

It was in [**CT51**] where Chin and Tarski proved that Axioms (9) – (15) can be proved from the first version of CR (excluding the axiom that forces models to be simple) and viceversa.

As we mentioned in Chapter 1, at the end of [**Tar41**], Tarski formulated some questions about the relation between ETBR and CR.

It is easy to see that the class PRA is contained in RA by verifying that the operations on binary relations presented in Definition 2.25 satisfy the axioms presented in Definition 2.32. The converse property (i.e. the representability of RA in PRA) is the first of Tarski's question.

It was Roger Lyndon who gave an answer to this question by constructing a finite, simple and non-trivial relation algebra with 56 atoms which was not representable[5].

Theorem 2.8. ([**Lyn50**, §8, Theorem III])
There exists a finite relation algebra which is not isomorphic to any proper relation algebra. ■

The proof of Theorem 2.8 is obtained by providing a family of conditions C which are satisfied by every complete relation algebra which is isomorphic to a proper relation algebra ([**Lyn50**, §6, Theorem I]). Then, as finite relation algebras are complete, the converse of [**Lyn50**, §6, Theorem I] also holds ([**Lyn50**, §7, Theorem II]). Finally, the proof follows by constructing a finite relation algebra in which one of the conditions in C fails.

Corollary 2.1. RA *is not representable in* PRA. ■

Let $\mathcal{R}$ is a set of relational variables, then *algebraic formulas* are obtained by composing equations from *CRForm*$(\mathcal{R})$ by using propositional connectives. An *algebraic axiom* for RA is any algebraic formula which is true in every proper relation algebra.

Lyndon showed in [**Lyn50**, §14, Theorem IV] that the class PRA can not be axiomatized by a set of algebraic axioms by showing that one of the conditions in C is not a consequence of any set of algebraic axioms for RA. Thus answering Tarski's second question negatively.

This theorem proves that CR is not complete for the class PRA. An immediate consequence of this result is that there exist formulas which are valid in every proper relation algebra but fail to be valid in a relation algebra. Lyndon's example of such a formula is the equation:

[5]Lyndon's construction was later improved by Ralph McKenzie in [**McK70**] by constructing another non-representable relation algebra but with only four atoms.

$$r;s\cdot t;u\cdot v;w \leq r;\left(\breve{r};t\cdot s;\breve{u}\cdot\left(\breve{r};v\cdot s;\breve{w}\right);\left(\breve{v};t\cdot w;\breve{u}\right)\right);u\ .$$

A relation algebra is *representable* if it is isomorphic to a proper relation algebra. In [**Tar55a**], Tarski proved that the class of *representable relation algebras* can be axiomatized by a set of equational axioms by proving that it is a variety. This result is synthesized in the following two theorems.

Theorem 2.9. ([**Tar55a**])
The class of representable relation algebras *is a variety.* ■

It was Donald Monk who proved in [**Mon64**] that a set of equations axiomatizing PRA could never be finite.

Theorem 2.10. ([**Mon64**])
PRA *is not finitely axiomatizable.* ■

Once again, this necessarily means that there must be a property that holds in every proper relation algebra but fails in some relation algebra, and in [**TG87**, pp. 55] Tarski and Givant gave an example of such property, which fails in the non-representable relation algebra presented by Ralph McKenzie in [**McK70**]. This also results in a negative answer to Tarski's second question about CR.

With regard to Tarski's third question, the answer was given by Korselt, (the proof appeared in [**Löw15**]), by proving the equipolence of CR and the three variable fragment of the dyadic first-order predicate logic. In [**TG87**, §3.4 (iv)], Tarski and Givant presented the following formula of ETBR which is not equivalent to any formula of the CR:

$$\forall x\forall y\forall z\exists u(u\ 0'\ x\wedge u\ 0'\ y\wedge u\ 0'\ z)\ .$$

3. Fork algebras and ω-closure fork algebras

As we mentioned in Chapter 1, the *fork algebras* were introduced by Armando Haeberer and Paulo A. S. Veloso in [**HV91**] in the search for a formalism that could be suitable for software specification and verification. A *proper fork algebra* is the extension of a proper relation algebra, obtained by adding a new operator called *fork* (denoted by "$\underline{\nabla}$"). Now, let U be a set and $\star : U\times U\to U$, then we can define fork of binary relations in the following way:

$$(1)\qquad r\underline{\nabla}s=\{\,\langle a,b\rangle\in U\times U\mid\exists x,y\in U\mid b=x\star y\wedge a\ r\ x\wedge a\ s\ y\,\}$$

Fork algebras evolved around the definition of the function $\star$. In [**HV91**] proper fork algebras were presented on a domain of binary relations on the set of finite trees built out from applications of $\star$; in that sense $\star$ acts as a set theoretical pairing function. In [**VH91**] Veloso and Haeberer moved to a definition where the domain is built from binary relations on finite strings; an immediate consequence of this decision is that $\star$ acts as string concatenation. Later on, in [**VHB92**] the base set is once again made from finite trees. In all the previously mentioned articles, no axiomatization is given. Mikulás, Sain and Simon proved, in [**MSS92**], that an extension of a proper relation algebra with $\underline{\nabla}$ (defined as in Equation 1), with $\star$ being either binary tree constructor or string concatenation is not finitely axiomatizable.

It is easy to observe that the operation $\underline{\nabla}$ induces a structure on the domain of the algebra. As it must be closed under $\underline{\nabla}$, U must be closed under applications of $\star$ thus U must be a set of the form $\langle S, \star\rangle$ for some set S.

If U is the base set of a fork algebra and $x \in U$, x is said to be a *urelement* if there are no $y, z \in U$ such that $x = y \star z$. Intuitively a urelement is a non-splitting element of U. It is easy to prove that having urelements is equivalent to having a non-surjective function $\star$.

DEFINITION 2.34. A *star proper fork algebra with urelements* is a two-sorted algebraic structure $\langle A, U, \cup, \cap, ^{-}, \emptyset, E, \circ, ^{\smile}, Id, \underline{\nabla}, \star\rangle$ in which A is a set of binary relations on a set U; $\cup$, $\cap$, $\circ$ and $\underline{\nabla}$ are binary operations on A; $^{-}$ and $^{\smile}$ are unary operations on A; $\emptyset$, E and Id are distinguished elements of A; and $\star$ is a binary operation on U satisfying:

- $\langle A, \cup, \cap, ^{-}, \emptyset, E, \circ, ^{\smile}, Id\rangle$ is a proper relation algebra on U,
- $\star$ is a binary function on U that is injective but not surjective on the restriction of its domain to E and
- A is closed under $\underline{\nabla}$ of binary relations, defined as follows:

$$r \underline{\nabla} s = \{\ \langle a, x \star y\rangle \in U \times U \mid a\ r\ x \wedge a\ s\ y\ \}\ .$$

If in addition, $\star$ is not restricted to be non-surjective, the algebra is referred to as a *star proper fork algebra*.

The class of all star proper fork algebras with urelements (star proper fork algebras) will be denoted as $\star$PFAU ($\star$PFA).

A graphical interpretation of the fork of binary relations is presented in Figure 2.1. Being $\star$ injective, intuitively means that $x\star y$ acts as an encoding of the pair $\langle x, y\rangle$ that not necessarily means its set theoretical (canonical) representation as $\{x, \{x, y\}\}$.

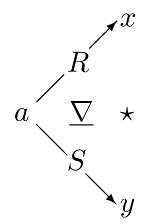


FIGURE 2.1. Graphical representation of “fork”.

DEFINITION 2.35. The class of *proper fork algebras with urelements* is obtained from $\star$PFAU as $\mathbf{Rd}_T\ \star$PFAU, where $T = \langle A, \cup, \cap, ^{-}, \emptyset, E, \circ, ^{\smile}, Id, \underline{\nabla}\rangle$.

The class of *proper fork algebras* is obtained in the same way but applying the operator $\mathbf{Rd}_T$ on the class $\star$PFA.

DEFINITION 2.36. A proper fork algebra with urelements (proper fork algebra) $\langle A, \cup, \cap, ^{-}, \emptyset, E, \circ, ^{\smile}, Id, \underline{\nabla}\rangle$ is said to be *full* if its relational reduct (which is a proper relation algebra) $\langle A, \cup, \cap, ^{-}, \emptyset, E, \circ, ^{\smile}, Id\rangle$ is full.

The class of full proper fork algebras with urelements (full proper fork algebras) will be denoted as fPFAU (fPFA).

The following two lemmas about full proper fork algebras will be useful in Chapter 5.

LEMMA 2.4. *Let $\mathfrak{A}, \mathfrak{B} \in$ fPFA, such that both $\mathfrak{A}$ and $\mathfrak{B}$ are full, $\mathfrak{B}$ is non-trivial and $\delta : \mathfrak{A} \to \mathfrak{B}$ be a homomorphism.*

Then, for all $\langle s_a, s'_a\rangle \in U^{\mathfrak{A}} \times U^{\mathfrak{A}}$, there exists $\langle s_b, s'_b\rangle \in U^{\mathfrak{B}} \times U^{\mathfrak{B}}$ such that $\delta(\{\,\langle s_a, s'_a\rangle\,\}) = \{\,\langle s_b, s'_b\rangle\,\}$.

PROOF. By definition of $^{\hat{=}}$, $\delta(\{\,\langle s_a, s'_a\rangle\,\}) = \delta(\{\,\langle s_a, s'_a\rangle\,\}^{\hat{=}})$. Now as δ is a homomorphism, $\delta(\{\,\langle s_a, s'_a\rangle\,\}^{\hat{=}}) = \delta(\{\,\langle s_a, s'_a\rangle\,\})^{\hat{=}}$. Then, by Lemma 2.1, $\delta(\{\,\langle s_a, s'_a\rangle\,\}) \neq \emptyset$, and consequently $\delta(\{\,\langle s_a, s'_a\rangle\,\})$ has exactly one pair, and as δ is defined for any relation in $U^{\mathfrak{A}} \times U^{\mathfrak{A}}$ then, there must be a pair $\langle s_b, s'_b\rangle \in U^{\mathfrak{B}} \times U^{\mathfrak{B}}$ for which $\delta(\{\,\langle s_a, s'_a\rangle\,\}) = \{\,\langle s_b, s'_b\rangle\,\}$. ∎

LEMMA 2.5. *Let $\mathfrak{A}, \mathfrak{B} \in$ fPFA, such that both $\mathfrak{A}$ and $\mathfrak{B}$ are full, $\mathfrak{B}$ is non-trivial and $\delta : \mathfrak{A} \to \mathfrak{B}$ be a homomorphism.*

Then, for all $\langle s, s'\rangle \in U^{\mathfrak{A}} \times U^{\mathfrak{A}}$, $\delta(\{\,\langle s, s'\rangle\,\}) \subseteq \delta(R)$ if and only if $\{\,\langle s, s'\rangle\,\} \subseteq R$.

PROOF. The fact that if $\{\,\langle s, s'\rangle\,\} \subseteq R$, then $\delta(\{\,\langle s, s'\rangle\,\}) \subseteq \delta(R)$ follows by observing that δ is a Boolean homomorphism, and therefore order-preserving.

Now we prove the converse property. $\delta(R) = \delta(\bigcup_{\langle a,b\rangle \in R} \{\,\langle a, b\rangle\,\}) = \bigcup_{\langle a,b\rangle \in R} \delta(\{\,\langle a, b\rangle\,\})$. Now, if $\delta(\{\,\langle s, s'\rangle\,\}) \subseteq \delta(R)$, by Lemma 2.4, there exists $\langle a, b\rangle \in R$ such that $\delta(\{\,\langle a, b\rangle\,\}) = \delta(\{\,\langle s, s'\rangle\,\})$. Then, as δ is an injection (by Lemma 2.3),$\langle a, b\rangle = \langle s, s'\rangle$ and consequently $\{\,\langle s, s'\rangle\,\} \subseteq R$. ∎

In the same way CR is the abstract counterpart of PRA (even considering its limitations), it is possible to define a calculus for proper fork algebras with urelements (CFAU for short).

DEFINITION 2.37. Let $\mathcal{R}$ be a set of relation variables, then the set of relation designations is the smallest set $\mathit{RelDes}(\mathcal{R})$ such that:

- $\mathcal{R} \cup \{\,1, 0, 1'\,\} \subseteq \mathit{RelDes}(\mathcal{R})$,
- If $r, s \in \mathit{RelDes}(\mathcal{R})$, then $\{\,r+s, r\cdot s, \overline{r}, r;s, \breve{r}, r \nabla s\,\} \subseteq \mathit{RelDes}(\mathcal{R})$.

Then, the set of formulas of CFAU is the smallest set $\mathit{CFAUForm}(\mathcal{R})$

- If $r, s \in \mathit{RelDes}(\mathcal{R})$, then $r = s \in \mathit{CFAUForm}(\mathcal{R})$,

DEFINITION 2.38. Let $\mathcal{R}$ be a set of relation variables, then CFAU is defined as follows:

- Formulas: $\mathit{CFAUForm}(\mathcal{R})$,
- Axioms:
  - Axioms (1) – (15) of Definition 2.32 for the relational operators
  - The following axioms for the fork operator: for all $r, s, t, u \in A$

$$r \nabla s = (r;(1' \nabla 1)) \cdot (s;(1 \nabla 1')) \tag{Ax. 16}$$

$$(r \nabla s);(t \nabla u)^{\smile} = (r;\breve{t}) \cdot (s;\breve{u}) \tag{Ax. 17}$$

$$(1' \nabla 1)^{\smile} \nabla (1 \nabla 1')^{\smile} \leq 1' \tag{Ax. 18}$$

$$1;(\overline{1\nabla 1}\cdot 1');1 = 1 \qquad \text{(Ax. 19)}$$

- Inference rules: the inference rules presented in Definition 2.30 for $=$.

The term $\overline{1\nabla 1}\cdot 1'$, appearing in Axiom (19), characterizes the partial identity on urelements. This term will be denoted by $1'_{\mathsf{U}}$. Then, it is easy to see that Axiom (19) enforces the existence of urelements.

If we remove the Axiom (19) we obtain what is called the *calculus for fork algebras* (CFA for short).

DEFINITION 2.39. The class of *fork algebras with urelements* (FAU for short), is the class of the models of the identities provable in CFAU.

The class of *fork algebras* (FA for short), is the class of the models of the identities provable in CFA.

Terms $(1'\nabla 1)^{\smile}$ and $(1\nabla 1')^{\smile}$, when interpreted in a proper fork algebra, act as projections of the first and second element involved in an application of $\star$. These two terms will be denoted by π and ρ, respectively. Figures 2.2 and 2.3 show a graphical representation of projections π and ρ.

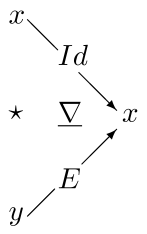


FIGURE 2.2. Graphical representation of "π".

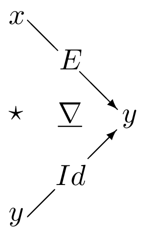


FIGURE 2.3. Graphical representation of "ρ".

These definitions allow us to rewrite Axioms (16), (18) and (19) as follows:

$$r\nabla s = (r;\breve{\pi})\cdot(s;\breve{\rho})$$

$$\pi\nabla\rho \leq 1'$$

$$1;1'_{\mathsf{U}};1 = 1$$

By resorting to the identity between non-splitting elements we define ${}_{\mathsf{U}}1_{\mathsf{U}} = 1'_{\mathsf{U}};1;1'_{\mathsf{U}}$. Relation ${}_{\mathsf{U}}1_{\mathsf{U}}$ relates every pair of non-splitting objects. Now, from the fork operator we define the binary operator $\otimes$ (*cross*) by the condition:

$$R \otimes S = (\pi;R) \nabla (\rho;S) \ . \tag{2}$$

When interpreted in an algebra $\mathfrak{A} \in \mathsf{fPFAU}$, $\otimes$ behaves as a parallel product (see Figure 2.4 for a graphical representation):

$$R \otimes S = \{ \langle a \star b, c \star d \rangle \mid \langle a, c \rangle \in R \wedge \langle b, d \rangle \in S \} .$$

$$\begin{tikzcd}
x \arrow[r, dash] & R \arrow[r] & w \in R(x) \\
\star & \otimes & \star \\
y \arrow[r, dash] & S \arrow[r] & z \in S(y)
\end{tikzcd}$$

FIGURE 2.4. Graphical representation of "$\otimes$".

As Frias points out in [**Fri02**], it is in [**HBS93b**][6] where Haeberer et al. presented the actual axiomatization for the fork algebras.

If specifications are to be written in the language of fork algebras, it is valuable to have an understandable semantics for these specifications allowing us to use our intuition on the operators, terms and equations.

Proper fork algebras with urelements are a good candidate to cope with this because its universe is made of binary relations and the operators have a simple set theoretical meaning. Then, it is important to determine the relationship between the class FAU and PFAU.

In [**FBHV95**], Frias et al. prove that FA is representable in PFA, but resorting to a non-equational axiom. Later on, in [**FHV95**], the same representability result was proved but only resorting to those equational axioms appearing in [**HBS93b**].

THEOREM 2.11. ([**FHV95**, Theorem 3.7])
$\mathsf{FA} = \mathbf{I}\ \mathsf{PFA}$ ■

The same representability result was obtained independently by Gyuris and presented in [**Gyu97**].

The proof of Theorem 2.11, published also in [**Fri02**, §4.1], can be easily adapted to a proof of the representability of FAU in PFAU.

COROLLARY 2.2. $\mathsf{FAU} = \mathbf{I}\ \mathsf{PFAU}$. ■

Even though PFAU is a good candidate as a semantics for CFAU (see Corollary 2.2), we will use the class fPFAU. To prove several theorem to come in this chapter we will need to explicitly construct proper fork algebras and some of these proofs relay on the fact that the algebras constructed are simple (see Definition 2.17). This decision is supported by the fact that the

[6]See also [**HBS93a**].

variety generated by fPFAU is the same as the variety generated by FAU, to be proved next.

DEFINITION 2.40. Let $\Gamma \subseteq \mathit{CFAUForm}$, $\phi \in \mathit{CFAUForm}$, then

- $\Phi_\Gamma = \{\, \phi \in \mathit{CFAUForm} \mid \Gamma \vdash_{\mathsf{CFAU}} \phi \,\}$,
- FAU^Γ is the class of fork algebras with urelements that are models of Φ_Γ. These algebras have the same operations as the structures in fPFAU, but their domain does not need to be made of binary relations. We only know that they satisfy the equations in Φ_Γ,
- $\mathcal{V}(\mathsf{fPFAU}, \Gamma)$ is the variety generated by those algebras $\mathfrak{A} \in \mathsf{fPFAU}$ such that $\mathfrak{A} \models_{\mathsf{fPFAU}} \gamma$, for all $\gamma \in \Gamma$. We will denote it by $\mathcal{V}_\Gamma$,
- We denote by $\models_{\mathcal{V}_\Gamma}$ the satisfiability relation on algebras from $\mathcal{V}_\Gamma$. As it is standard, $\Theta \models_{\mathcal{V}_\Gamma} \phi$ if and only if for all $\mathfrak{A} \in \mathcal{V}_\Gamma$, if $\mathfrak{A} \models_{\mathcal{V}_\Gamma} \theta$ (for all $\theta \in \Theta$), then $\mathfrak{A} \models_{\mathcal{V}_\Gamma} \phi$,
- We denote by $\vdash_=$ the entailment relation of equational logic.

LEMMA 2.6. ([**Fri02**, Theorem 6.2])
$\mathsf{FAU}^\Gamma = \mathcal{V}_\Gamma$.

PROOF. Let $\mathfrak{A} \in \mathsf{FAU}^\Gamma$. Then, $\mathfrak{A} \in \mathsf{FAU}^\emptyset$. By Corollary 2.2, there is $\mathfrak{B} \in \mathcal{V}_\emptyset$ such that $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$. Since $\mathcal{V}_\emptyset$ is closed under isomorphisms, also $\mathfrak{A} \in \mathcal{V}_\emptyset$. Since $\mathfrak{A} \models \Gamma$, $\mathfrak{A} \in \mathcal{V}_\Gamma$.

The other inclusion follows easily once we notice that the axioms of CFAU all hold in fPFAU. ∎

COROLLARY 2.3. *The equational theories of $\mathcal{V}_\Gamma$ and FAU^Γ (denoted by $EqTh(\mathcal{V}_\Gamma)$ and $EqTh(\mathsf{FAU}^\Gamma)$, respectively), coincide.* ∎

Then, it is possible to state the following theorem.

THEOREM 2.12. ([**Fri02**, Theorem 5.5])
Let α be a FAU equation. Then

$$\models_{\mathsf{fPFAU}} \alpha \quad \Longleftrightarrow \quad \vdash_{\mathsf{CFAU}} \alpha \; .$$

PROOF. The inclusion $\vdash_{\mathsf{CFAU}} \subseteq \models_{\mathsf{fPFAU}}$ follows from the soundness of the calculus CFAU, and can be proved by induction on the structure of proofs.

In order to prove the inclusion $\models_{\mathsf{fPFAU}} \subseteq \vdash_{\mathsf{CFAU}}$, we reason as follows.

$$\begin{array}{ll}
 & \Gamma \models_{\mathsf{fPFAU}} \alpha \\
\Longrightarrow & \forall \mathfrak{A} \in \mathsf{fPFAU}\ (\mathfrak{A} \models_{\mathsf{fPFAU}} \Gamma \Rightarrow \mathfrak{A} \models_{\mathsf{fPFAU}} \alpha) \\
 & \qquad \text{[by Definition } \models_{\mathsf{fPFAU}}] \\
\Longrightarrow & \alpha \in EqTh(\mathcal{V}(\mathsf{fPFAU}, \Gamma)) \\
 & \qquad \text{[by Definition } \mathcal{V}] \\
\Longrightarrow & \alpha \in EqTh(\mathsf{FAU}^{\Gamma}) \\
 & \qquad \text{[by Corollary 2.3]} \\
\Longrightarrow & \forall \mathfrak{A} \in \mathsf{FAU}^{\Gamma}, \mathfrak{A} \models_{\mathcal{V}_{\Gamma}} \alpha \\
 & \qquad \text{[by Definition } EqTh] \\
\Longrightarrow & \forall \mathfrak{A} \in Models(\Phi_{\Gamma}), \mathfrak{A} \models_{\mathcal{V}_{\Gamma}} \alpha \\
 & \qquad \text{[by Definition } \mathsf{FAU}^{\Gamma}] \\
\Longrightarrow & \Phi_{\Gamma} \models_{\mathcal{V}_{\Gamma}} \alpha \\
 & \qquad \text{[by Definition } \models_{\mathcal{V}_{\Gamma}}] \\
\Longrightarrow & \Phi_{\Gamma} \vdash_{=} \alpha \\
 & \qquad \text{[by equational logic completeness]} \\
\Longrightarrow & \alpha \in \Phi_{\Gamma} \\
 & \qquad \text{[because } \Phi_{\Gamma} \text{ is a closed theory]} \\
\Longrightarrow & \Gamma \vdash_{\mathsf{CFAU}} \alpha \\
 & \qquad \text{[by Definition } \Phi_{\Gamma}]
\end{array}$$

■

Fork algebras were used to interpret several logics. The approach consists on building a relational algebraization of a logic $\mathcal{L}$. This is done by resorting to a semantics preserving mapping $T_{\mathcal{L}} : \mathit{Formulas}_{\mathcal{L}} \to \mathit{RelDes}(X)$ for some set of relational variables X, translating $\mathcal{L}$-formulas to relational terms.

Let $\Gamma \cup \{\alpha\} \subseteq \mathit{Formulas}_{\mathcal{L}}$, then the property of being semantics preserving is characterized by the following condition:

$$\Gamma \models_{\mathcal{L}} \alpha \iff \{\, T_{\mathcal{L}}(\gamma) = 1 \mid \gamma \in \Gamma \,\} \vdash_{\mathsf{CFAU}} T_{\mathcal{L}}(\alpha) = 1,$$

In [**Fri02**] Frias presented the interpretability result of FOL in an extension of FAU with relational constants representing logic constants, functions and predicates. In [**FO98**], it was proved an interpretability result of PDL in FAU, but this time extending the language of FAU with a set of constants representing the atomic programs. Then, in [**FBM02**], it was presented an interpretability result for FODL by resorting to a mixing of the techniques used in [**FO98**] for FOL and PDL. In [**FL03, FL06**], temporal logics LTL and TL and their first-order versions were respectively interpreted in FAU.

More precisely, most of these interpretability results were proved by resorting to a more complex algebraic framework called *ω-closure fork algebras with urelements* (ω-CFAU for short).

Definition 2.41. A *star proper closure fork algebra with urelements* is a two-sorted algebraic structure $\langle A, U, \cup, \cap, ^-, \emptyset, E, \circ, ^{\smile}, Id, \underline{\nabla}, \star, ^{\hat{*}}, ^{\underline{*}} \rangle$ in which A is a set of binary relations on U; $\cup$, $\cap$, $\circ$ and $\underline{\nabla}$ are binary operations on A; $^-$, $^{\smile}$, $^{\hat{*}}$ and $^{\underline{*}}$ are unary operations on A; $\emptyset$, E and Id are distinguished elements of A; and $\star$ is a binary operation on U, satisfying:

- $\langle A, U, \cup, \cap, \circ, \emptyset, E, \circ, ^{\smile}, Id, \underline{\nabla}, \star \rangle$ is a *star proper fork algebra with urelements*,
- A is closed under $\underline{^\diamond}$ and $\underline{^*}$ of binary relations, defined as follows[7]:

$$r^{\underline{\diamond}} \subseteq r \wedge (|r^{\underline{\diamond}}| = 1 \iff r \neq \emptyset)\,, \tag{3}$$

$$r^{\underline{*}} = \bigcup_{0 \leq i} r^{;i}. \tag{4}$$

The operators $\underline{^\diamond}$ and $\underline{^*}$ are interpreted in a star proper closure fork algebra as non-deterministic choice of a single pair, and reflexive-transitive closure of the binary relation to which they are applied, respectively. Formulas (3) and (4) formalize their meaning.

The class of all star proper closure fork algebra with urelements will be denoted $\star$PCFAU.

It is now possible to define the class of *proper closure fork algebra with urelements* (PCFAU for short) as follows.

Definition 2.42. The class of the *proper closure fork algebras with urelements* (denoted as PCFAU) is obtained from $\star$PCFAU as $\mathbf{Rd}_T$ $\star$PCFAU, where $T = \langle A, \cup, \cap, ^-, \emptyset, E, \circ, ^{\smile}, Id, \underline{\nabla}, \underline{^\diamond}, \underline{^*} \rangle$.

Definition 2.43. Let $\langle A, \cup, \cap, ^-, \emptyset, E, \circ, ^{\smile}, Id, \underline{\nabla}, \underline{^\diamond}, \underline{^*} \rangle$ be a proper closure fork algebra with urelements; it is said to be *full* if its relational reduct (which is a proper relation algebra) $\langle A, \cup, \cap, ^-, \emptyset, E, \circ, ^{\smile}, Id \rangle$ is full.

The class of full proper closure fork algebras with urelements will be denoted as fPCFAU.

Lemma 2.7. *Let* $\mathfrak{A} \in$ fPCFAU, *then* $\mathfrak{A}$ *is atomistic.*

Proof. The proof follows by observing that the set of all singleton relations is a complete atomic basis. ■

Once again, it is possible to define the abstract counterpart for PCFAU in order to obtain a calculus for this class of algebras. This calculus is called calculus for closure fork algebras (CCFAU for short).

Definition 2.44. Let $\mathcal{R}$ be a set of relation variables, then the set of relation designations is the smallest set $RelDes(\mathcal{R})$ such that:

- $\mathcal{R} \cup \{1, 0, 1'\} \subseteq RelDes(\mathcal{R})$,
- If $r, s \in RelDes(\mathcal{R})$, then $\{r+s,\ r{\cdot}s,\ \overline{r},\ r;s,\ \breve{r},\ r\nabla s,\ r^\diamond,\ r^*\} \subseteq RelDes(\mathcal{R})$.

Then, the set of formulas of CCFAU is the smallest set $CCFAUForm(\mathcal{R})$

- If $r, s \in RelDes(\mathcal{R})$, then $r = s \in CCFAUForm(\mathcal{R})$,

Definition 2.45. Let $\mathcal{R}$ be a set of relation variables, then CCFAU is defined as follows:

[7]The $i^{th.}$ folded iteration of a relation r, denoted $r^{;i}$, is defined by the following two conditions:

$$\begin{aligned} r^{;0} &= 1', \\ r^{;n+1} &= r;r^{;n}. \end{aligned}$$

- Formulas: *CCFAUForm*$(\mathcal{R})$,
- Axioms:
  - Axioms (1) – (19) of Definition 2.38 for the fork algebra operators
  - The following axioms for the closure and choice operators: for all $r, s \in A$

$$r^{\diamond};1;\breve{r^{\diamond}} \leq 1' \tag{Ax. 20}$$

$$\breve{r^{\diamond}};1;r^{\diamond} \leq 1' \tag{Ax. 21}$$

$$1;(r\cdot r^{\diamond});1 = 1;r;1 \tag{Ax. 22}$$

$$r^{*} = 1'+r;r^{*} \tag{Ax. 23}$$

$$r^{*};s \leq s+r^{*};(\overline{s}\cdot r;s) \tag{Ax. 24}$$

- Inference rules: the inference rules presented in Definition 2.30 for $=$.

The Axioms (20) – (22) characterizing the choice operators $^{\diamond}$ where presented by Maddux in [**Mad89**], and the Axioms (23) – (24) characterizing the operator * where introduced by Baum et al. in [**BFM98**].

Definition 2.46. The class of all *closure fork algebras with urelements*, (CFAU for short) is the class of the models of the identities provable in CCFAU.

In order to completely characterize *, it is possible to define the *ω-calculus of closure fork algebras with urelements* (ω-CCFAU for short) as an extension of CCFAU by the addition of a new inference rule.

Definition 2.47. Let $\mathcal{R}$ be a set of relation variables, then ω-CCFAU is defined as follows:

- Formulas: *CCFAUForm*$(\mathcal{R})$,
- Axioms: the Axioms (1) – (24) of Definition 2.45 for the closure fork algebra operators,
- Inference rules:
  - The inference rules presented for $=$ in Definition 2.30,
  - ω-rule:

$$\frac{\vdash 1' \leq s \qquad r^{;i} \leq s \vdash r^{;i+1} \leq s \quad (\forall i \in \mathbb{N})}{\vdash r^{*} \leq s}$$

Definition 2.48. The class of all *ω-closure fork algebras with urelements* (ω-CFAU for short) is the class of the models of the identities provable in ω-CCFAU.

ω-CFAU are the only models of ω-CCFAU but in [**FBM02**] was proved that PCFAU can play this rôle as a consequence of the following two theorems which state the relationship between the classes PCFAU and ω-CFAU.

Theorem 2.13. PCFAU $\subseteq$ ω-CFAU.

PROOF. The proof of this theorem follows trivially by observing that every proper closure fork algebra with urelements is a model of ω-CCFAU. ■

THEOREM 2.14. ([**FBM02**, Theorem 1])
ω-CFAU is representable in PCFAU. ■

Once again, even when the natural candidate to play the rôle of models of ω-CCFAU is the class of algebras PCFAU, we prove that ω-CCFAU is complete for the class fPCFAU. This result allows us to consider the class fPCFAU when assigning semantics to ω-CCFAU.

THEOREM 2.15. ([**LF06**, Theorem 4])
Let α be a ω-CFAU-equation. Then

$$\models_{\mathsf{fPCFAU}} \alpha \iff \vdash_{\omega\text{-}\mathsf{CCFAU}} \alpha \ .$$

■

We now have precisely defined the language of full proper closure fork algebras with urelements, which is intended to be the formalism we will use in order to write (and understand) software specifications; and ω-CCFAU which will be the tool used to prove the properties of specifications. Software specifications will be theories in which the set of axioms will be formed by all the axioms presented in Definition 2.47 and a set of formulas exhibiting software's known behavior.

Notice that only extralogical symbols belong to an equational or first-order signature. Symbols such as $=$ in equational logic, or $\vee$ in first-order logic, have a meaning that is univocally determined by the carriers and the interpretation of the extralogical symbols. Similarly, once the field has been fixed, all the operators can be assigned a standard meaning. This gives rise to the following definition of fPCFAU-signatures[8].

From now on we will assume a fixed but arbitrary denumerable set of relation variables $\mathcal{R}$ over which formulas are written. We will also assume the existence of a total order $< \subseteq \mathcal{R} \times \mathcal{R}$, then if $i \in \mathbb{N}$, $\mathcal{R}_i$ will denote the $i^{th.}$ element of $\mathcal{R}$ under the ordering $<$, and if r is an element of $\mathcal{R}$, $\mathcal{R}_r$ will be the position of r in the ordering $<$ of $\mathcal{R}$.

DEFINITION 2.49. An fPCFAU-signature is a structure $\langle \{f_i\}_{i \in \mathcal{I}} \rangle$. Each function symbol f_i, for all $i \in \mathcal{I}$, comes equipped with its arity. Notice that since full proper closure fork algebras with urelements have only one sort, the arity is just a natural number.

The set of fPCFAU-signatures will be denoted as $Sign_{\mathsf{fPCFAU}}$.

In order to interpret the logics mentioned in previous sections, constant relational symbols (rather than functions in general) suffice. Since new operators may be necessary in order to interpret new logics in the future, signatures will be allowed to contain functions of arbitrary rank.

[8]To be precise, CCFAU-signatures or ω-CCFAU-signatures should be used because signatures are more related to the calculus than to its semantics. But, considering the way they will be used throughout the rest of the thesis and Theorem 2.15, we prefer the use of fPCFAU-signatures instead.

In order to extend the definitions of terms and formulas of the closure fork algebras (Definition 2.44) to fPCFAU-signatures, we only need to add the following rule.

- If $t_1, \ldots, t_{arity(f_i)} \in \mathit{RelDes}(\mathcal{R})$, then

$$f_i(t_1, \ldots, t_{arity(f_i)}) \in \mathit{RelDes}(\mathcal{R}) \text{ (for all } i \in \mathcal{I}).$$

If $\Sigma \in \mathit{Sign}_{\mathsf{fPCFAU}}$, the set of Σ-terms will be denoted as $Term_\Sigma$. In the same way, Sen_Σ will denote the set of equalities between Σ-terms (i.e. the set of Σ-formulas).

Definition 2.50. Let $\Sigma = \langle \{f_i\}_{i\in\mathcal{I}} \rangle \in \mathit{Sign}_{\mathsf{fPCFAU}}$, then $\langle \mathcal{P}, \{\underline{f_i}\}_{i\in\mathcal{I}} \rangle \in Mod_\Sigma$ if and only if:

- $\mathcal{P} \in \mathsf{fPCFAU}$,
- $\underline{f_i} : |\mathcal{P}|^{arity(f_i)} \to |\mathcal{P}|$, for all $i \in \mathcal{I}$.

Definition 2.51. Let $\Sigma = \langle \{f_i\}_{i\in\mathcal{I}} \rangle \in \mathit{Sign}_{\mathsf{fPCFAU}}$, $\mathcal{M} = \langle \mathcal{P}, \{\underline{f_i}\}_{i\in\mathcal{I}} \rangle \in Mod_\Sigma$ and $val : \mathcal{R} \to |\mathcal{P}|$, then $m^{val}_{\mathcal{M}} : Term_\Sigma \to |\mathcal{P}|$ is defined as follows:

- $m^{val}_{\mathcal{M}}(\star) = \star^{\mathcal{P}}$, for all $\star \in \{1, 0, 1'\}$,
- $m^{val}_{\mathcal{M}}(r) = val(r)$, for all $r \in \mathcal{R}$.
- if $\{t, t_1, t_2\} \subseteq Term_\Sigma$, then
  - $m^{val}_{\mathcal{M}}(t^\star) = m^{val}_{\mathcal{M}}(t)^{\star^{\mathcal{P}}}$, for all $\star \in \{^-, ^\smile, ^\diamond, ^*\}$,
  - $m^{val}_{\mathcal{M}}(t_1 \star t_2) = m^{val}_{\mathcal{M}}(t_1) \star^{\mathcal{P}} m^{val}_{\mathcal{M}}(t_1)$, for all $\star \in \{+, \cdot, ;, \nabla\}$,
- if $\{t_1, \ldots, t_n\} \subseteq Term_\Sigma$, then
  $m^{val}_{\mathcal{M}}(f_i(t_1, \ldots, t_{arity(f_i)})) = \underline{f_i}(m^{val}_{\mathcal{M}}(t_1), \ldots, m^{val}_{\mathcal{M}}(t_{arity(f_i)}))$, for all $i \in \mathcal{I}$.

Definition 2.52. Let $\langle \{f_i\}_{i\in\mathcal{I}} \rangle \in \mathit{Sign}_{\mathsf{fPCFAU}}$ and $\langle \mathcal{P}, \{\underline{f_i}\}_{i\in\mathcal{I}} \rangle \in Mod_\Sigma$ be denoted by Σ and $\mathcal{M}$ respectively, then $\models^{\Sigma}_{\mathsf{fPCFAU}} \subseteq Mod_\Sigma \times Sen_\Sigma$ is defined as follows:

$$\mathcal{M} \models^{\Sigma}_{\mathsf{fPCFAU}} t_1 = t_2 \text{ iff for all } val : \mathcal{R} \to |\mathcal{P}|,\ m^{val}_{\mathcal{M}}(t_1) = m^{val}_{\mathcal{M}}(t_2)\ .$$

CHAPTER 3

Category theory and general logic

In this chapter we introduce those concepts coming from category theory, and particularly from general logics, that will be used throughout the rest of this work. This chapter does not pretend to be an exhaustive presentation of category theory or general logics so we point the interested reader to [**Fia05**] for a gentle introduction to category theory for computer scientists, to [**McL71**] for an introduction to category theory for mathematicians, and to [**Mes89, Tar96**] for a complete presentation of general logics.

The chapter is organized as follows. In §1 we provide the basic definitions and results from the field of category theory we will use throughout the rest of the thesis. In §2 we do the same with the basic definitions and results from the field of general logics, introducing the notions of institution, entailment system, logic, etc. And in §3 we present the more complex constructions used in the field of general logics in order to relate different institutions.

1. Category theory

From here on, we assume the reader has a nodding acquaintance with basic concepts from category theory such as the notions of category, functor, natural transformation and co-limit. This section only summarizes these definitions in order to fix the notation.

Definition 3.1. (Graph)
A *graph* is a structure $\langle G_0, G_1\rangle$ where

- G_0 is a collection (of nodes),
- G_1 is a collection (of arrows),
- $src : G_1 \to G_0$ maps every arrow to a node (the source of the arrow),
- $trg : G_1 \to G_0$ maps every arrow to a node (the target of the arrow).

If $f \in G_1$ is an arrow from x to y, it will be denoted as $f : x \to y$ or $x \xrightarrow{f} y$.

Definition 3.2. (Graph homomorphism)
Let G, and H be graphs. A *homomorphism* of graphs $\phi : G \to H$ is a

pair of maps $\phi_0 : G_0 \to H_0$ and $\phi_1 : G_1 \to H_1$ such that for each arrow $f : x \to y \in G_1$, we have $\phi_1(f) : \phi_0(x) \to \phi_0(y) \in H_1$.

DEFINITION 3.3. (Path in a graph)
Let $G = \langle G_0, G_1 \rangle$ be a graph, and $x, y \in G_0$. A *path* from x to y of length $k > 0$ is a sequence $f_1, \ldots, f_k$ such that:

- $f_i \in G_1$ $(1 \leq i \leq k)$,
- $src(f_1) = x$,
- $trg(f_i) = src(f_{i+1})$ $(1 < i < k)$,
- $trg(f_k) = y$.

We will denote by G_i the collection of paths of length i.

DEFINITION 3.4. (Category)
A *category* is a structure $\langle G, \circ, id \rangle$ where:

- $G = \langle G_0, G_1 \rangle$ is a graph,
- $\circ : G_2 \to G_1$ is a map from G_2 into G_1 (called composition). If $f, g \in G_1$, then the composition of f and g is denoted[1] as $f \circ g : x \to z$, and
- $id : G_0 \to G_1$ is a map from G_0 into G_1 (called identity). If $x \in G_0$, $id_x : x \to x$ is the identity arrow for x,

such that for all $x, y \in G_0$ and $f, g, h \in G_1$:

- if $f : x \to y \in G_0$, then $f = id_x \circ f = f \circ id_y$,
- if $src(g) = trg(f)$ and $src(h) = trg(g)$, then $(f \circ g) \circ h = f \circ (g \circ h)$.

If C is a category, by $graph(\mathsf{C})$ we denote the graph of C.

Elements in a category are called *objects* and arrows are referred to as *morphisms*.

DEFINITION 3.5. (The category Set)
$\mathsf{Set} = \langle G, \circ, id \rangle$ such that:

- $G = \langle G_0, G_1 \rangle$, where G_0 is the collection of all sets and G_1 is the collection of all total functions between sets,
- if $S, S', S'' \in G_0$ and $f : S \to S', f' : S' \to S'' \in G_1$, then the composition of f and f', $f \circ f' : S \to S''$ is defined as $f \circ f'(s) = f'(f(s))$ for all $s \in S$,
- if $S \in G_0$, then $id_S : S \to S$ is the identity on S.

Instead of following the formal definition of a category, we will usually define them just by declaring what are their objects and morphisms, and by defining their composition and identities, omitting any mention to its graph. Thus, we will present a category as a structure $\langle \mathcal{O}, \mathcal{A} \rangle$ where $\mathcal{O}$ is the collection of objects and $\mathcal{A}$ is the collection of morphisms together with an appropriate definition of $id_x \in \mathcal{A}$ for each object $x \in \mathcal{O}$ and $\circ : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$. Given a category C, the collection of objects of C, will be denoted $|\mathsf{C}|$.

Sometimes categories have distinguished objects with particular behaviors which can be identified by the way they are related to other objects in the category. Among these we single out the *initial* and *terminal* objects.

[1]We use the notation $f \circ g$ meaning the sequential (or diagrammatic) composition of f and g. This operator is often noted in the literature as ";", but since ";" has been widely used in the context of relation algebra, we will adopt "$\circ$" instead.

Definition 3.6. (Initial object)
Let C be a category and $x \in |\mathsf{C}|$. Then, x is *initial* if and only if for all $y \in |\mathsf{C}|$, there exists a unique morphism $f : x \to y$.

Analogously, x is *terminal* if and only if for all $y \in |\mathsf{C}|$, there exists a unique morphism $f : y \to x$.

These distinguished elements have some interesting properties. The next proposition shows a very useful property of initial objects.

Lemma 3.1. ([**Fia05**, §4.1.2])
The following two properties hold:

- *Any two initial objects are isomorphic.*
- *Any objects isomorphic to an initial object are also initial.*

■

Most of the reasoning in category theory can be carried out by observing properties in *diagrams*.

Definition 3.7. (Diagram)
Let C be a category and I a graph. A *diagram* with shape I in C is a graph homomorphism $\delta = \langle \delta_0, \delta_1 \rangle : I \to \mathit{graph}(\mathsf{C})$.

Definition 3.8. (Commutative diagrams)
Let C be a category and $I = \langle G_0, G_1 \rangle$ a graph. Then, a diagram $\delta = \langle \delta_0, \delta_1 \rangle : I \to \mathit{graph}(\mathsf{C})$ commutes if and only if for $f_1, \ldots, f_i \in G_i$, $g_1, \ldots, g_j \in G_j$ such that $\mathit{src}(f_1) = \mathit{src}(g_1)$ and $\mathit{trg}(f_i) = \mathit{trg}(g_j)$, then

$$\delta_1(f_1) \circ \cdots \circ \delta_1(f_i) = \delta_1(g_1) \circ \cdots \circ \delta_1(g_j)$$

holds in C.

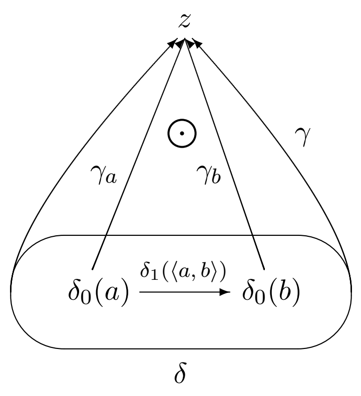
Diagrams will be usually presented by their graphical representation resorting to their image on the category instead of using the previous definitions. Commutative diagrams will be distinguished by decorating them with the symbol $\odot$ inside.

Given a diagram, it is possible to identify the collective behavior of the objects it involves. To capture this collective behavior we introduce the notion of *(co)cones* and *(co)limits*.

Definition 3.9. (Co-cone)
Let C be a category, $I = \langle G_0, G_1 \rangle$ be a graph, and $\delta = \langle \delta_0, \delta_1 \rangle$ be a diagram $\delta : I \to \mathit{graph}(\mathsf{C})$. A *co-cone* with base δ is an object $z \in |\mathsf{C}|$ (the vertex of the co-cone), and a family of morphisms $\{\gamma_{\delta_0(a)} : \delta_0(a) \to z\}_{a \in \delta_0(G_0)}$ (the edges of the co-cone), usually denoted $\gamma : \delta \to z$.

A co-cone is *commutative* if and only if for any pair of vertexes $a, b \in G_0$ such that $\langle a, b \rangle \in G_1$, $\delta_1(\langle a, b \rangle) \circ \gamma_b = \gamma_a$.

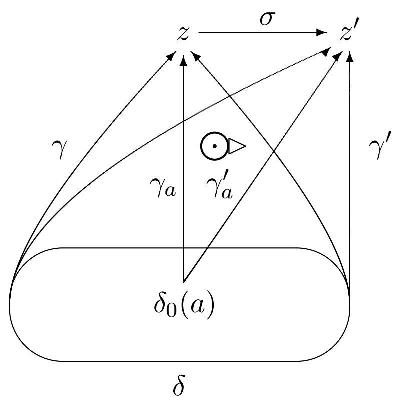
The concept of *cone* is the dual notion of co-cone (i.e. the equivalent definition but reversing the arrows).



Definition 3.10. (Co-limit)
Let C be a category, $I = \langle G_0, G_1\rangle$ be a graph, and $\delta = \langle \delta_0, \delta_1\rangle$ be a diagram $\delta : I \to graph(\mathsf{C})$. A *co-limit* is a commutative co-cone $\gamma : \delta \to z$ such that for every commutative co-cone $\gamma' : \delta \to z'$, there is a unique morphism $\sigma : z \to z'$ satisfying $\gamma \circ \sigma = \gamma'$ (i.e. for all $a \in G_0$, $\gamma_a \circ \sigma = \gamma'_a$).

The concept of *limit* is the dual notion of co-limit (i.e. the equivalent definition but reversing the arrows).

As in the case of diagrams, (co)cones and (co)limits can be presented by its graphical representation by resorting to its image on the category and decorating co-limits and limits with $\odot\!\rhd$ and $\odot\!\lhd$ respectively.



Definition 3.11. (Co-completeness)
A category is (finitely) co-complete if and only if all (finite) diagrams have co-limits.

A category is (finitely) complete if and only if all (finite) diagrams have limits.

Definition 3.12. (Pushouts)
Let C be a category and $f : x \to y$, $g : x \to z$ morphisms in C, then a *pushout* of f and g consists of $f' : y \to w$, $g' : z \to w$ morphisms in C such that:

- $f \circ f' = g \circ g'$, and

- for any other $f'' : y \to w'$, $g'' : z \to w'$ morphisms in C such that $f \circ f'' = g \circ g''$, there exists a unique $h : w \to w'$ morphism in C such that $f' \circ h = f''$ and $g' \circ h = g''$.

Pullback are defined analogously to pushouts but considering the arrows in the opposite direction.

When pushouts and pullbacks are presented in a diagrammatic way they are decorated with $\odot\!\rhd$ and $\odot\!\lhd$ respectively.

$$\begin{tikzcd}
& y \arrow[dr, "f'"] \arrow[drr, bend left, "f''"] & \odot & \\
x \arrow[ur, "f"] \arrow[dr, "g"'] & \odot\!\rhd & w \arrow[r, "h"] & w' \\
& z \arrow[ur, "g'"'] \arrow[urr, bend right, "g''"'] & \odot &
\end{tikzcd}$$

There are several results on (co)completeness of categories, one of them is the following proposition.

Lemma 3.2. ([**Fia05**, §4.4.7])
A category is finitely co-complete if and only if it has initial objects and pushouts of all pairs of morphisms with common source.

A category is finitely complete if and only if it has final objects and pullbacks of all pairs of morphisms with common source. ■

Categories can not only be constructed by giving its graph, composition operation and identity map, new categories can be built from existing ones by using some elementary operations. The advantage of constructing new categories from existing ones is that the former "inherit" properties from the latter.

We will now present some of these elementary constructions.

Definition 3.13. (Opposite, or dual, category)
Let $\mathsf{C} = \langle \mathcal{O}, \mathcal{A} \rangle$ be a category. Then C^{op} (the *opposite category* of C) is defined as $\langle \mathcal{O}, \mathcal{A}^{\mathsf{op}} \rangle$ where the morphisms in $\mathcal{A}^{\mathsf{op}}$ are the morphisms of $\mathcal{A}$ reverted (i.e. $f^{\mathsf{op}} : y \to x \in \mathcal{A}^{\mathsf{op}}$ if and only if $f : x \to y \in \mathcal{A}$) and such that for all $f, g \in G_1$ and $f \circ g \in G_2$, $(f \circ g)^{\mathsf{op}} = g^{\mathsf{op}} \circ f^{\mathsf{op}}$.

The previous definitions reflects that the direction of morphisms in a category has no essential significance because structural properties of the category are, in some sense, preserved in its opposite category; but the interpretation of the morphisms has a "natural" direction which helps the understanding of the category. Of course the interpretation of morphisms differ from one category to its opposite. If we recall the definition of Set, morphisms in $\mathsf{Set}^{\mathsf{op}}$ can not be interpreted as total functions.

An example of a property which can be proved by using this universal construction is the following proposition.

Lemma 3.3. *Let C be a category and $x \in |\mathsf{C}|$, then x is terminal if and only if x is initial in C^{op}.*

PROOF. The proof follows trivially by Definitions 3.4, 3.13 and 3.6. ∎

DEFINITION 3.14. (Product category)
Let $\mathsf{C} = \langle \mathcal{O}_\mathsf{C}, \mathcal{A}_\mathsf{C} \rangle$ and $\mathsf{D} = \langle \mathcal{O}_\mathsf{D}, \mathcal{A}_\mathsf{D} \rangle$ be categories. Then $\mathsf{C} \times \mathsf{D}$ (the *product category* of C and D) is defined as $\langle \mathcal{O}_\mathsf{C} \times \mathcal{O}_\mathsf{D}, \mathcal{A} \rangle$ where the morphisms in $\mathcal{A} \subseteq \mathcal{A}_\mathsf{C} \times \mathcal{A}_\mathsf{D}$ are such that $f_\mathsf{C} : x_\mathsf{C} \to y_\mathsf{C} \in \mathcal{A}_\mathsf{C}$ and $f_\mathsf{D} : x_\mathsf{D} \to y_\mathsf{D} \in \mathcal{A}_\mathsf{D}$ if and only if $\langle f_\mathsf{C}, f_\mathsf{D} \rangle : \langle x_\mathsf{C}, x_\mathsf{D} \rangle \to \langle y_\mathsf{C}, y_\mathsf{D} \rangle \in \mathcal{A}$.

There are many of these elementary operations which permit the construction of new categories from existing ones but will not be used in this work. The interested reader is pointed to [**McL71**] for a complete presentation of these operations. Those that are most commonly used in computer science can be found in [**Fia05**].

Another way to create a new category is to consider it as a *subcategory* of another one. This means that, given a category C, it is possible to build a new one by removing some objects and morphisms from C.

DEFINITION 3.15. (Subcategory)
Let $\mathsf{C} = \langle \mathcal{O}_\mathsf{C}, \mathcal{A}_\mathsf{C} \rangle$ and $\mathsf{D} = \langle \mathcal{O}_\mathsf{D}, \mathcal{A}_\mathsf{D} \rangle$ be categories. Then D is a *subcategory* of C if and only if:

- $\mathcal{O}_\mathsf{D} \subseteq \mathcal{O}_\mathsf{C}$,
- $\mathcal{A}_\mathsf{D} \subseteq \mathcal{A}_\mathsf{C}$ such that:
  - if $x \in \mathcal{O}_\mathsf{D}$, then $id_{\mathsf{D}x} = id_{\mathsf{C}x}$,
  - if $f : x \to y, g : y \to z \in \mathcal{A}_\mathsf{D}$, then $f \circ_\mathsf{D} g = f \circ_\mathsf{C} g$.

Whenever D is a subcategory of C, it is denoted by $\mathsf{D} \hookrightarrow \mathsf{C}$.

There are many ways in which categories relate to each other. The most common way categories relate is through *functors*.

DEFINITION 3.16. (Functor)
Let $\mathsf{C} = \langle \mathcal{O}_\mathsf{C}, \mathcal{A}_\mathsf{C} \rangle$ and $\mathsf{D} = \langle \mathcal{O}_\mathsf{D}, \mathcal{A}_\mathsf{D} \rangle$ be categories. Then $\delta : \mathsf{C} \to \mathsf{D}$ is a *functor* if and only if:

- if $x \in \mathcal{O}_\mathsf{C}$, then $\delta(x) \in \mathcal{O}_\mathsf{D}$,
- if $f : x \to y \in \mathcal{A}_\mathsf{C}$, then $\delta(f) \in \mathcal{A}_\mathsf{D}$, such that:
  - if $x \in \mathcal{O}_\mathsf{C}$, $\delta({id^\mathsf{C}}_x) = {id^\mathsf{D}}_{\delta(x)}$,
  - if $f : x \to y, g : y \to z \in \mathcal{A}_\mathsf{C}$, then $\delta(f \circ_\mathsf{C} g) = \delta(f) \circ_\mathsf{D} \delta(g)$.

It is now possible to define the category Cat, whose objects and morphisms are categories and functors respectively. This requires to prove that the identity law and composition law are preserved by functors which trivially follows from Definition 3.16.

There are certain behaviors of functors in which we will be interested. In some situations we will be interested in functors that, in some sense, relate two categories in the same way. This behavior is usually referred as "natural". This property of functors is formalized in the following definition.

DEFINITION 3.17. (Natural transformation)
Let $\mathsf{C}, \mathsf{D} \in \mathsf{Cat}$, $\psi : \mathsf{C} \to \mathsf{D}$ and $\phi : \mathsf{C} \to \mathsf{D}$ be functors, then $\tau : \psi \to \phi$ is a *natural transformation* if it is a function that assigns to each object $c \in |\mathsf{C}|$ a morphism $\tau_c : \psi(c) \to \phi(c)$ such that for each $c, c' \in |\mathsf{C}|$, $f : c \to c'$ morphism in C, the following diagram commutes:

$$\begin{tikzcd}
\psi(c) \arrow[r, "\tau_c"] \arrow[d, "\psi(f)"] & \phi(c) \arrow[d, "\phi(f)"] \\
\psi(c') \arrow[r, "\tau_{c'}"] & \phi(c')
\end{tikzcd}$$

A particular class of categories are the *monoidal categories.* A category is monoidal if it comes equipped with a product $\otimes$, which is associative, and a two-sided identity object e. We now define *strict monoidal categories.* There is a notion of (relaxed) monoidal category but it will not be used in this work so we point the interested reader to [**McL71**, Chapter VII] for an exhaustive presentation of monoidal categories.

Definition 3.18. (Strict monoidal categories)
A *strict monoidal category* is a structure $\langle \mathsf{C}, \otimes, e\rangle$ where C is a category, and the following properties hold:

- $\otimes$ is a bifunctor, thus for all $x, y, z, x', y', z' \in |\mathsf{C}|$, $f : x \to y$, $f' : x' \to y'$, $g : y \to z$, $g' : y' \to z'$ morphisms in C:
$$\begin{aligned} id_x \otimes id_y &= id_{x\otimes y} \ , \\ (f \otimes f') \circ (g \otimes g') &= (f \circ g) \otimes (f' \circ g') \ . \end{aligned}$$
- $\otimes$ is associative, thus for all $w, x, y, z \in |\mathsf{C}|$, and $f : w \to x$, $g : x \to y$, $h : y \to z$ morphisms in C:
$$\begin{aligned} x \otimes (y \otimes z) &= (x \otimes y) \otimes z \ , \\ f \otimes (g \otimes h) &= (f \otimes g) \otimes h \ . \end{aligned}$$
- e is left and right identity for $\otimes$, thus for all $x, y \in |\mathsf{C}|$, $f : x \to y$ morphism in C:
$$e \otimes x = x \otimes e = x \ ,$$
$$id_e \otimes f = f \otimes id_e = f \ .$$

Once again, we can define the category of strict monoidal categories SMCat whose objects and morphisms are strict monoidal categories and functors between them respectively.

2. Institutions and general logics

In [**GB84**], Goguen and Burstall introduced the notion of institution as a general and abstract description of the model theory of a logic. This semantic approach to the description of a logic was then followed by a proof-theoretic approach due to Meseguer [**Mes89**], and Fiadeiro and A. Sernadas [**FS87**]. In this section we present the definition of institution, and use the notion of *entailment system* (or π-institution) in order to capture certain proof theoretical aspects of a logic. These concepts are then related by the notion of *logic*, *proof calculus* and *logical system* [**Mes89**].

In the same way categories define mathematical objects by means of their “social life”, reflected by morphisms, *institutions* formalize the model theory of a logic by observing the relation existing between signatures, the relation between the sets of formulas of two related signatures, the relation between, (a) two models of the same signature and (b) the classes of models of two related signatures, and the relation between the semantic consequence relations of two related signatures. Each of these aspects is reflected by introducing the category of signatures and functors going from this category to the category Set (for the case of sets of sentences) and Cat (for the case of categories of models of a given signature).

Definition 3.19. An *institution* is a structure of the form

$$\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}|}\rangle$$

satisfying the following conditions:

- Sign is a category of signatures,
- $\mathbf{Sen} : \mathsf{Sign} \to \mathsf{Set}$ is a functor (let $\Sigma \in |\mathsf{Sign}|$, then $\mathbf{Sen}(\Sigma)$ returns the set of Σ-sentences),
- $\mathbf{Mod} : \mathsf{Sign}^{\mathsf{op}} \to \mathsf{Cat}$ is a functor (let $\Sigma \in |\mathsf{Sign}|$, then $\mathbf{Mod}(\Sigma)$ returns the category of Σ-models),
- $\{\models^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}|}$, where $\models^{\Sigma}\subseteq |\mathbf{Mod}(\Sigma)|\times\mathbf{Sen}(\Sigma)$, is a family of binary relations,

and for any signature morphism $\sigma : \Sigma \to \Sigma'$, Σ-sentence $\phi \in \mathbf{Sen}(\Sigma)$ and Σ'-model $\mathcal{M}' \in |\mathbf{Mod}(\Sigma)|$ the following $\models$-invariance condition holds:

$$\mathcal{M}' \models^{\Sigma'} \mathbf{Sen}(\sigma)(\phi) \text{ iff } \mathbf{Mod}(\sigma^{\mathsf{op}})(\mathcal{M}') \models^{\Sigma} \phi\ .$$

Let $\Sigma \in |\mathsf{Sign}|$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$, then we define the functor $\mathbf{Mod}(\Sigma,\Gamma)$ as the full subcategory of $\mathbf{Mod}(\Sigma)$ determined by those models $\mathcal{M} \in |\mathbf{Mod}(\Sigma)|$ such that for all $\gamma \in \Gamma$, $\mathcal{M} \models^{\Sigma} \gamma$. In addition, it is possible to define a relation $\models^{\Sigma}$ between sets of formulas and formulas in the following way: let $\alpha \in \mathbf{Sen}(\Sigma)$, then

$$\Gamma \models^{\Sigma} \alpha \text{ if and only if } \mathcal{M} \models^{\Sigma} \alpha \text{ for all } \mathcal{M} \in |\mathbf{Mod}(\Sigma,\Gamma)|.$$

An *entailment system* is conceived, in the same way we did in the previous definition, by identifying a family of deductive relations, instead of a family of semantic consequence relations, where each of the elements is associated to a signature. Thus it only rests to require these relations to satisfy the properties of reflexivity, monotonicity, transitivity, a notion of translation between two related signatures, and to reflect the properties of soundness and completeness of the deductive relation.

Definition 3.20. An *entailment system* is a structure of the form

$$\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}|}\rangle$$

satisfying the following conditions:

- Sign is a category of signatures,
- $\mathbf{Sen} : \mathsf{Sign} \to \mathsf{Set}$ is a functor (let $\Sigma \in |\mathsf{Sign}|$, then $\mathbf{Sen}(\Sigma)$ returns the set of Σ-sentences),
- $\{\vdash^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}|}$, where $\vdash^{\Sigma}\subseteq 2^{\mathbf{Sen}(\Sigma)} \times \mathbf{Sen}(\Sigma)$, is a family of binary relations such that for any $\Sigma, \Sigma' \in |\mathsf{Sign}|$, $\{\phi\} \cup \{\phi_i\}_{i\in\mathcal{I}} \subseteq \mathbf{Sen}(\Sigma)$, $\Gamma, \Gamma' \subseteq \mathbf{Sen}(\Sigma)$ the following conditions are satisfied:
  1. reflexivity: $\{\phi\} \vdash^{\Sigma} \phi$,

(2) monotonicity: if $\Gamma \vdash^{\Sigma} \phi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \vdash^{\Sigma} \phi$,
(3) transitivity: if $\Gamma \vdash^{\Sigma} \phi_i$ for all $i \in \mathcal{I}$ and $\{\phi_i\}_{i\in\mathcal{I}} \vdash^{\Sigma} \phi$, then $\Gamma \vdash^{\Sigma} \phi$, and
(4) $\vdash$-translation: if $\Gamma \vdash^{\Sigma} \phi$, then for any morphism $\sigma : \Sigma \to \Sigma'$ in Sign, $\mathbf{Sen}(\sigma)(\Gamma) \vdash^{\Sigma'} \mathbf{Sen}(\sigma)(\phi)$.

Now, from the definition of entailment system, it is possible to give a definition of the category of theories. The relations between theories must be considered as an extension of the relation between the underlying signatures but taking into account the sets of formulas playing the rôle of axioms of the theories.

Definition 3.21. Let $\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}|}\rangle$ be an entailment system, then Th, its category of theories, is a pair $\langle \mathcal{O}, \mathcal{A}\rangle$ such that:

- $\mathcal{O} = \{\, \langle \Sigma, \Gamma\rangle \mid \Sigma \in |\mathsf{Sign}| \text{ and } \Gamma \subseteq \mathbf{Sen}(\Sigma) \,\}$, and
- $\mathcal{A} = \left\{ \sigma : (\Sigma, \Gamma) \to (\Sigma', \Gamma') \;\middle|\; \begin{array}{l} \langle \Sigma, \Gamma\rangle, \langle \Sigma', \Gamma'\rangle \in \mathcal{O}, \\ \sigma : \Sigma \to \Sigma' \text{ is a morphism in } \mathsf{Sign} \text{ and} \\ \text{for all } \gamma \in \Gamma, \Gamma' \vdash^{\Sigma'} \mathbf{Sen}(\sigma)(\gamma) \end{array} \right\}$.

In addition, if a morphism $\sigma : (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ satisfies $\mathbf{Sen}(\sigma)(\Gamma) \subseteq \Gamma'$ it is called *axiom preserving.* This defines the category Th_0 by keeping only those morphisms of Th that are axiom preserving. It is easy to notice that $\mathsf{Th}_0 \hookrightarrow \mathsf{Th}$. Now, if we consider the definition of $\mathbf{Mod}$, extended to signatures and set of sentences, we get a functor $\mathbf{Mod} : \mathsf{Th}^{\mathsf{op}} \to \mathsf{Cat}$ defined as follows: let $T = \langle \Sigma, \Gamma\rangle \in |\mathsf{Th}|$, then

$$\mathbf{Mod}(T) = \mathbf{Mod}(\Sigma, \Gamma)\ .$$

The following definitions will be used in §3 to define one of the ways institutions can relate to each other.

Definition 3.22. Let $\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}|}\rangle$ be an entailment system and $\langle \Sigma, \Gamma\rangle \in |\mathsf{Th}_0|$, then we define ${}^{\bullet} : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma)$ such that if $\Gamma^{\bullet} = \left\{ \gamma \;\middle|\; \Gamma \vdash^{\Sigma} \gamma \right\}$, and ${}^{\bullet} : \mathsf{Th}_0 \to \mathsf{Th}_0$ such that if $\langle \Sigma, \Gamma\rangle^{\bullet} = \langle \Sigma, \Gamma^{\bullet}\rangle$.

Definition 3.23.
Let $\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}|}\rangle$ and $\langle \mathsf{Sign}', \mathbf{Sen}', \{\vdash'^{\Sigma}\}_{\Sigma\in|\mathsf{Sign}'|}\rangle$ be entailment systems, $\Phi : \mathsf{Th}_0 \to \mathsf{Th'}_0$ be a functor and $\alpha : \mathbf{Sen} \to \mathbf{Sen}' \circ \Phi$ a natural transformation; Φ is said to be *α-sensible* if and only if the following conditions are satisfied:

(1) there is a functor $\Phi^{\diamond} : \mathsf{Sign} \to \mathsf{Sign}'$ such that $\mathbf{sign}' \circ \Phi = \Phi^{\diamond} \circ \mathbf{sign}$, where $\mathbf{sign}$ and $\mathbf{sign}$ are the functors from the corresponding category of theories to the corresponding category of signatures, that when applied to a given theory projects its signature, and
(2) if $\langle \Sigma, \Gamma\rangle \in |\mathsf{Th}_0|$ and $\langle \Sigma', \Gamma'\rangle \in \mathsf{Th'}_0$ such that $\Phi(\langle \Sigma, \Gamma\rangle) = \langle \Sigma', \Gamma'\rangle$, then $(\Gamma')^{\bullet} = (\emptyset' \cup \alpha_{\Sigma}(\Gamma))^{\bullet}$.

Φ is said to be *α-simple* if and only if $\Gamma' = \emptyset' \cup \alpha_{\Sigma}(\Gamma)$ is satisfied in Condition 2, instead of $(\Gamma')^{\bullet} = (\emptyset' \cup \alpha_{\Sigma}(\Gamma))^{\bullet}$.

It is trivial to see, based on the monotonicity of ${}^{\bullet}$, that α-simplicity implies α-sensibility.

α-sensible functors have the property that its natural transformation α only depends on signatures, which is a consequence of the following lemma.

Lemma 3.4. *(*[**Mes89**, Lemma 22]*)*
Let $\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ *and* $\langle \mathsf{Sign}', \mathbf{Sen}', \{\vdash'^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}'|}\rangle$ *be entailment systems,* $\Phi : \mathsf{Th}_0 \to \mathsf{Th}'_0$ *be a functor satisfying Condition 1, then any natural transformation* $\alpha : \mathbf{Sen} \to \mathbf{Sen}' \circ \Phi$ *can be obtained from a natural transformation* $\alpha^{\diamond} : \mathbf{Sen} \to \mathbf{Sen}' \circ \Phi^{\diamond}$ *by the horizontal composition with the functor* $\mathbf{sign} : \mathsf{Th}_0 \to \mathsf{Sign}$*.* ■

Now, from Definitions 3.19 and 3.20, it is possible to give a definition of *logic* by relating both of its aspects, its model-theoretical side and proof-theoretical side.

Definition 3.24. A *logic* is a structure of the form

$$\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\vdash^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}, \{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$$

satisfying the following conditions:

- $\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ is an entailment system,
- $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ is an institution, and
- the following *soundness* condition is satisfied: for any $\Sigma \in |\mathsf{Sign}|$, $\phi \in \mathbf{Sen}(\Sigma)$, $\Gamma \subseteq \mathbf{Sen}(\Sigma)$
$$\Gamma \vdash^{\Sigma} \phi \Longrightarrow \Gamma \models^{\Sigma} \phi \ .$$

A logic is *complete* if in addition the following condition is also satisfied: for any $\Sigma \in |\mathsf{Sign}|$, $\phi \in \mathbf{Sen}(\Sigma)$, $\Gamma \subseteq \mathbf{Sen}(\Sigma)$

$$\Gamma \vdash^{\Sigma} \phi \Longleftarrow \Gamma \models^{\Sigma} \phi \ .$$

In Definition 3.20 we associated deductive relations to signatures, but we did not say a word on how these relations are obtained. Now, we introduce the notion of *proof calculus* in order to associate a proof-theoretic structure to the deductive relations introduced by the definitions of entailment system. To do that we will provide a functor $\mathbf{P}$ such that for $T = \langle \Sigma, \Gamma\rangle \in |\mathsf{Th}_0|$, $\mathbf{P}(T)$ is the set of all the proofs whose axioms are in Γ. To this purpose we will give a definition of proof calculus relaying on a category Struct_{PC} whose objects should be those proofs. In [**Mes89**, Example 11, pp. 15], Meseguer presents natural deduction as a proof calculus for first-order logic by resorting to *multicategories* (see [**Mes89**, Definition 10]). From this example it is easy to see why we would like to present the proof theory and the proof-theoretic structure of a logic separately. It is because, following [**Mes89**], any proof-theoretic structure provides an “implementation” of a single proof theory of a logic.

Definition 3.25. A *proof calculus* is a structure of the form

$$\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}, \mathbf{P}, \mathbf{Pr}, \pi\rangle$$

satisfying the following conditions:

- $\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ is an entailment system,
- $\mathbf{P} : \mathsf{Th}_0 \to \mathsf{Struct}_{PC}$ is a functor (let $T \in |\mathsf{Th}_0|$, then $\mathbf{P}(T) \in |\mathsf{Struct}_{PC}|$ is the proof-theoretical structure of T),
- $\mathbf{Pr} : \mathsf{Struct}_{PC} \to \mathsf{Set}$ is a functor (let $T \in |\mathsf{Th}_0|$, then $\mathbf{Pr}(\mathbf{P}(T))$ is the set of proofs of T; the composite functor $\mathbf{Pr} \circ \mathbf{P} : \mathsf{Th}_0 \to \mathsf{Set}$ will be denoted by $\mathbf{proofs}$), and

- $\pi : \mathbf{proofs} \to \mathbf{Sen}$ is a natural transformation (let $T \in |\mathsf{Th}_0|$, then $\pi_T : \mathbf{proofs}(T) \to \mathbf{Sen}(T)$ is the projection of the set of theorem of T).

Finally, a *logical system* will be a logic plus a proof calculus for its proof theory.

Definition 3.26. A *logical system* is a structure of the form

$$\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\vdash^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \{\models^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \mathbf{P}, \mathbf{Pr}, \pi \rangle$$

satisfying the following conditions:

- $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\vdash^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \{\models^\Sigma\}_{\Sigma \in |\mathsf{Sign}|} \rangle$ is a logic, and
- $\langle \mathsf{Sign}, \mathbf{Sen}, \{\vdash^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \mathbf{P}, \mathbf{Pr}, \pi \rangle$ is an proof calculus.

3. Institution morphisms and representation maps

As we mentioned before, institutions capture, in an abstract way, the model theory of a logic. They can be related by means of different kinds of mappings such as institution morphisms [**GB84**] and institution representations [**Mes89**]. These mappings between institutions are extensively discussed by Tarlecki in [**Tar96**]. The main difference between them being that institution morphisms allow one to build a richer institution from poorer ones, while representations allow us to encode poorer institutions into a richer one.

Definition 3.27. (Institution morphism)
Let $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models_\Sigma\}_{\Sigma \in |\mathsf{Sign}|} \rangle$ and $\langle \mathsf{Sign}', \mathbf{Sen}', \mathbf{Mod}', \{\models'_\Sigma\}_{\Sigma \in |\mathsf{Sign}'|} \rangle$ be the institutions I and I' respectively, then $\langle \gamma^{Sign}, \gamma^{Sen}, \gamma^{Mod} \rangle : I \to I'$ is an *institution morphism* if and only if:

- $\gamma^{Sign} : \mathsf{Sign}' \to \mathsf{Sign}$ is a functor,
- $\gamma^{Sen} : \gamma^{Sign} \circ \mathbf{Sen} \to \mathbf{Sen}'$, is a natural transformation (i.e. a natural family of functions $\gamma^{Sen}_{\Sigma'} : \mathbf{Sen}(\gamma^{Sign}(\Sigma')) \to \mathbf{Sen}'(\Sigma')$), such that for each $\Sigma'_1, \Sigma'_2 \in |\mathsf{Sign}'|$ and $\sigma' : \Sigma'_1 \to \Sigma'_2$ morphism in Sign',

$$\begin{tikzcd}
\mathbf{Sen}(\gamma^{Sign}(\Sigma'_2)) \arrow[r, "\gamma^{Sen}_{\Sigma'_2}"] & \mathbf{Sen}'(\Sigma'_2) & \Sigma'_2 \\
\mathbf{Sen}(\gamma^{Sign}(\Sigma'_1)) \arrow[r, "\gamma^{Sen}_{\Sigma'_1}"] \arrow[u, "\mathbf{Sen}(\gamma^{Sign}(\sigma'))"'] & \mathbf{Sen}'(\Sigma'_1) \arrow[u, "\mathbf{Sen}'(\sigma')"'] & \Sigma'_1 \arrow[u, "\sigma'"']
\end{tikzcd}$$

- $\gamma^{Mod} : \mathbf{Mod'} \to (\gamma^{Sign})^{\mathsf{op}} \circ \mathbf{Mod}$,[2] is a natural transformation (i.e. the family of functors $\gamma^{Mod}_{\Sigma'} : \mathbf{Mod}'(\Sigma') \to \mathbf{Mod}((\gamma^{Sign})^{\mathsf{op}}(\Sigma'))$ is natural), such that for each $\Sigma'_1, \Sigma'_2 \in |\mathsf{Sign}'|$ and $\sigma' : \Sigma'_1 \to \Sigma'_2$ morphism in Sign',

[2]The functor $(\gamma^{Sign})^{\mathsf{op}} : \mathsf{Sign}'^{\mathsf{op}} \to \mathsf{Sign}^{\mathsf{op}}$ is the same as $\gamma^{Sign} : \mathsf{Sign}' \to \mathsf{Sign}$ but considered between the opposite categories.

$$\begin{tikzcd}
\mathbf{Mod}'(\Sigma'_2) \arrow[r, "\gamma^{Mod}_{\Sigma'_2}"] \arrow[d, "\mathbf{Mod}'(\sigma'^{\mathsf{op}})"] & \mathbf{Mod}((\gamma^{Sign})^{\mathsf{op}}(\Sigma'_2)) \arrow[d, "\mathbf{Mod}((\gamma^{Sign})^{\mathsf{op}}(\sigma'^{\mathsf{op}}))"] & \Sigma'_2 \\
\mathbf{Mod}'(\Sigma'_1) \arrow[r, "\gamma^{Mod}_{\Sigma'_1}"] & \mathbf{Mod}((\gamma^{Sign})^{\mathsf{op}}(\Sigma'_1)) & \Sigma'_1 \arrow[u, "\sigma'"']
\end{tikzcd}$$

such that for any $\Sigma' \in |\mathsf{Sign}'|$, the function $\gamma^{Sen}_{\Sigma'} : \mathbf{Sen}(\gamma^{Sign}(\Sigma')) \to \mathbf{Sen}'(\Sigma')$ and the functor $\gamma^{Mod}_{\Sigma'} : \mathbf{Mod}'(\Sigma') \to \mathbf{Mod}((\gamma^{Sign})^{\mathsf{op}}(\Sigma'))$ preserves the following satisfaction condition: for any $\alpha \in \mathbf{Sen}(\gamma^{Sign}(\Sigma'))$ and $\mathcal{M} \in |\mathbf{Mod}(\Sigma')|$,

$$\mathcal{M} \models_{\Sigma'} \gamma^{Sen}_{\Sigma'}(\alpha) \text{ iff } \gamma^{Mod}_{\Sigma'}(\mathcal{M}) \models_{\gamma^{Sign}(\Sigma')} \alpha \ .$$

In [**Tar96**], Tarlecki calls our attention to the fact that institution morphism capture how a "richer" institution (richer in terms of its model theory) is built on top of "poorer" ones and also shows in [**Tar96**, Definition 4.2] that institutions together with institution morphisms form a category, denoted as Ins. In this sense, Ins provides the formal framework in which it is possible to build specifications incrementally in a modular way by constructing limits. To support this observation Tarlecki proves that Ins is complete in [**Tar96**, Theorem 4.3].

Definition 3.28. (Representation map of institution)
Let $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models_\Sigma\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ and $\langle \mathsf{Sign}', \mathbf{Sen}', \mathbf{Mod}', \{\models'_\Sigma\}_{\Sigma \in |\mathsf{Sign}'|}\rangle$ be the institutions I and I' respectively, then $\langle \gamma^{Sign}, \gamma^{Sen}, \gamma^{Mod}\rangle : I \to I'$ is a *representation map* of institutions if and only if:

- $\gamma^{Sign} : \mathsf{Sign} \to \mathsf{Sign}'$ is a functor,
- $\gamma^{Sen} : \mathbf{Sen} \to \gamma^{Sign} \circ \mathbf{Sen}'$, is a natural transformation (i.e. a natural family of functions $\gamma^{Sen}_\Sigma : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\gamma^{Sign}(\Sigma)))$, such that for each $\Sigma_1, \Sigma_2 \in |\mathsf{Sign}|$ and $\sigma : \Sigma_1 \to \Sigma_2$ morphism is Sign,

$$\begin{tikzcd}
\mathbf{Sen}(\Sigma_2) \arrow[r, "\gamma^{Sen}_{\Sigma_2}"] & \mathbf{Sen}'(\gamma^{Sign}(\Sigma_2)) & \Sigma_2 \\
\mathbf{Sen}(\Sigma_1) \arrow[r, "\gamma^{Sen}_{\Sigma_1}"] \arrow[u, "\mathbf{Sen}(\sigma)"'] & \mathbf{Sen}'(\gamma^{Sign}(\Sigma_1)) \arrow[u, "\mathbf{Sen}'(\gamma^{Sign}(\sigma))"'] & \Sigma_1 \arrow[u, "\sigma"']
\end{tikzcd}$$

- $\gamma^{Mod} : (\gamma^{Sign})^{\mathsf{op}} \circ \mathbf{Mod}' \to \mathbf{Mod}$, is a natural transformation (i.e. the family of functors $\gamma^{Mod}_\Sigma : \mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\Sigma)) \to \mathbf{Mod}(\Sigma)$ is natural), such that for each $\Sigma_1, \Sigma_2 \in |\mathsf{Sign}|$ and $\sigma : \Sigma_1 \to \Sigma_2$ morphism in Sign,

$$\begin{tikzcd}
\mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\Sigma_2)) \arrow[r, "\gamma^{Mod}_{\Sigma_2}"] \arrow[d, "\mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\sigma^{\mathsf{op}}))"] & \mathbf{Mod}(\Sigma_2) \arrow[d, "\mathbf{Mod}(\sigma^{\mathsf{op}})"] & \Sigma_2 \\
\mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\Sigma_1)) \arrow[r, "\gamma^{Mod}_{\Sigma_1}"] & \mathbf{Mod}(\Sigma_1) & \Sigma_1 \arrow[u, "\sigma"']
\end{tikzcd}$$

such that for any $\Sigma \in |\mathsf{Sign}|$, the function $\gamma^{Sen}_{\Sigma} : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\gamma^{Sign}(\Sigma))$ and the functor $\gamma^{Mod}_{\Sigma} : \mathbf{Mod}'(\gamma^{Sign}(\Sigma)) \to \mathbf{Mod}(\Sigma)$ preserves the following satisfaction condition: for any $\alpha \in \mathbf{Sen}(\Sigma)$ and $\mathcal{M}' \in |\mathbf{Mod}(\gamma^{Sign}(\Sigma))|$,

$$\mathcal{M}' \models_{\gamma^{Sign}(\Sigma)} \gamma^{Sen}_{\Sigma}(\alpha) \text{ iff } \gamma^{Mod}_{\Sigma}(\mathcal{M}') \models_{\Sigma} \alpha \ .$$

Even though representation maps between institutions differ from institution morphisms only in the direction in which the arrow relates two institutions, this technical detail induces a totally different interpretation on the constructions they perform. An institution morphism $\gamma : I' \to I$ expresses how the institution I' is built over the institution I; this fact can be observed in the way γ^{Sen} and γ^{Mod} are defined in Definition 3.27. Both natural transformations characterize how the "richer" set of sentences (respectively, category of models) is built from the "poorer" one. This is done by:

- projecting, from a given I'-signature Σ', the I-signature interpreted by Σ',
- projecting, for a given I'-signature Σ', the set of I-sentences interpreted by the Σ'-sentences,
- projecting, for a given I'-signature Σ', the category of I-models, interpreted by the category of Σ'-models.

The direction of the arrows shows that only some parts of I' are used (those parts reflected by the target theories of γ^{Sign}) to interpret I.

On the other hand, a representation map $\gamma : I \to I'$ expresses how the "poorer" set of sentences (respectively, category of models) associated to I is encoded in the "richer" one associated to I', and this is done by:

- constructing, for a given I-signature Σ, an I'-signature into which Σ can be interpreted,
- translating, for a given I-signature Σ, the set of Σ-sentences to the corresponding I'-sentences,
- obtaining, for a given I-signature Σ, the category of Σ-models from the corresponding category of Σ'-models.

In this case, the direction of the arrows shows that the whole of I is represented by some parts of I'.

The main difference between institution morphisms and representation maps can be found in the behavior of the natural transformation γ^{Mod}. In both cases γ^{Mod} performs an operation that can be thought-of as "taking reduct" to a simpler class of models. As we mentioned before, the only difference is that in the case of institution morphisms, it exposes that some parts of I'-models reflect I-models, while in the case of representation maps, it shows that I-models are completely represented by I'-models.

Whether representation maps are a good choice for relating institutions, we are interested in an extension of Definition 3.28 in which, instead of having a functor γ^{Sign} between the categories of signatures, we will have a functor $\gamma^{Th_0} : \mathsf{Sign}^{I} \to \mathsf{Th}^{I'}{}_0$ between the category of signatures of the institution I, and the category of theories with axiom preserving morphisms of the institution I'.

As we are only interested in this new version of representation maps we will refer to it with the same name.

Definition 3.29.
Let $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models_\Sigma\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ and $\langle \mathsf{Sign}', \mathbf{Sen}', \mathbf{Mod}', \{\models'_\Sigma\}_{\Sigma \in |\mathsf{Sign}'|}\rangle$ be the institutions I and I' respectively, then $\langle \gamma^{\mathsf{Th}_0}, \gamma^{Sen}, \gamma^{Mod}\rangle : I \to I'$ is a *representation map* of institutions if and only if:

- $\gamma^{Th_0} : \mathsf{Sign} \to \mathsf{Th}_0$ is a functor,
- $\gamma^{Sen} : \mathbf{Sen} \to \gamma^{Th_0} \circ \mathbf{Sen}'$, is a natural transformation (i.e. a natural family of functions $\gamma^{Sen}_\Sigma : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\gamma^{Th_0}(\Sigma)))$, such that for each $\Sigma_1, \Sigma_2 \in |\mathsf{Sign}|$ and $\sigma : \Sigma_1 \to \Sigma_2$ morphism is Sign,

$$\begin{tikzcd}
\mathbf{Sen}(\Sigma_2) \arrow[r, "\gamma^{Sen}_{\Sigma_2}"] & \mathbf{Sen}'(\gamma^{Sign}(\Sigma_2)) & \Sigma_2 \\
\mathbf{Sen}(\Sigma_1) \arrow[u, "\mathbf{Sen}(\sigma)"'] \arrow[r, "\gamma^{Sen}_{\Sigma_1}"] & \mathbf{Sen}'(\gamma^{Sign}(\Sigma_1)) \arrow[u, "\mathbf{Sen}'(\gamma^{Sign}(\sigma))"'] & \Sigma_1 \arrow[u, "\sigma"']
\end{tikzcd}$$

- $\gamma^{Mod} : (\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}' \to \mathbf{Mod}$, is a natural transformation (i.e. the family of functors $\gamma^{Mod}_\Sigma : \mathbf{Mod}'((\gamma^{Th_0})^{\mathsf{op}}(\Sigma)) \to \mathbf{Mod}(\Sigma)$ is natural), such that for each $\Sigma_1, \Sigma_2 \in |\mathsf{Sign}|$ and $\sigma : \Sigma_1 \to \Sigma_2$ morphism in Sign,

$$\begin{tikzcd}
\mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\Sigma_2)) \arrow[r, "\gamma^{Mod}_{\Sigma_2}"] \arrow[d, "\mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\sigma^{\mathsf{op}}))"] & \mathbf{Mod}(\Sigma_2) \arrow[d, "\mathbf{Mod}(\sigma^{\mathsf{op}})"] & \Sigma_2 \\
\mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\Sigma_1)) \arrow[r, "\gamma^{Mod}_{\Sigma_1}"] & \mathbf{Mod}(\Sigma_1) & \Sigma_1 \arrow[u, "\sigma"']
\end{tikzcd}$$

such that for any $\Sigma \in |\mathsf{Sign}|$, the function $\gamma^{Sen}_\Sigma : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\gamma^{Th_0}(\Sigma))$ and the functor $\gamma^{Mod}_\Sigma : \mathbf{Mod}'(\gamma^{Th_0}(\Sigma)) \to \mathbf{Mod}(\Sigma)$ preserves the following satisfaction condition: for any $\alpha \in \mathbf{Sen}(\Sigma)$ and $\mathcal{M}' \in |\mathbf{Mod}(\gamma^{Th_0}(\Sigma))|$,

$$\mathcal{M}' \models_{\gamma^{Th_0}(\Sigma)} \gamma^{Sen}_\Sigma(\alpha) \text{ iff } \gamma^{Mod}_\Sigma(\mathcal{M}') \models_\Sigma \alpha \ .$$

This alternative definition of representation map between institutions (which, in fact, is very similar to the original definition of representation map given by Meseguer in [**Mes89**, Definition 27] under the name *map of institutions*) was presented in order to be able to add restrictions on the richer class of models. The need for this alternative definition will be clear in Chapter 5, §3.

CHAPTER 4

The logical system behind full proper closure fork algebra with urelements

This chapter addresses the problem of building a logic (in the sense of Definition 3.26) on top of full proper closure fork algebras with urelements. Since the variety generated by fPCFAU is completely characterized by ω-CCFAU, we might consider to relativize the institution (entailment system) of equational logic rather than introducing a new one from scratch. This might work for a while, but there are technical and methodological reasons for presenting the explicit construction.

On the technical side, notice that the actual proof systems for equational logic and ω-CCFAU differ in their proof rules (ω-CCFAU has an extra rule – see Definition 2.47). This prevents us from modeling the proof calculus [**Mes89**, Definition 12] ω-CCFAU as a proof subcalculus [**Mes89**, Definition 14] of equational logic. From the methodological point of view, the categorical construction provides important information to the reader on what operations are part of the logic, how morphisms are defined, etc.

The chapter is organized in three main section. The first one (§1) is dedicated to construct the institution behind full proper closure fork algebras with urelements. To do so, we first prove that the signatures of full proper closure fork algebras with urelements together with total mappings between extralogical function symbols form a category (see Lemma 4.1). Second, (from Definition 4.1, to Lemma 4.8) we provide the definition of the functor **Sen**. Then (from Definition 4.4 to Lemma 4.14), we provide the definition of the functor **Mod**. And finally (from Lemma 4.15 to Lemma 4.17) we prove the invariance of the satisfiability relation under change of notation.

The second section (§2) defines an entailment system in a standard way, and constructs, by means of the institution presented in §1 and this entailment system, a sound and complete logic.

The third section (§3) provides the necessary definitions and results in order to fit ω-CCFAU in the definition of proof calculus, thus obtaining a logical system.

From now on, we will omit the subscript fPCFAU except in those cases where its absence could introduce ambiguities.

1. The institution behind full proper closure fork algebras with urelements

In this section we will define an institution on top of fPCFAU. The section is structured following the order of requirements stated in Definition 3.19.

From now on we will assume fixed but arbitrary fPCFAU-signatures with the shape $\Sigma = \langle \{f_i\}_{i \in \mathcal{I}} \rangle$, and resorting to superindexing with $'$ if more than one is needed (for example, $\Sigma' = \langle \{f'_{i'}\}_{i' \in \mathcal{I}'} \rangle$). In the same way, if Σ is a fPCFAU-signature, we will assume fixed but arbitrary Σ-models with the shape $\mathcal{M} = \langle \mathcal{P}, \{\underline{f_i}\}_{i \in \mathcal{I}} \rangle$ subindexing them if more than one is needed (for example, $\mathcal{M}_k = \langle \mathcal{P}_k, \{\underline{f_{ki}}\}_{i \in \mathcal{I}} \rangle$), and superindexing with $'$ when referring to Σ'-models.

Lemma 4.1. *We define* Sign $= \langle Sign_{\mathsf{fPCFAU}}, \mathcal{A} \rangle$*, where*

$$\mathcal{A} = \left\{ \sigma : \langle \{f_i\}_{i \in \mathcal{I}} \rangle \to \langle \{f'_i\}_{i \in \mathcal{I}'} \rangle \;\middle|\; \begin{array}{c} \sigma : \mathcal{I} \to \mathcal{I}' \textit{ is a total function, such that} \\ (\forall i \in \mathcal{I})(arity(f_i) = arity(f'_{\sigma(i)})) \;. \end{array} \right\}$$

Then, Sign *is a category.*

Proof. It is trivial to see that $id_{\mathcal{I}}$ is an arity preserving total function so $id_\Sigma \in \mathcal{A}$. Now, as the composition of arity preserving total functions is itself an arity preserving total function, then if $\sigma : \Sigma \to \Sigma', \sigma' : \Sigma' \to \Sigma'' \in \mathcal{A}$, we get that $\sigma \circ \sigma' : \Sigma \to \Sigma'' \in \mathcal{A}$.

It is also trivial to see that (1) for any $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ morphism in $|\mathsf{Sign}|$, $id_\Sigma \circ \sigma = \sigma = \sigma \circ id_{\Sigma'}$, and (2) composition of morphisms, being the composition of arity preserving total functions, is associative.

Consequently Sign is a category. ■

The intuitive meaning of Sign is that an object is an fPCFAU-signature and an arrow $\sigma : \Sigma \to \Sigma'$ express a translation of Σ-symbols to Σ'-symbols. Since the fork algebra operators are not part of the signatures, they are not translated.

Definition 4.1. Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$, and $\sigma : \Sigma \to \Sigma'$ be a morphism in Sign. Then $\sigma_{term} : Term_\Sigma \to Term_{\Sigma'}$ is defined inductively on the structure of terms as follows:

- $\sigma_{term}(\star) \equiv \star$, for all $\star \in \{1, 0, 1'\}$,
- $\sigma_{term}(r) \equiv r$, for all $r \in \mathcal{R}$,
- if $\{t, t_1, t_2\} \subseteq Term_\Sigma$, then
  - $\sigma_{term}(t\star) \equiv \sigma_{term}(t)\star$, for all $\star \in \{^{-}, ^{\smile}, ^{\diamond}, ^{*}\}$,
  - $\sigma_{term}(t_1 \star t_2) \equiv \sigma_{term}(t_1) \star \sigma_{term}(t_2)$, for all $\star \in \{+, \cdot, ;, \nabla\}$,
- if $\{t_1, \ldots, t_n\} \subseteq Term_\Sigma$, then
  $\sigma_{term}(f_i(t_1, \ldots, t_{arity(f_i)})) \equiv f'_{\sigma(i)}(\sigma_{term}(t_1), \ldots, \sigma_{term}(t_n))$, for all $i \in \mathcal{I}$.

Lemma 4.2. *Let* $\Sigma \in |\mathsf{Sign}|$ *and* $t \in Term_\Sigma$*, then* $(id_\Sigma)_{term}(t) \equiv t$.

Proof. The proof follows by induction on the structure of t and using Definition 4.1. ■

LEMMA 4.3. *Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ and $\sigma' : \Sigma' \to \Sigma''$ be morphisms in* Sign *then for all $t \in Term_\Sigma$, $(\sigma \circ \sigma')_{term}(t) \equiv \sigma_{term} \circ \sigma'_{term}(t)$.*

PROOF. The proof follows by induction on the structure of t. ■

LEMMA 4.4. *Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in* Sign*, then σ_{term} is a total function.*

PROOF. The proof follows trivially from Definition 4.1. ■

DEFINITION 4.2. Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in Sign. Then, $\sigma_{eq} : Sen_\Sigma \to Sen_{\Sigma'}$ is defined as $\sigma_{eq}(t_1 = t_2) \equiv \sigma_{term}(t_1) = \sigma_{term}(t_2)$.

LEMMA 4.5. *Let $\Sigma \in |\mathsf{Sign}|$ and $s \in Sen_\Sigma$, then*

$$(id_\Sigma)_{eq}(s) \equiv s\ .$$

PROOF. The proof follows from Lemma 4.2. ■

LEMMA 4.6. *Let $\Sigma, \Sigma', \Sigma'' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma', \sigma' : \Sigma' \to \Sigma''$ be morphisms in* Sign. *Then for all $t_1 = t_2 \in Sen_\Sigma$, $(\sigma \circ \sigma')_{eq}(t_1 = t_2) \equiv \sigma_{eq} \circ \sigma'_{eq}(t_1 = t_2)$.*

PROOF.

$$\begin{array}{ll}
 & (\sigma \circ \sigma')_{eq}(t_1 = t_2) \\
\equiv & (\sigma \circ \sigma')_{term}(t_1) = (\sigma \circ \sigma')_{term}(t_2) \\
 & \qquad \text{[by Definition 4.2]} \\
\equiv & \sigma_{term} \circ \sigma'_{term}(t_1) = \sigma_{term} \circ \sigma'_{term}(t_2) \\
 & \qquad \text{[by Lemma 4.3]} \\
\equiv & \sigma'_{term}(\sigma_{term}(t_1)) = \sigma'_{term}(\sigma_{term}(t_2)) \\
\equiv & \sigma'_{eq}(\sigma_{term}(t_1) = \sigma_{term}(t_2)) \\
 & \qquad \text{[by Definition 4.2]} \\
\equiv & \sigma'_{eq}(\sigma_{eq}(t_1 = t_2)) \\
 & \qquad \text{[by Definition 4.2]} \\
\equiv & \sigma_{eq} \circ \sigma'_{eq}(t_1 = t_2)
\end{array}$$

■

LEMMA 4.7. *Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in* Sign *then σ_{eq} is a total function.*

PROOF. This lemma follows from Lemma 4.4, which states that σ_{term} is a total function. ■

DEFINITION 4.3. Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in Sign. We define **Sen** : Sign $\to$ Set as:

- **Sen**$(\Sigma) = Sen_\Sigma$ (see the paragraph before Definition 2.50), and
- **Sen**$(\sigma) = \sigma_{eq}$.

LEMMA 4.8. **Sen** *is a functor.*

PROOF. From Definition 4.3 we know that if $\Sigma \in |\mathsf{Sign}|$, then **Sen**$(\Sigma)$ yields an object in the category Set and that if $\sigma : \Sigma \to \Sigma'$ is a morphism in Sign, then **Sen**(σ) yields a morphism in the category Set.

We will first prove that $\mathbf{Sen}(id_\Sigma) = id_{\mathbf{Sen}(\Sigma)}$. Let $s \in \mathbf{Sen}(\Sigma)$, then

$$\begin{array}{ll} & \mathbf{Sen}(id_\Sigma)(s) \\ = & (id_\Sigma)_{eq}(s) \\ & \quad \text{[by Definition 4.3]} \\ = & s \\ & \quad \text{[by Lemma 4.5]} \\ = & id_{\mathbf{Sen}(\Sigma)}(s) \\ & \quad \text{[by Definition 4.3]} \end{array}$$

Next we prove that if $\sigma : \Sigma \to \Sigma'$ and $\sigma' : \Sigma' \to \Sigma''$ are morphisms in Sign, then $\mathbf{Sen}(\sigma \circ \sigma') = \mathbf{Sen}(\sigma) \circ \mathbf{Sen}(\sigma')$ as a consequence of Lemma 4.6. Let $s \in \mathbf{Sen}(\Sigma)$, then

$$\begin{array}{ll} & \mathbf{Sen}(\sigma \circ \sigma')(s) \\ = & (\sigma \circ \sigma')_{eq}(s) \\ & \quad \text{[by Definition 4.3]} \\ = & \sigma_{eq} \circ \sigma'_{eq}(s) \\ & \quad \text{[by Lemma 4.6]} \\ = & \sigma'_{eq}(\sigma_{eq}(s)) \\ = & \mathbf{Sen}(\sigma')(\sigma_{eq}(s)) \\ & \quad \text{[by Definition 4.3]} \\ = & \mathbf{Sen}(\sigma')(\mathbf{Sen}(\sigma)(s)) \\ & \quad \text{[by Definition 4.3]} \\ = & (\mathbf{Sen}(\sigma) \circ \mathbf{Sen}(\sigma'))(s) \\ & \quad \text{[by Definition of } \circ \text{]} \end{array}$$

■

Definition 4.4. Let $\Sigma \in |\mathsf{Sign}|$, then $\mathbf{Mod}(\Sigma) = \langle \mathcal{O}, \mathcal{A} \rangle$ is defined as follows:

- $\mathcal{O} = Mod_\Sigma$ (see Definition 2.50),
- $\mathcal{A} = \{\, \gamma : \mathcal{M} \to \mathcal{M}' \mid \mathcal{M}, \mathcal{M}' \in \mathcal{O}$ and γ is an homomorphism $\}$.

Lemma 4.9. *Let* $\Sigma \in |\mathsf{Sign}|$*, then* $\mathbf{Mod}(\Sigma)$ *is a category.*

Proof. The proof of this lemma follows by observing that for all $\mathcal{M} \in |\mathbf{Mod}(\Sigma)|$ the identity function $id_\mathcal{M} : |\mathcal{M}| \to |\mathcal{M}|$ is a homomorphism and that, by [**BS81**, Theorem 6.5], composition of homomorphisms yields homomorphisms.

Finally, it is easy to prove that if $\mathcal{M}, \mathcal{M}' \in |\mathbf{Mod}(\Sigma)|$ and $\sigma : \mathcal{M} \to \mathcal{M}'$ is a morphism in $\mathbf{Mod}(\Sigma)$, then $id_\mathcal{M} \circ \sigma = \sigma = \sigma \circ id_{\mathcal{M}'}$, and that $\circ$ is associative. ■

Definition 4.5. Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in Sign. Let $\mathcal{M}' \in |\mathbf{Mod}(\Sigma')|$, then

$$\mathcal{M}' \restriction_{\sigma^{\mathsf{op}}} = \langle \mathcal{P}', \{\underline{f'_{\sigma(i)}}\}_{i \in \mathcal{I}} \rangle \ .$$

Lemma 4.10. *Let* $\Sigma, \Sigma', \Sigma'' \in |\mathsf{Sign}|$ *and* $\sigma : \Sigma \to \Sigma'$*,* $\sigma' : \Sigma' \to \Sigma''$ *be morphisms in* Sign*. Let* $\mathcal{M}'' \in |\mathbf{Mod}(\Sigma'')|$*.*

Then $\mathcal{M}'' \restriction_{\sigma'^{\mathsf{op}} \circ \sigma^{\mathsf{op}}} = (\mathcal{M}'' \restriction_{\sigma'^{\mathsf{op}}}) \restriction_{\sigma^{\mathsf{op}}}$*.*

Proof.

$$\begin{array}{rl}
& \langle \mathcal{P}'', \{\underline{f''_{i''}}\}_{i'' \in \mathcal{I}''} \rangle \restriction_{\sigma'^{\mathsf{op}} \circ \sigma^{\mathsf{op}}} \\
= & \langle \mathcal{P}'', \{\underline{f''_{(\sigma \circ \sigma')(i)}}\}_{i \in \mathcal{I}} \rangle \\
& \quad [\text{by Definition 4.5 and } (\sigma \circ \sigma')^{\mathsf{op}} = \sigma'^{\mathsf{op}} \circ \sigma^{\mathsf{op}}] \\
= & \langle \mathcal{P}'', \{\underline{f''_{\sigma'(\sigma(i))}}\}_{i \in \mathcal{I}} \rangle \\
= & \langle \mathcal{P}'', \{\underline{f''_{\sigma'(i')}}\}_{\sigma(i) = i' \wedge i \in \mathcal{I}} \rangle \\
= & \langle \mathcal{P}'', \{\underline{f''_{\sigma'(i')}}\}_{i' \in \mathcal{I}'} \rangle \restriction_{\sigma^{\mathsf{op}}} \\
& \quad [\text{by Definition 4.5}] \\
= & \langle \mathcal{P}'', \{\underline{f''_{i''}}\}_{\sigma'(i') = i'' \wedge i' \in \mathcal{I}'} \rangle \restriction_{\sigma^{\mathsf{op}}} \\
= & \langle \mathcal{P}'', \{\underline{f''_{i''}}\}_{i'' \in \mathcal{I}''} \rangle \restriction_{\sigma'^{\mathsf{op}}} \restriction_{\sigma^{\mathsf{op}}} \\
& \quad [\text{by Definition 4.5}]
\end{array}$$

■

The next definition characterizes the behavior of **Mod** when it is applied to morphisms in Sign. As morphisms $\sigma : \Sigma \to \Sigma'$ in Sign can be interpreted as translations from Σ-symbols to Σ'-symbols and **Mod** must be contravariant, **Mod**(σ) is, intuitively, the operation that takes reduct of Σ'-algebras to the similarity type of Σ-algebras.

Definition 4.6. Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in Sign. Let $\mathcal{M}' \in |\mathbf{Mod}(\Sigma')|$ and γ' be a morphism in $\mathbf{Mod}(\Sigma')$, then

- $\mathbf{Mod}(\sigma^{\mathsf{op}})(\mathcal{M}') = \mathcal{M}' \restriction_{\sigma^{\mathsf{op}}}$,
- $\mathbf{Mod}(\sigma^{\mathsf{op}})(\gamma') = \gamma'$.

As we mentioned before, if $\sigma : \Sigma \to \Sigma'$ is a morphism in Sign, $\mathbf{Mod}(\sigma^{\mathsf{op}})$ acts on Σ'-models taking Σ-reducts so we expect $\mathbf{Mod}(\sigma^{\mathsf{op}})$ to act on morphisms between Σ'-models by yielding morphisms between the Σ-reducts of the Σ'-models.

Lemma 4.11. *Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in* Sign *and $\mathcal{M}' \in |\mathbf{Mod}(\Sigma')|$, then $\mathbf{Mod}(\sigma^{\mathsf{op}})(\mathcal{M}') \in |\mathbf{Mod}(\Sigma)|$.*

Proof. The proof of this lemma follows from Definition 4.5. ■

Lemma 4.12. *Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in* Sign*, and γ' be a morphism in $\mathbf{Mod}(\Sigma')$. Then, $\mathbf{Mod}(\sigma^{\mathsf{op}})(\gamma')$ is a morphism in $\mathbf{Mod}(\Sigma)$.*

Proof. The proof follows by observing that, given $\mathcal{M}_1$ and $\mathcal{M}_2$ two algebras of type Σ', a homomorphism $h : \mathcal{M}_1 \to \mathcal{M}_2$ is a homomorphism between the Σ-reducts of $\mathcal{M}_1$ and $\mathcal{M}_2$. ■

Lemma 4.13. *Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ and $\sigma : \Sigma \to \Sigma'$ be a morphism in* Sign*. Then $\mathbf{Mod}(\sigma^{\mathsf{op}}) : \mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ is a functor.*

Proof. By Lemma 4.9 we know that if $\Sigma \in |\mathsf{Sign}|$, then $\mathbf{Mod}(\Sigma)$ is a category. By Lemma 4.11 we know that $\mathbf{Mod}(\sigma^{\mathsf{op}})$ sends Σ'-models to Σ-models, and by Lemma 4.12 we know that $\mathbf{Mod}(\sigma^{\mathsf{op}})$ maps Σ'-model morphisms to Σ-model morphisms. From Definition 4.6, it follows that if $\mathcal{M}' \in |\mathbf{Mod}(\Sigma')|$, then $\mathbf{Mod}(\sigma^{\mathsf{op}})$ maps the morphism $id_{\mathcal{M}'} : \mathcal{M}' \to \mathcal{M}'$ to the morphism $id_{\mathbf{Mod}(\sigma^{\mathsf{op}})(\mathcal{M}')} : \mathbf{Mod}(\sigma^{\mathsf{op}})(\mathcal{M}') \to \mathbf{Mod}(\sigma^{\mathsf{op}})(\mathcal{M}')$; and that $\mathbf{Mod}(\sigma^{\mathsf{op}})(\gamma \circ \gamma') = \mathbf{Mod}(\sigma^{\mathsf{op}})(\gamma) \circ \mathbf{Mod}(\sigma^{\mathsf{op}})(\gamma')$. ■

LEMMA 4.14. $\mathbf{Mod} : \mathsf{Sign}^{\mathsf{op}} \to \mathsf{Cat}$ *is a functor.*

PROOF. If $\sigma : \Sigma \to \Sigma'$ is a morphism in Sign, then by Lemma 4.9 $\mathbf{Mod}(\Sigma)$ is a category, and by Lemma 4.13 $\mathbf{Mod}(\sigma^{\mathsf{op}})$ is a functor.

By Definitions 4.4 and 4.6, $\mathbf{Mod}(id_\Sigma) = id_{\mathbf{Mod}(\Sigma)}$.

Finally, by Definition 4.6 and Lemma 4.10, if $\sigma : \Sigma \to \Sigma'$ and $\sigma' : \Sigma' \to \Sigma''$ are morphisms in Sign, then $\mathbf{Mod}(\sigma'^{\mathsf{op}} \circ \sigma^{\mathsf{op}}) = \mathbf{Mod}(\sigma'^{\mathsf{op}}) \circ \mathbf{Mod}(\sigma^{\mathsf{op}})$. ∎

Next we will provide some definitions and prove some lemmas required by the proof of the preservation of the $\models$-invariance condition from Definition 3.19.

LEMMA 4.15. *Let* $\Sigma, \Sigma' \in |\mathsf{Sign}|$, *and* $\sigma : \Sigma \to \Sigma'$ *be a morphism in* Sign. *Then, for all* $val : \mathcal{R} \to |\mathcal{P}'|$, *and* $t \in Term_\Sigma$

$$m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(t) = m^{val}_{\mathcal{M}'}(\sigma_{term}(t)) \ .$$

PROOF. The proof of this lemma follows by induction on the structure of the term. We will only provide a proof for the base case where $t = r$ with $r \in \mathcal{R}$, the rest of the cases follows trivially by using the induction hypothesis.

$$\begin{array}{rl} & m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(r) \\ = & val(r) \\ & \qquad \text{[by Definition 2.51]} \\ = & val(\sigma_{term}(r)) \\ & \qquad \text{[by Definition 4.1]} \\ = & m^{val}_{\mathcal{M}'}(\sigma_{term}(r)) \\ & \qquad \text{[by Definition 2.51]} \end{array}$$

∎

LEMMA 4.16. *Let* $\Sigma, \Sigma' \in |\mathsf{Sign}|$, $\sigma : \Sigma \to \Sigma'$ *be a morphism in* Sign, $t_1, t_2 \in Term_\Sigma$ *and* $\mathcal{M}' \in |\mathbf{Mod}(\Sigma')|$. *Then, for all* $val : \mathcal{R} \to |\mathcal{P}'|$ *the following conditions are equivalent:*

(1) $m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(t_1) = m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(t_2)$.
(2) $m^{val}_{\mathcal{M}'}(\sigma_{term}(t_1)) = m^{val}_{\mathcal{M}'}(\sigma_{term}(t_2))$.

PROOF. We first prove that Condition 1 implies Condition 2.

$$\begin{array}{rl} & m^{val}_{\mathcal{M}'}(\sigma_{term}(t_1)) \\ = & m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(t_1) \\ & \qquad \text{[by Lemma 4.15]} \\ = & m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(t_2) \\ & \qquad \text{[by hypothesis]} \\ = & m^{val}_{\mathcal{M}'}(\sigma_{term}(t_2)) \\ & \qquad \text{[by Lemmas 4.15]} \end{array}$$

The proof of the other implication is analogous to this one. ∎

LEMMA 4.17. ($\models$-invariance condition)
Let $\Sigma, \Sigma' \in |\mathsf{Sign}|$ *and* $\sigma : \Sigma \to \Sigma'$ *be a morphism in* Sign, $t_1 = t_2 \in \mathbf{Sen}(\Sigma)$ *and* $\mathcal{M}' \in |\mathbf{Mod}(\Sigma')|$. *Then,*

$$\mathbf{Mod}(\sigma)(\mathcal{M}') \models^{\Sigma} t_1 = t_2 \ \ \textit{iff} \ \ \mathcal{M}' \models^{\Sigma'} \mathbf{Sen}(\sigma)(t_1 = t_2) \ .$$

PROOF.

$$\begin{array}{ll}
& \mathbf{Mod}(\sigma)(\mathcal{M}') \models^{\Sigma} t_1 = t_2 \\
\text{iff} & \text{for all } val : \mathcal{R} \to |\mathcal{P}'|,\ m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(t_1) = m^{val}_{\mathbf{Mod}(\sigma)(\mathcal{M}')}(t_2) \\
& \qquad \text{[by Definition 2.52]} \\
\text{iff} & \text{for all } val : \mathcal{R} \to |\mathcal{P}'|,\ m^{val}_{\mathcal{M}'}(\sigma_{term}(t_1)) = m^{val}_{\mathcal{M}'}(\sigma_{term}(t_2)) \\
& \qquad \text{[by Lemma 4.16]} \\
\text{iff} & \mathcal{M}' \models^{\Sigma'} \sigma_{term}(t_1) = \sigma_{term}(t_2) \\
& \qquad \text{[by Definition 2.52]} \\
\text{iff} & \mathcal{M}' \models^{\Sigma'} \sigma_{eq}(t_1 = t_2) \\
& \qquad \text{[by Definition 4.2]} \\
\text{iff} & \mathcal{M}' \models^{\Sigma'} \mathbf{Sen}(\sigma)(t_1 = t_2) \\
& \qquad \text{[by Definition 4.3]}
\end{array}$$

■

THEOREM 4.1. (fPCFAU-signatures, sentences and models form an institution)
$\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ *is an institution.*

PROOF. The proof follows from Lemmas 4.1, 4.8, 4.14 and 4.17. ■

The institution behind full proper closure fork algebras with urelements is denoted as $\mathscr{I}$.

2. The entailment system behind full proper closure fork algebras with urelements

In this section we use a standard model theoretic construction [**Mes89**, Proposition 4] in order to build a candidate entailment system. This entailment system, though it defines an entailment relation, does not guarantee the existence of axioms and proof rules implementing the deduction relation. We will address this issue later on in §3 when we will provide a proof calculus for this deduction relation.

DEFINITION 4.7. Let $\Sigma \in |\mathsf{Sign}|$. Let $\Gamma \subseteq \mathbf{Sen}(\Sigma)$. We define the category $\mathbf{Mod}(\Sigma, \Gamma)$ as the full subcategory of $\mathbf{Mod}(\Sigma)$ determined by those models $\mathcal{M} \in |\mathbf{Mod}(\Sigma)|$ that satisfy all the sentences in Γ, i.e., $\mathcal{M} \models^{\Sigma} \phi$ for each $\phi \in \Gamma$.

We also define a relation between sets of sentences and sentences $\Vdash^{\Sigma}$, as follows:

$$\Gamma \Vdash^{\Sigma} \phi \ \text{ iff } \ \mathcal{M} \models^{\Sigma} \phi \ \text{ for each } \mathcal{M} \in |\mathbf{Mod}(\Sigma, \Gamma)| \ .$$

Then, we can prove the following theorem.

THEOREM 4.2. (fPCFAU-signatures and sentences form an entailment system)
$\langle \mathsf{Sign}, \mathbf{Sen}, \{\Vdash^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ *is an entailment system.*

PROOF. By Lemma 4.1, Sign is a category and by Lemma 4.8 **Sen** is a functor. Finally, by [**Mes89**, Proposition 4], given $\Sigma \in |\mathsf{Sign}|$, $\Vdash^{\Sigma}$ satisfies the conditions presented in Definition 3.20. ■

The entailment system behind full proper closure fork algebras with urelements is denoted as $\mathscr{I}^+$.

At this point, it is possible to prove that $\mathscr{I}$ and $\mathscr{I}^+$ form a logic in the sense of Definition 3.24.

THEOREM 4.3. ($\mathscr{I}$ and $\mathscr{I}^+$ form a logic)
$\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\Vdash^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \{\models^\Sigma\}_{\Sigma \in |\mathsf{Sign}|} \rangle$ *is a logic.*

PROOF. The proof follows as a consequence of Theorems 4.1 and 4.2; and observing that, by Definition 4.7, $\Vdash^\Sigma$ is a sound and complete entailment relation. ■

The logic behind full proper closure fork algebras with urelements will be denoted as $\mathscr{L}$.

3. The proof calculus behind full proper closure fork algebras with urelements

Having proved the existence of a sound and complete entailment relation $\Vdash^\Sigma$ in the way we did, is of little interest. The entailment relation does not give any hints as to how to deduce properties, what would be the axioms, or what are the proof rules employed in order to generate the relation. Actually, it might be the case that no deduction mechanism is available. Fortunately, as was shown in Theorem 2.15, this is not the case when working with closure fork algebras because ω-CCFAU is a complete calculus for fPCFAU.

Now we will develop, in the sense of Definition 3.25, the proof calculus presented in Definition 2.47.

We first define Struct_{PC}, the subcategory of SMCat whose objects are those strict monoidal categories whose monoid of objects is the subsets of ω-CCFAU-equations on a given signature $\Sigma \in |\mathsf{Sign}|$. We consider $\cup$ as the binary operations and $\emptyset$ as the neuter element.

LEMMA 4.18. *Let $\Sigma \in |\mathsf{Sign}|$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$, we define $Str^{\Sigma,\Gamma}$ as the structure $\langle 2^{\mathbf{Sen}(\Sigma)}, \mathcal{A}^{\Sigma,\Gamma} \rangle$ where:*

$$\mathcal{A}^{\Sigma,\Gamma} = \left\{ \alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}} \;\middle|\; \begin{array}{c} \alpha = \{\alpha_j\}_{j\in\mathcal{J}} \text{ such that} \\ \alpha_j \text{ is a proof tree of } \{A_i\}_{i\in\mathcal{I}} \cup \Gamma \vdash_{\omega\text{-CCFAU}} B_j \\ \text{, for } j \in \mathcal{J} \end{array} \right\},$$

- *$id_{\{A_i\}_{i\in\mathcal{I}}} : \{A_i\}_{i\in\mathcal{I}} \to \{A_i\}_{i\in\mathcal{I}} \in \mathcal{A}^{\Sigma,\Gamma}$ denotes the set of formulas $\{A_i\}_{i\in\mathcal{I}}$ seen as proof trees.*
- *If $\alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}, \beta : \{B_j\}_{j\in\mathcal{J}} \to \{C_k\}_{k\in\mathcal{K}} \in \mathcal{A}^{\Sigma}$, $\alpha = \{\alpha_j\}_{j\in\mathcal{J}}$ and $\beta = \{\beta_k\}_{k\in\mathcal{K}}$, then $\alpha \circ \beta = \{\gamma_k\}_{k\in\mathcal{K}}$ such that γ_k is the proof tree obtained from β_k by gluing α_j at each occurrence of B_j as a leaf in β_k for each $k \in \mathcal{K}$ and $j \in \mathcal{J}$.*

Then $Str^{\Sigma,\Gamma}$ is a category.

PROOF. The proof of this lemma follows by observing that if $\{A_i\}_{i\in\mathcal{I}} \subseteq \mathbf{Sen}(\Sigma)$ then, $id_{\{A_i\}_{i\in\mathcal{I}}} \in \mathcal{A}^{\Sigma,\Gamma}$, and that if $\alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}} \in \mathcal{A}^{\Sigma,\Gamma}$ and $\beta : \{B_j\}_{j\in\mathcal{J}} \to \{C_k\}_{k\in\mathcal{K}} \in \mathcal{A}^{\Sigma,\Gamma}$, then $\alpha \circ \beta : \{A_i\}_{i\in\mathcal{I}} \to \{C_k\}_{k\in\mathcal{K}} \in \mathcal{A}^{\Sigma,\Gamma}$.

It is easy to see that, from definition of $\circ$, given $\{A_i\}_{i\in\mathcal{I}}, \{B_j\}_{j\in\mathcal{J}} \in |Str^{\Sigma,\Gamma}|$ and $\alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$ a morphism in $Str^{\Sigma,\Gamma}$, $id_{\{A_i\}_{i\in\mathcal{I}}} \circ \alpha = \alpha = \alpha \circ id_{\{B_j\}_{j\in\mathcal{J}}}$, and that $\circ$ is associative. ■

Notice that if $\Sigma \in |\mathsf{Sign}|$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$, a morphism of the form $\alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$ in $Str^{\Sigma,\Gamma}$ denotes a set of proof trees. Each tree is a proof of one of the sentences $\{B_j\}_{j\in\mathcal{J}}$ from the set of hypothesis $\{A_i\}_{i\in\mathcal{I}} \cup \Gamma$.

Lemma 4.19. *Let $\Sigma \in |\mathsf{Sign}|$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$. We define the operator $\cup : Str^{\Sigma,\Gamma} \times Str^{\Sigma,\Gamma} \to Str^{\Sigma,\Gamma}$ as follows:*

- *$\cup$ is set union when applied to subsets of $\mathbf{Sen}(\Sigma)$, and*
- *let $\alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$ and $\alpha' : \{A'_i\}_{i\in\mathcal{I}'} \to \{B'_j\}_{j\in\mathcal{J}'}$ be morphisms in $Str^{\Sigma,\Gamma}$, then $\alpha \cup \alpha' : \{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'} \to \{B_j\}_{j\in\mathcal{J}} \cup \{B'_j\}_{j\in\mathcal{J}'}$ denotes the morphism consisting of the union of the proof trees denoted by α and α'.*

Then $\cup : Str^{\Sigma,\Gamma} \times Str^{\Sigma,\Gamma} \to Str^{\Sigma,\Gamma}$ is a bifunctor.

Proof. To prove that $\cup$ is a bifunctor we first prove that $\cup$ preserves identities.

$$\begin{array}{rl} & id_{\{A_i\}_{i\in\mathcal{I}}} \cup id_{\{A'_i\}_{i\in\mathcal{I}'}} \\ = & \{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'} \\ & \quad\text{[by Lemma 4.18]} \\ = & id_{\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}} \\ & \quad\text{[by Lemma 4.18]} \end{array}$$

Then, it only rests to prove that $\bullet \cup \bullet$ preserves composition of morphisms. Let $f : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}} \in \mathcal{A}^{\Sigma,\Gamma}$, $g : \{B_j\}_{j\in\mathcal{J}} \to \{C_k\}_{k\in\mathcal{K}} \in \mathcal{A}^{\Sigma,\Gamma}$, $f' : \{A'_i\}_{i\in\mathcal{I}'} \to \{B'_j\}_{j\in\mathcal{J}'} \in \mathcal{A}^{\Sigma,\Gamma}$, $g' : \{B'_j\}_{j\in\mathcal{J}'} \to \{C'_k\}_{k\in\mathcal{K}'} \in \mathcal{A}^{\Sigma,\Gamma}$.

$$\begin{array}{rl}
& f \circ g \cup f' \circ g' \\
= & \left(\left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \\ \hline \vdots \\ \hline B_j \end{array} \right\}_{j\in\mathcal{J}} \circ \left\{ \begin{array}{c} \{B_j\}_{j\in\mathcal{J}} \\ \hline \vdots \\ \hline C_k \end{array} \right\}_{k\in\mathcal{K}} \right) \cup \\
& \qquad \left(\left\{ \begin{array}{c} \{A'_i\}_{i\in\mathcal{I}'} \\ \hline \vdots \\ \hline B'_j \end{array} \right\}_{j\in\mathcal{J}'} \circ \left\{ \begin{array}{c} \{B'_j\}_{j\in\mathcal{J}'} \\ \hline \vdots \\ \hline C'_k \end{array} \right\}_{k\in\mathcal{K}'} \right) \\
= & \left\{ \begin{array}{c} \left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \\ \hline \vdots \\ \hline B_j \end{array} \right\}_{j\in\mathcal{J}} \\ \hline \vdots \\ \hline C_k \end{array} \right\}_{k\in\mathcal{K}} \cup \left\{ \begin{array}{c} \left\{ \begin{array}{c} \{A'_i\}_{i\in\mathcal{I}'} \\ \hline \vdots \\ \hline B'_j \end{array} \right\}_{j\in\mathcal{J}'} \\ \hline \vdots \\ \hline C'_k \end{array} \right\}_{k\in\mathcal{K}'} \\
& \text{[by Lemma 4.18]} \\
= & \left\{ \begin{array}{c} \left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'} \\ \hline \vdots \\ \hline B_j \end{array} \right\}_{j\in\mathcal{J}} \cup \left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'} \\ \hline \vdots \\ \hline B'_j \end{array} \right\}_{j\in\mathcal{J}'} \\ \hline \vdots \\ \hline C_k \end{array} \right\}_{k\in\mathcal{K}} \cup \\
& \quad \left\{ \begin{array}{c} \left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'} \\ \hline \vdots \\ \hline B_j \end{array} \right\}_{j\in\mathcal{J}} \cup \left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'} \\ \hline \vdots \\ \hline B'_j \end{array} \right\}_{j\in\mathcal{J}'} \\ \hline \vdots \\ \hline C'_k \end{array} \right\}_{k\in\mathcal{K}'} \\
& \text{[by monotonicity of } \omega\text{-}\mathsf{CCFAU}]
\end{array}$$

On the other hand,

$$
\begin{array}{ll}
& (f \cup f') \circ (g \cup g') \\
= & \left(\left\{ \dfrac{\{A_i\}_{i\in\mathcal{I}}}{\dfrac{\vdots}{B_j}} \right\}_{j\in\mathcal{J}} \cup \left\{ \dfrac{\{A'_i\}_{i\in\mathcal{I}'}}{\dfrac{\vdots}{B'_j}} \right\}_{j\in\mathcal{J}'} \right) \circ \\
& \qquad \left(\left\{ \dfrac{\{B_j\}_{j\in\mathcal{J}}}{\dfrac{\vdots}{C_k}} \right\}_{k\in\mathcal{K}} \cup \left\{ \dfrac{\{B'_j\}_{j\in\mathcal{J}'}}{\dfrac{\vdots}{C'_k}} \right\}_{k\in\mathcal{K}'} \right) \\
= & \left(\left\{ \dfrac{\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}}{\dfrac{\vdots}{B_j}} \right\}_{j\in\mathcal{J}} \cup \left\{ \dfrac{\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}}{\dfrac{\vdots}{B'_j}} \right\}_{j\in\mathcal{J}'} \right) \circ \\
& \qquad \left(\left\{ \dfrac{\{B_j\}_{j\in\mathcal{J}} \cup \{B'_j\}_{j\in\mathcal{J}'}}{\dfrac{\vdots}{C_k}} \right\}_{k\in\mathcal{K}} \cup \left\{ \dfrac{\{B_j\}_{j\in\mathcal{J}} \cup \{B'_j\}_{j\in\mathcal{J}'}}{\dfrac{\vdots}{C'_k}} \right\}_{k\in\mathcal{K}'} \right) \\
& \text{[by monotonicity of } \omega\text{-CCFAU]} \\
= & \left\{ \dfrac{\left\{ \dfrac{\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}}{\dfrac{\vdots}{B_j}} \right\}_{j\in\mathcal{J}} \cup \left\{ \dfrac{\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}}{\dfrac{\vdots}{B'_j}} \right\}_{j\in\mathcal{J}'}}{\dfrac{\vdots}{C_k}} \right\}_{k\in\mathcal{K}} \cup \\
& \qquad \left\{ \dfrac{\left\{ \dfrac{\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}}{\dfrac{\vdots}{B_j}} \right\}_{j\in\mathcal{J}} \cup \left\{ \dfrac{\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}}{\dfrac{\vdots}{B'_j}} \right\}_{j\in\mathcal{J}'}}{\dfrac{\vdots}{C'_k}} \right\}_{k\in\mathcal{K}'} \\
& \text{[by Lemma 4.18]}
\end{array}
$$

■

LEMMA 4.20. *Let* $\Sigma \in |\mathsf{Sign}|$ *and* $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ *then:*

- $\cup : \langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle \times \langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle \rightarrow \langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle$ *is associative,*
- $\emptyset \in |Str^{\Sigma,\Gamma}|$ *is left and right identity for* $\cup$.

PROOF. This proof follows by showing that $\langle |Str^{\Sigma,\Gamma}|, \cup, \emptyset \rangle$ is a monoid (which is trivial by set theory); and that $\langle \{\alpha : Str^{\Sigma,\Gamma} \rightarrow Str^{\Sigma,\Gamma}\}, \cup, \emptyset \rightarrow \emptyset \rangle$ is also a monoid.

Let us first prove that $\cup$ is associative.

$$
\begin{array}{ll}
& (\{A_i\}_{i\in\mathcal{I}} \rightarrow \{B_j\}_{j\in\mathcal{J}} \cup \{A'_i\}_{i\in\mathcal{I}'} \rightarrow \{B'_j\}_{j\in\mathcal{J}'}) \cup \{A''_i\}_{i\in\mathcal{I}''} \rightarrow \{B''_j\}_{j\in\mathcal{J}''} \\
= & ((\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}) \rightarrow (\{B_j\}_{j\in\mathcal{J}} \cup \{B'_j\}_{j\in\mathcal{J}'})) \cup \{A''_i\}_{i\in\mathcal{I}''} \rightarrow \{B''_j\}_{j\in\mathcal{J}''} \\
& \quad \text{[by Lemma 4.18]} \\
= & ((\{A_i\}_{i\in\mathcal{I}} \cup \{A'_i\}_{i\in\mathcal{I}'}) \cup \{A''_i\}_{i\in\mathcal{I}''}) \rightarrow ((\{B_j\}_{j\in\mathcal{J}} \cup \{B'_j\}_{j\in\mathcal{J}'}) \cup \{B''_j\}_{j\in\mathcal{J}''}) \\
& \quad \text{[by Lemma 4.18]} \\
= & (\{A_i\}_{i\in\mathcal{I}} \cup (\{A'_i\}_{i\in\mathcal{I}'} \cup \{A''_i\}_{i\in\mathcal{I}''})) \rightarrow (\{B_j\}_{j\in\mathcal{J}} \cup (\{B'_j\}_{j\in\mathcal{J}'} \cup \{B''_j\}_{j\in\mathcal{J}''})) \\
& \quad \text{[because } \langle |Str^{\Sigma,\Gamma}|, \cup, \emptyset \rangle \text{ is a monoid]} \\
= & \{A_i\}_{i\in\mathcal{I}} \rightarrow \{B_j\}_{j\in\mathcal{J}} \cup ((\{A'_i\}_{i\in\mathcal{I}'} \cup \{A''_i\}_{i\in\mathcal{I}''}) \rightarrow (\{B'_j\}_{j\in\mathcal{J}'} \cup \{B''_j\}_{j\in\mathcal{J}''})) \\
& \quad \text{[by Lemma 4.18]} \\
= & \{A_i\}_{i\in\mathcal{I}} \rightarrow \{B_j\}_{j\in\mathcal{J}} \cup (\{A'_i\}_{i\in\mathcal{I}'} \rightarrow \{B'_j\}_{j\in\mathcal{J}'} \cup \{A''_i\}_{i\in\mathcal{I}''} \rightarrow \{B''_j\}_{j\in\mathcal{J}''}) \\
& \quad \text{[by Lemma 4.18]}
\end{array}
$$

Now we prove that $\emptyset \to \emptyset$ is a left and right identity for $\cup$.

$$\begin{array}{rl} & (\emptyset \to \emptyset) \cup (\{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}) \\ = & (\emptyset \cup \{A_i\}_{i\in\mathcal{I}}) \to (\emptyset \cup \{B_j\}_{j\in\mathcal{J}}) \\ & \qquad \text{[by Lemma 4.18]} \\ = & \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}} \\ & \qquad \text{[because } \langle |Str^{\Sigma,\Gamma}|, \cup, \emptyset\rangle \text{ is a monoid]} \end{array}$$

The proof that $\emptyset \to \emptyset$ is a right identity for $\cup$ is analogous to the previous one. ■

LEMMA 4.21. *Let $\Sigma \in |\mathsf{Sign}|$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ then $\langle Str^{\Sigma,\Gamma}, \cup, \emptyset\rangle$ is a strict monoidal category.*

PROOF. The proof follows from Lemmas 4.19 and 4.20. ■

To define the functor $\mathbf{P} : \mathsf{Th}_0 \to \mathsf{Struct}_{PC}$ we will first provide some preliminary definitions and lemmas.

DEFINITION 4.8. Let $\sigma : \langle\Sigma,\Gamma\rangle \to \langle\Sigma',\Gamma'\rangle$ be a morphism in $|\mathsf{Th}_0|$, we define $\widehat{\sigma} : Str^{\Sigma,\Gamma} \to Str^{\Sigma',\Gamma'}$ as:

- $\widehat{\sigma}(\{A_i\}_{i\in\mathcal{I}}) = \{\sigma_{eq}(A_i)\}_{i\in\mathcal{I}}$,
- $\widehat{\sigma}(\Gamma \to \{B_j\}_{j\in\mathcal{J}}) = \{\widehat{\sigma}(\Gamma \to B_j)\}_{j\in\mathcal{J}}$,

where:

$$\widehat{\sigma}\left(\frac{\left\{ \begin{array}{c} \Gamma_i \\ \hline \vdots \pi_i \end{array} \right\}_{i\in\mathcal{I}}}{\alpha}\,(r) \right) = \frac{\left\{ \widehat{\sigma}\left(\begin{array}{c} \Gamma_i \\ \hline \vdots \pi_i \end{array} \right) \right\}_{i\in\mathcal{I}}}{\sigma_{eq}(\alpha)}\,(r) \quad \text{, for } \mathcal{I} \subseteq \mathbb{N}\ .$$

In the introduction of this chapter we argued in favor of building the logical system of the full proper closure fork algebras with urelements on the base that we needed to include a new proof rule, the ω-rule. The previous definition hides that need. Even when it does not depend on the particular rules, the potential existence of an infinite number of proof sub-trees to which the ω-rule is applied, is expressed in the last clause when we define the result of applying $\widehat{\sigma}$ to single proof tree.

LEMMA 4.22. *Let $\langle\Sigma,\Gamma\rangle \in |\mathsf{Th}_0|$ and $\{A_i\}_{i\in\mathcal{I}} \cup \{B_j\}_{j\in\mathcal{J}} \subseteq \mathbf{Sen}(\Sigma)$. Then, for all $\alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$, $\widehat{id_{\langle\Sigma,\Gamma\rangle}}(\alpha) = \alpha$.*

PROOF. The proof of this lemma follows by induction on the height of the proof tree by using Lemma 4.5. ■

LEMMA 4.23. *Let $\sigma : \langle\Sigma,\Gamma\rangle \to \langle\Sigma',\Gamma'\rangle, \sigma' : \langle\Sigma',\Gamma'\rangle \to \langle\Sigma'',\Gamma''\rangle$ be morphisms in $|\mathsf{Th}_0|$, and let $\{A_i\}_{i\in\mathcal{I}} \cup \{B_j\}_{j\in\mathcal{J}} \subseteq \mathbf{Sen}(\Sigma)$. Then, for all $\alpha : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$, $\widehat{\sigma \circ \sigma'}(\alpha) = (\widehat{\sigma} \circ \widehat{\sigma'})(\alpha)$.*

PROOF. The proof of this lemma follows by induction on the height of the proof tree by using Lemma 4.6. ■

LEMMA 4.24. *Let $\sigma : \langle\Sigma,\Gamma\rangle \to \langle\Sigma',\Gamma'\rangle$ be a morphism in $|\mathsf{Th}_0|$, and let $f : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$ and $g : \{B_j\}_{j\in\mathcal{J}} \to C$ be morphisms in $Str^{\Sigma,\Gamma}$.*

Then

$$\widehat{\sigma}\left(\begin{array}{c}\left\{\begin{array}{c}\underline{\{A_i\}_{i\in\mathcal{I}}}\\ \vdots\\ \overline{B_j}\end{array}\right\}_{j\in\mathcal{J}}\\ \hline \vdots\\ \hline C\end{array}\,(r)\right) = \begin{array}{c}\widehat{\sigma}\left(\left\{\begin{array}{c}\underline{\{A_i\}_{i\in\mathcal{I}}}\\ \vdots\\ \overline{B_j}\end{array}\right\}_{j\in\mathcal{J}}\right)\\ \hline \widehat{\sigma}\left(\begin{array}{c}\vdots\\ \hline C\end{array}\right)\end{array}\,(r) .$$

Proof. The proof of this lemma follows by induction on the height of the proof of C from the conclusions of applying rule r. Let π be such proof. If π has height 1, i.e. C is the conclusion of applying rule r, then the proof follows trivially by applying Definition 4.8.

If π has height greater than 1, then the following reasoning can be applied to prove the lemma.

$$\begin{array}{cl} & \widehat{\sigma}\left(\begin{array}{c}\left\{\begin{array}{c}\underline{\{A_i\}_{i\in\mathcal{I}}}\\ \vdots\\ \overline{B_j}\end{array}\right\}_{j\in\mathcal{J}}\\ \hline \vdots\\ \hline C\end{array}\,(r)\right)\\ = & \widehat{\sigma}\left(\begin{array}{c}\left\{\begin{array}{c}\underline{\{A_i\}_{i\in\mathcal{I}}}\\ \vdots\\ \overline{B_j}\end{array}\right\}_{j\in\mathcal{J}}\\ \hline \vdots\\ \hline \sigma_{eq}(C)\end{array}\,(r)\right)\\ & \text{[by Definition 4.8]}\\ = & \begin{array}{c}\widehat{\sigma}\left(\left\{\begin{array}{c}\underline{\{A_i\}_{i\in\mathcal{I}}}\\ \vdots\\ \overline{B_j}\end{array}\right\}_{j\in\mathcal{J}}\right)\\ \hline \widehat{\sigma}\left(\begin{array}{c}\vdots\\ \hline \sigma_{eq}(C)\end{array}\right)\end{array}\,(r)\\ & \text{[by inductive hypothesis]}\\ = & \begin{array}{c}\widehat{\sigma}\left(\left\{\begin{array}{c}\underline{\{A_i\}_{i\in\mathcal{I}}}\\ \vdots\\ \overline{B_j}\end{array}\right\}_{j\in\mathcal{J}}\right)\\ \hline \widehat{\sigma}\left(\begin{array}{c}\vdots\\ \hline C\end{array}\right)\end{array}\,(r)\\ & \text{[by Definition 4.8]}\end{array}$$

■

LEMMA 4.25. *Let $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ be a morphism in $|\mathsf{Th_0}|$ and let $f : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$ and $g : \{B_j\}_{j\in\mathcal{J}} \to \{C_k\}_{k\in\mathcal{K}}$ be morphisms in $Str^{\Sigma,\Gamma}$. Then*

$$\left\{ \frac{\widehat{\sigma}\left(\left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \\ \hline \vdots \\ \hline B_j \end{array} \right\}_{j\in\mathcal{J}}\right)}{\widehat{\sigma}\left(\begin{array}{c} \vdots \\ \hline C_k \end{array} \right)} \right\}_{k\in\mathcal{K}} = \widehat{\sigma}\left(\left\{ \begin{array}{c} \{A_i\}_{i\in\mathcal{I}} \\ \hline \vdots \\ \hline B_j \end{array} \right\}_{j\in\mathcal{J}}\right) \circ \widehat{\sigma}\left(\left\{ \begin{array}{c} \{B_j\}_{j\in\mathcal{J}} \\ \hline \vdots \\ \hline C_k \end{array} \right\}_{k\in\mathcal{K}}\right).$$

PROOF. The proof of this lemma is analogous to the proof of Lemma 4.24 using the definition of $\circ$ presented in Lemma 4.18. ■

LEMMA 4.26. *Let $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ be a morphism in $|\mathsf{Th_0}|$, then $\widehat{\sigma}$ is a functor.*

PROOF. By Definition 4.8 we know that $\widehat{\sigma}$ yields $Str^{\Sigma',\Gamma'}$-objects whenever it is applied to $Str^{\Sigma,\Gamma}$-objects and $Str^{\Sigma',\Gamma'}$-morphisms whenever it is applied to $Str^{\Sigma,\Gamma}$-morphisms.

We will first prove that $\widehat{\sigma}(id_{\{A_i\}_{i\in\mathcal{I}}}) = id_{\widehat{\sigma}(\{A_i\}_{i\in\mathcal{I}})}$.

$$\begin{array}{rl}
 & \widehat{\sigma}(id_{\{A_i\}_{i\in\mathcal{I}}}) \\
= & \widehat{\sigma}\left(\{A_i\}_{i\in\mathcal{I}}\right) \\
 & \qquad \text{[by Lemma 4.18]} \\
= & \{\sigma_{eq}(A_i)\}_{i\in\mathcal{I}} \\
 & \qquad \text{[by Definition 4.8]} \\
= & id_{\{\sigma_{eq}(A_i)\}_{i\in\mathcal{I}}} \\
 & \qquad \text{[by Lemma 4.18]} \\
= & id_{\widehat{\sigma}(\{A_i\}_{i\in\mathcal{I}})} \\
 & \qquad \text{[by Definition 4.8]}
\end{array}$$

Next, we prove that if $f : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$ and $g : \{B_j\}_{j\in\mathcal{J}} \to \{C_k\}_{k\in\mathcal{K}}$ are morphisms in $\langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle$, then $\widehat{\sigma}(f \circ g) = \widehat{\sigma}(f) \circ \widehat{\sigma}(g)$.

Let $f : \{A_i\}_{i\in\mathcal{I}} \to \{B_j\}_{j\in\mathcal{J}}$ and $g : \{B_j\}_{j\in\mathcal{J}} \to \{C_k\}_{k\in\mathcal{K}}$.

$$\begin{array}{rl}
& \widehat{\sigma}(f \circ g) \\
= & \widehat{\sigma}\left(\left\{\dfrac{\{A_i\}_{i\in\mathcal{I}}}{\dfrac{\vdots}{B_j}}\right\}_{j\in\mathcal{J}} \circ \left\{\dfrac{\{B_j\}_{j\in\mathcal{J}}}{\dfrac{\vdots}{C_k}}\right\}_{k\in\mathcal{K}}\right) \\
= & \widehat{\sigma}\left(\left\{\dfrac{\left\{\dfrac{\{A_i\}_{i\in\mathcal{I}}}{\dfrac{\vdots}{B_j}}\right\}_{j\in\mathcal{J}}}{\dfrac{\vdots}{C_k}}\right\}_{k\in\mathcal{K}}\right) \\
& \text{[by Lemma 4.18]} \\
= & \left\{\widehat{\sigma}\left(\dfrac{\left\{\dfrac{\{A_i\}_{i\in\mathcal{I}}}{\dfrac{\vdots}{B_j}}\right\}_{j\in\mathcal{J}}}{\dfrac{\vdots}{C_k}}\right)\right\}_{k\in\mathcal{K}} \\
& \text{[by Definition 4.8]} \\
= & \left\{\dfrac{\widehat{\sigma}\left(\left\{\dfrac{\{A_i\}_{i\in\mathcal{I}}}{\dfrac{\vdots}{B_j}}\right\}_{j\in\mathcal{J}}\right)}{\widehat{\sigma}\left(\dfrac{\vdots}{C_k}\right)}\right\}_{k\in\mathcal{K}} \\
& \text{[by Lemma 4.24]} \\
= & \widehat{\sigma}\left(\left\{\dfrac{\{A_i\}_{i\in\mathcal{I}}}{\dfrac{\vdots}{B_j}}\right\}_{j\in\mathcal{J}}\right) \circ \widehat{\sigma}\left(\left\{\dfrac{\{B_j\}_{j\in\mathcal{J}}}{\dfrac{\vdots}{C_k}}\right\}_{k\in\mathcal{K}}\right) \\
= & \widehat{\sigma}(f) \circ \widehat{\sigma}(g) \\
& \text{[by Lemma 4.25]}
\end{array}$$

■

LEMMA 4.27. *Let* $\mathsf{Struct}_{PC} = \langle \mathcal{O}, \mathcal{A} \rangle$ *where:*

- $\mathcal{O} = \left\{ \langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle \in |\mathsf{SMCat}| \,\middle|\, \Sigma \in |\mathsf{Sign}| \text{ and } \Gamma \subseteq \mathbf{Sen}(\Sigma) \right\}$,
- $\mathcal{A} = \Big\{ \widehat{\sigma} : \langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle \to \langle Str^{\Sigma',\Gamma'}, \cup, \emptyset \rangle \mid$
  $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ *is a morphism in* $|\mathsf{Th}_0|\}$.

Then Struct_{PC} *is a category.*

PROOF. The proof easily follows from Lemma 4.26. ■

DEFINITION 4.9. Let $\langle \Sigma, \Gamma \rangle, \langle \Sigma', \Gamma' \rangle \in |\mathsf{Th}_0|$ and $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ be a morphism in Th_0, we define $\mathbf{P} : \mathsf{Th}_0 \to \mathsf{Struct}_{PC}$ as:

- $\mathbf{P}(\langle \Sigma, \Gamma \rangle) = \langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle$,

- $\mathbf{P}(\sigma) = \widehat{\sigma}$.

LEMMA 4.28. $\mathbf{P}$ *is a functor.*

PROOF. By Definition 4.9 we know that if $T \in |\mathsf{Th}_0|$, then $\mathbf{P}(T) \in \mathsf{Struct}_{PC}$. By Lemma 4.26 we also know that if $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ is a morphism in Th_0, then $\mathbf{P}(\sigma) : \langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle \to \langle Str^{\Sigma',\Gamma'}, \cup, \emptyset \rangle$ is a functor (i.e. a morphism in Struct_{PC}).

Lemma 4.22 proves that $\mathbf{P}(id_{\langle \Sigma, \Gamma \rangle}) = id_{\langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle}$. Then, by Lemma 4.23, if $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ and $\sigma' : \langle \Sigma', \Gamma' \rangle \to \langle \Sigma'', \Gamma'' \rangle$ are morphisms in Th_0 then $\mathbf{P}(\sigma \circ \sigma') = \mathbf{P}(\sigma) \circ \mathbf{P}(\sigma')$. ∎

DEFINITION 4.10. Let $\langle \Sigma, \Gamma \rangle, \langle \Sigma', \Gamma' \rangle \in |\mathsf{Th}_0|$ and $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ be a morphism in Th_0. We define $\mathbf{Pr} : \mathsf{Struct}_{PC} \to \mathsf{Set}$ as follows:

- $\mathbf{Pr}(\langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle) = \left\{ \alpha : \emptyset \to A \;\middle|\; \alpha \text{ is a morphism in } Str^{\Sigma,\Gamma} \right\}$,
- $\mathbf{Pr}(\widehat{\sigma}) = \widehat{\sigma}$.

LEMMA 4.29. $\mathbf{Pr}$ *is a functor.*

PROOF. Recalling on Definition 4.10, we know that if $\langle \Sigma, \Gamma \rangle \in |\mathsf{Th}_0|$, then $\mathbf{Pr}(\langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle)$ yields a set and that if $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ is a morphism in Th_0, then $\mathbf{Pr}(\widehat{\sigma})$ yields a total function between sets. Finally, the proof that $\mathbf{Pr}$ preserves identities ($\mathbf{Pr}(\widehat{id_{\langle \Sigma, \Gamma \rangle}}) = id_{\mathbf{Pr}(\langle Str^{\Sigma,\Gamma}, \cup, \emptyset \rangle)}$), and compositions (if $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ and $\sigma' : \langle \Sigma', \Gamma' \rangle \to \langle \Sigma'', \Gamma'' \rangle$ are morphisms in Th_0, then $\mathbf{Pr}(\widehat{\sigma \circ \sigma'}) = \mathbf{Pr}(\widehat{\sigma}) \circ \mathbf{Pr}(\widehat{\sigma'})$) follows by Lemmas 4.22 and 4.23, and observing that $\mathbf{Pr}(\widehat{\sigma}) = \widehat{\sigma}$. ∎

DEFINITION 4.11. Let $T, T' \in |\mathsf{Th}_0|$, we define $\pi_T : \mathbf{P} \circ \mathbf{Pr}(T) \to \mathbf{Sen}(T)$ as $\pi_T(\alpha) = A$, for each $\alpha : \emptyset \to A \in \mathbf{P} \circ \mathbf{Pr}(T)$.

LEMMA 4.30. π *is a natural transformation.*

PROOF. Let $\sigma : T \to T'$ be a morphism in Th_0, then the commutativity of the following diagram

$$\begin{tikzcd}
\mathbf{P} \circ \mathbf{Pr}(T) \arrow[r, "\pi_T"] \arrow[d, "\mathbf{P} \circ \mathbf{Pr}(\sigma)"] & \mathbf{Sen}(T) \arrow[d, "\mathbf{Sen}(\sigma)"] \\
\mathbf{P} \circ \mathbf{Pr}(T') \arrow[r, "\pi_{T'}"] & \mathbf{Sen}(T')
\end{tikzcd}$$

follows by observing that for any $A \in \mathbf{Sen}(T)$ such that $\emptyset \to A \in \mathbf{P} \circ \mathbf{Pr}(T)$, then $(\pi_T \circ \mathbf{Sen}(\sigma))(\emptyset \to A) = ((\mathbf{P} \circ \mathbf{Pr}(\sigma)) \circ \pi_{T'})(\emptyset \to A)$.

$$\begin{array}{rl}
 & (\pi_T \circ \mathbf{Sen}(\sigma))(\emptyset \to A) \\
= & \mathbf{Sen}(\sigma)(\pi_T(\emptyset \to A)) \\
 & \qquad \text{[by definition of } \circ \text{]} \\
= & \mathbf{Sen}(\sigma)(A) \\
 & \qquad \text{[by Definition 4.11]} \\
= & \sigma_{eq}(A) \\
 & \qquad \text{[by Lemma 4.8]}
\end{array}$$

Now we prove that $((\mathbf{P} \circ \mathbf{Pr}(\sigma)) \circ \pi_{T'})(\emptyset \to A) = \sigma_{eq}(A)$.

$$\begin{array}{rl}
& ((\mathbf{P} \circ \mathbf{Pr}(\sigma)) \circ \pi_{T'})(\emptyset \to A) \\
= & \pi_{T'}((\mathbf{P} \circ \mathbf{Pr}(\sigma))(\emptyset \to A)) \\
& \qquad \text{[by definition of } \circ] \\
= & \pi_{T'}(\mathbf{Pr}(\mathbf{P}(\sigma))(\emptyset \to A)) \\
& \qquad \text{[by definition of } \circ] \\
= & \pi_{T'}(\mathbf{Pr}(\widehat{\sigma})(\emptyset \to A)) \\
& \qquad \text{[by Lemma 4.28]} \\
= & \pi_{T'}(\widehat{\sigma}(\emptyset \to A)) \\
& \qquad \text{[by Lemma 4.29]} \\
= & \pi_{T'}\left(\widehat{\sigma}\left(\dfrac{\dfrac{\Gamma}{\vdots}}{A}\right)\right) \\
= & \pi_{T'}\left(\dfrac{\widehat{\sigma}\left(\dfrac{\Gamma}{\vdots}\right)}{\sigma_{eq}(A)}\right) \\
& \qquad \text{[by Definition 4.8]} \\
= & \sigma_{eq}(A) \\
& \qquad \text{[by Definition 4.11]}
\end{array}$$

■

THEOREM 4.4. (I^+, $\mathbf{P}$, $\mathbf{Pr}$ and π form a proof calculus)
$\langle \mathsf{Sign}, \mathbf{Sen}, \{\Vdash^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \mathbf{P}, \mathbf{Pr}, \pi \rangle$ *is a proof calculus.*

PROOF. The proof follows from Theorem 4.2 and Lemmas 4.28, 4.29 and 4.30. ■

Finally, we define the logical system behind full proper closure fork algebras with urelements by putting together the constructions carried out in §1 and §3 in the following theorem.

THEOREM 4.5. (L and ω-CCFAU form a logical system)
$\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\Vdash^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \{\models^\Sigma\}_{\Sigma \in |\mathsf{Sign}|}, \mathbf{P}, \mathbf{Pr}, \pi \rangle$ *is a logical system.*

PROOF. The proof follows from Theorems 4.3 and 4.4. ■

In this chapter we presented the logical system behind full proper closure fork algebras with urelements. This class of algebras and its calculus, ω-CCFAU, were proved to be useful in interpreting several logics. Nevertheless, there are some other logics that require more complex algebraic structures to be interpreted, and in some cases the calculus must be extended in order to completely characterize the class of algebras. To avoid obscuring the concepts and results on ω–closure fork algebras with urelements we preferred to adhere to its purer version, leaving any further extension of this logical system to the reader, who can use this chapter as a guideline in the process.

CHAPTER 5

Interpretability as a representation map between institutions

In this chapter we will address the formalization of the interpretability result of first order linear temporal logic in an extension of fork algebras presented in [**FL06**], but this time resorting to institutions. As we mentioned before, institutions can be related by means of institution morphisms or institution representations. Both kinds of mappings were extensively discussed by Tarlecki in [**Tar96**]. In this work, the author goes even further when he writes:

> "... this suggests that we should strive at a development of a convenient to use proof theory (with support tools!) for a sufficiently rich "universal" institution, and then reuse it for other institutions linked to it by institution representations."

An interpretability result of a logic into ω-CCFAU exposes the possibility to translate formulas from that logic to ω-CFAU equations in a semantics preserving way, allowing us to use the calculus to reason about properties coming from a theory written in this logic.

Our claim is that, considering the existing representability results of logics into ω-CCFAU, the latter could be considered as a good candidate to play the rôle of Tarlecki's "universal" institution.

In general, if L is a logic and Σ_L an L-signature, an interpretability result of L in ω-CCFAU is presented by resorting to:

- A mapping S from Σ_L to $\Sigma \in |\mathsf{Sign}|$.
- A translation $T_{L\to\omega\text{-}\mathsf{CFAU}}$ of Σ_L-formulas to Σ-equations.
- A mapping $M_{L\to\mathsf{fPCFAU}}$ of Σ_L-models to Σ-models (full proper closure fork algebras with urelements), satisfying:

$$\forall \mathfrak{A} \in |\mathbf{Mod}_L(\Sigma_L)| \left(\mathfrak{A} \models_L^{\Sigma_L} \alpha \text{ iff } M_{L\to\mathsf{fPCFAU}}(\mathfrak{A}) \models^{\Sigma} T_{L\to\omega\text{-}\mathsf{CFAU}}(\alpha)\right) .$$

- A mapping $M_{\mathsf{fPCFAU}\to L}$ of Σ-models (full proper closure fork algebras with urelements) to Σ_L-models, satisfying:

$$\forall \mathfrak{B} \in \mathsf{fPCFAU} \left(M_{\mathsf{fPCFAU}\to L}(\mathfrak{B}) \models_L^{\Sigma_L} \alpha \text{ iff } \mathfrak{B} \models^{\Sigma} T_{L\to\omega\text{-}\mathsf{CFAU}}(\alpha)\right) .$$

As we mentioned at the end of Chapter 4, there are some interpretability results for which it is not sufficient to consider the logical system presented in Theorem 4.5. One of this cases is the interpretability of first-order linear temporal logic. In order to show how interpretability results can be seen as representation maps between institutions we could choose a logic for which ω–closure fork algebras with urelements is sufficient, like LTL, PDL or FOL but we considered that the most interesting interpretability result we developed is the one for first-order linear temporal logic.

This chapter is organized as follows. The first section (§1) is dedicated to present first-order linear temporal logic (FOLTL for short) and review some definitions and results from the field of modal logics. Then, in §2, we review the interpretability result of FOLTL into ω-CCFAU we presented in [**FL06**]. And finally, in §3 we rephrase it as an institution representation map. This last section is structured following the order of requirements stated in Definition 3.29.

1. First-order linear temporal logic

In this section we present the interpretability result for a first-order extension of the propositional temporal logic LTL [**Eme90**]. The first-order extension of LTL we choose adopts the quantification provided in [**MP95**]. In order to interpret FOLTL, we will define a translation of FOLTL-formulas to relational expressions. We will then prove that this translation preserves (in a way to be defined) the semantics of the logic.

Definition 5.1. Let $\mathcal{V}$ be a totally ordered denumerable set of variable symbols, and let $\Sigma = \langle \{f_i\}_{I\in\mathcal{I}}, \{p_j\}_{j\in\mathcal{J}}\rangle$ be a FOLTL-signature then we define the set $Term_{\mathsf{FOLTL}}(\Sigma)$ of the FOLTL-terms on the signature Σ as the smallest set $T(\Sigma)$ satisfying:

- $v \in T(\Sigma)$ for all $v \in \mathcal{V}$,
- if $t_1, \ldots, t_n \in T(\Sigma)$, then $f_i(t_1, \ldots, t_{arity(f_i)}) \in T(\Sigma)$, for all $i \in \mathcal{I}$.

We define the set $Form_{\mathsf{FOLTL}}(\Sigma)$ of the FOLTL formulas on the signature Σ as the smallest set $F(\Sigma)$ satisfying:

- If $t_1, \ldots, t_n \in T(\Sigma)$, then $p_j(t_1, \ldots, t_{arity(p_j)}) \in F(\Sigma)$, for all $j \in \mathcal{J}$,
- if $\alpha, \beta \in F(\Sigma)$ and $v \in \mathcal{V}$, then $\{\neg\alpha, \alpha \vee \beta, \mathsf{X}\alpha, \alpha\mathsf{U}\beta, (\exists v)\alpha\} \subseteq F(\Sigma)$.

The set of FOLTL-signatures will be denoted as $Sign_{\mathsf{FOLTL}}$.

Definition 5.2. Let $\Sigma = \langle \{f_i\}_{i\in\mathcal{I}}, \{p_j\}_{j\in\mathcal{J}}\rangle \in Sign_{\mathsf{FOLTL}}$, then a Σ-structure is a structure $\mathcal{A} = \langle A, \{f_i^{\mathcal{A}}\}_{i\in\mathcal{I}}, \{p_j^{\mathcal{A}}\}_{j\in\mathcal{J}}\rangle$ such that:

- A is a nonempty set.
- $f_i^{\mathcal{A}} : A^{arity(f_i)} \to A$, for all $i \in \mathcal{I}$.
- $p_j^{\mathcal{A}} \subseteq A^{arity(p_j)}$, for all $j \in \mathcal{J}$.

Definition 5.3. Let $\Sigma \in Sign_{\mathsf{FOLTL}}$, and a Σ-structure $\mathcal{A}$, then a Kripke structure $\langle \mathcal{A}, St, St_0, T\rangle \in Mod_\Sigma$ if and only if:

- St is the set of *states* (valuations of the variables on A),
- $St_0 \subseteq St$ is the set of *initial states*, and

- $T \subseteq St \times St$ is the *transition relation*. The transition relation T is assumed to be *complete*; that is, every state has at least one successor.

Given a Kripke structure $\mathfrak{K}$, the set of paths (or traces) of $\mathfrak{K}$ is denoted by $\Delta_{\mathfrak{K}}$. A trace $tr \in \Delta_{\mathfrak{K}}$ is an infinite sequence $st_0, st_1, \ldots$ such that $st_i \in St$ and $(st_i, st_{i+1}) \in T$ for all $i \geq 0$. We denote by tr^i the suffix of tr starting at position i. Similarly, we denote by tr_i the $i^{th.}$ state in the trace tr. A v-variant of a state st ($v \in \mathcal{V}$) is a state $\widehat{st}$ that agrees with st in the value of the state variables w ($w \in \mathcal{V}, w \neq v$). This concept generalizes to traces as follows. A trace $\widehat{tr} = \widehat{st_0}, \widehat{st_1}, \ldots, \widehat{st_n}, \ldots$ is a v-variant of a trace $tr = st_0, st_1, \ldots, st_n, \ldots$ if $\widehat{st_j}$ is a v-variant of st_j for all $j \geq 0$.

In the following two definitions we provide the semantics of terms (which agrees with the semantics of terms in classical first-order logic), as well as the satisfiability relation for FOLTL-formulas. States are given by the values of the variables, i.e., a state is a valuation of the variables.

Definition 5.4. Let $\Sigma = \langle \{f_i\}_{i \in \mathcal{I}}, \{p_j\}_{j \in \mathcal{J}} \rangle \in Sign_{\mathsf{FOLTL}}$ and a Kripke structure $\langle \mathcal{A}, St, St_0, T \rangle$. Then, $V : Term_{\mathsf{FOLTL}}(\Sigma) \to (St \to A)$ is defined as follows:

- $V(v)(val) = val(v)$, for all $v \in \mathcal{V}$.
- $V(f_i(t_1, \ldots, t_n))(val) = f_i^{\mathcal{A}}(V(t_1)(val), \ldots, V(t_n)(val))$, for all $i \in \mathcal{I}$.

Definition 5.5. Let $\Sigma = \langle \{f_i\}_{i \in \mathcal{I}}, \{p_j\}_{j \in \mathcal{J}} \rangle \in Sign_{\mathsf{FOLTL}}$, and let $\alpha, \beta \in Form_{\mathsf{FOLTL}}(\Sigma)$. Let $\mathfrak{K} = \langle \mathcal{A}, St, St_0, T \rangle \in Mod_{\Sigma}$, where $\mathcal{A}$ is the structure $\langle \{f_i^{\mathcal{A}}\}_{i \in \mathcal{I}}, \{p_j^{\mathcal{A}}\}_{j \in \mathcal{J}} \rangle$, and $tr \in \Delta_{\mathfrak{K}}$, the semantics of a FOLTL-formula is defined recursively as follows:

- $\mathfrak{K}, tr \models_{\mathsf{FOLTL}} p_j(t_1, \ldots, t_n)$ iff $p_j^{\mathcal{A}}(V(t_1)(tr_0), \ldots, V(t_n)(tr_0))$,
- $\mathfrak{K}, tr \models_{\mathsf{FOLTL}} \neg\alpha$ iff $\mathfrak{K}, tr \not\models \alpha$,
- $\mathfrak{K}, tr \models_{\mathsf{FOLTL}} \alpha \vee \beta$ iff $\mathfrak{K}, tr \models \alpha$ or $\mathfrak{K}, tr \models \beta$,
- $\mathfrak{K}, tr \models_{\mathsf{FOLTL}} \mathsf{X}\alpha$ iff $\mathfrak{K}, tr^1 \models \alpha$,
- $\mathfrak{K}, tr \models_{\mathsf{FOLTL}} \alpha \mathsf{U} \beta$ iff there exists $i \in \mathbb{N}$ such that $i \geq 0$, $\mathfrak{K}, tr^i \models \beta$, and for all $j \in \mathbb{N}$ such that $0 \leq j < i$, $\mathfrak{K}, tr^j \models \alpha$, and
- $\mathfrak{K}, tr \models_{\mathsf{FOLTL}} (\exists v)\alpha$ iff $\mathfrak{K}, \widehat{tr} \models \alpha$, for some $\widehat{tr}$, v-variant of tr, where $v \in \mathcal{V}$.

A formula is satisfied in a Kripke structure $\mathfrak{K}$ if it is satisfied along a trace $tr_0, tr_1, \ldots \in \Delta_{\mathfrak{K}}$ such that $tr_0 \in St_0$. A formula is valid in a Kripke structure $\mathfrak{K}$ if it is satisfied along all traces $tr_0, tr_1, \ldots \in \Delta_{\mathfrak{K}}$ such that $tr_0 \in St_0$.

When trying to relate two Kripke structures in a way that satisfiability is preserved, the equivalent notion to the algebraic homomorphism is called *bounded morphism* (or *p-morphism*). Besides bounded morphisms there exists other possibilities, such as *homomorphism* or *strong homomorphism*, but these relations are stronger than bounded morphisms because they not only enforce modal equivalence, but also establish structural restrictions on the accessibility relation which are not required to preserve semantic equivalence of terms and formulas. For a complete explanation on why, in this case, the best choice are *bounded morphisms* we point the reader to [**BdV01**, §2].

The next definition extends the definition of bounded morphism given in [**BdV01**, Definition 2.10] to the class of Kripke structures used to give semantics to FOLTL.

DEFINITION 5.6. Let $\Sigma \in Sign_{\mathsf{FOLTL}}$, and $\mathfrak{K}_1 = \langle \mathcal{A}_1, St_1, St_{01}, T_1 \rangle$, $\mathfrak{K}_2 = \langle \mathcal{A}_2, St_2, St_{02}, T_2 \rangle \in Mod_\Sigma$.

Then $\gamma : St_1 \to St_2$ is a *bounded morphism* if and only if:

(1) for all $st_1 \in St_{01}$, $\gamma(st_1) \in St_{02}$,
(2) for all $st_2 \in St_{02}$, there exists $st_1 \in St_{01}$ such that $\gamma(st_1) = st_2$,
(3) for all $tr \in \Delta_{\mathfrak{K}_1}$ and $j \in \mathcal{J}$, $\mathfrak{K}_1, tr \models p_j(t_1, \ldots, t_{arity(p_j)})$ if and only if $\mathfrak{K}_2, \gamma(tr) \models p_j(t_1, \ldots, t_{arity(p_j)})$,[1]
(4) for all $st_1, st_1' \in St_1$, if $st_1 \; T_1 \; st_1'$, then $\gamma(st_1) \; T_2 \; \gamma(st_1')$ (the "forth" condition),
(5) for all $st_1 \in St_1$ and $st_2' \in St_2$, if $\gamma(st_1) \; T_2 \; st_2'$, then there exists $st_1' \in St_1$ such that $\gamma(st_1') = st_2'$ and $st_1 \; T_1 \; st_1'$ (the "back" condition).

If there is a surjective bounded morphism from $\mathfrak{K}_1 = \langle \mathcal{A}_1, St_1, St_{01}, T_1 \rangle$ to $\mathfrak{K}_2 = \langle \mathcal{A}_2, St_2, St_{02}, T_2 \rangle$, then we say that $\mathfrak{K}_2$ is a *bounded morphic image* of $\mathfrak{K}_1$, and is denoted $\mathfrak{K}_1 \twoheadrightarrow \mathfrak{K}_2$.

THEOREM 5.1. *Let $\Sigma \in Sign_{\mathsf{FOLTL}}$, and $\mathfrak{K}_1, \mathfrak{K}_2 \in Mod_\Sigma$, $tr \in \Delta_{\mathfrak{K}_1}$ and $\gamma : St_1 \to St_2$ a bounded morphism. Then for all $\phi \in Form_{\mathsf{FOLTL}}(\Sigma)$*

$$\mathfrak{K}_1, tr \models_{\mathsf{FOLTL}} \phi \quad \textit{iff} \quad \mathfrak{K}_2, \gamma(tr) \models_{\mathsf{FOLTL}} \phi \; .$$

PROOF. The proof is simple extension of the proof of [**BdV01**, §2, Proposition 2.14]. ∎

2. Interpretability of FOLTL in ω-CCFAU

Defining the translation for a first-order temporal language with function symbols $\{f_i\}_{i \in \mathcal{I}}$, predicate symbols $\{p_j\}_{j \in \mathcal{J}}$ and variables $\mathcal{V}$, requires extending the language of full proper closure fork algebras with urelements with new constants **St**, **T**, $\mathbf{St_0}$, **tr**, and families of constants $\{\mathbf{F}_i\}_{i \in \mathcal{I}}$, $\{\mathbf{P}_j\}_{j \in \mathcal{J}}$ and $\{\mathbf{V}_k\}_{k \in \mathbb{N}}$.

There are two usual ways to represent sets as binary relations: using partial identities (i.e., relations contained in the identity relation), or using right-ideal relations. Right-ideal relations relate each element in their domain to every element in the universe. Thus, the range provides no information. A right-ideal relation can be used to model the set provided by its domain.

In the following paragraphs we will present axioms characterizing the meaning of the added constants. The partial identity **St** will model the set St. Similarly, relation $\mathbf{St_0}$ is a partial identity modeling the set St_0. Relation **T** models the accessibility relation T. Relation **tr** models the set of traces. The constants $\{\mathbf{F}_i\}_{i \in \mathcal{I}}$ model the meaning of the function symbols. Similarly, relations $\{\mathbf{P}_j\}_{j \in \mathcal{J}}$ will model the meaning of predicate symbols.

[1] $\gamma : \Delta_{\mathfrak{K}_1} \to \Delta_{\mathfrak{K}_2}$ is defined as the homomorphic extension of $\gamma : St_1 \to St_2$ to traces.

$$\mathbf{St} \leq 1', \tag{5}$$
$$\breve{\pi};\mathbf{St};\pi = 1'_{\mathsf{U}}, \tag{6}$$
$$\mathbf{St} = 1'_{\mathsf{U}} \otimes \mathbf{St}, \tag{7}$$
$$\mathbf{St_0} \leq \mathbf{St}, \tag{8}$$
$$Dom(\mathbf{T}) = \mathbf{St}, \tag{9}$$

Equation (5) establishes that **St** is a partial identity (a set), Equation (6) and (7) establishes that states are infinite sequences of urelements. Equation (8) establishes that $\mathbf{S_0}$ is a subset of the set of states. Equation (9) establishes that **T** is a total (and therefore complete) relation on the set of states.

For each function symbol f, with arity n, we add the equations:

$$\breve{\mathbf{F}};\mathbf{F} \leq 1'_{\mathsf{U}}, \tag{10}$$
$$\underbrace{(1'_{\mathsf{U}} \otimes \cdots \otimes 1'_{\mathsf{U}})}_{n \text{ times}};\mathbf{F} = \mathbf{F}. \tag{11}$$

Equation (10) establishes that **F** is a functional relation, and (11) establishes that **F** expects an n-tuple as input and produces an urelement as output.

For each predicate symbol p, with arity n, we add the equation:

$$\underbrace{(1'_{\mathsf{U}} \otimes \cdots \otimes 1'_{\mathsf{U}})}_{n \text{ times}};\mathbf{P};1 = \mathbf{P}. \tag{12}$$

Equation (12) establishes that **P** a right-ideal relation, therefore representing a set. **P** represents the set of n-tuples that satisfy predicate p.

Since the semantics of temporal formulas is defined in terms of traces of states, we will model these notions in a fork algebra. Given a fork algebra $\mathfrak{A}$, we model states and traces in $\mathfrak{A}$ with elements from $U_{\mathfrak{A}}$ as the ones described by Figure 5.1.

First-order states are valuations from the denumerable set of variable symbols $\mathcal{V}$ to a set S. As we already mentioned, we require $\mathcal{V}$ to be totally ordered, thus allowing to interpret these valuations as functions from $\mathbb{N}$ to S. Now, as infinite right degenerated trees can be interpreted as infinite sequences, it is enough to label the leaves with elements from S to get a correct interpretation of valuations by means of elements from $U_{\mathfrak{A}}$.

In the case of traces, being infinite sequences of states, we will simply model them as infinite right degenerated trees whose leaves are labeled with the interpretation presented before for states.

The next definition provides a relational characterization of traces. The relation **tr**, characterizing the traces in a closure fork algebra is defined by the following equations:

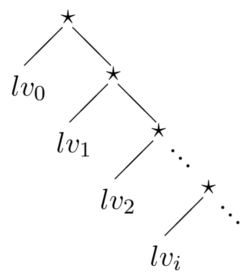


FIGURE 5.1. Infinite right degenerate trees pattern.

$$\mathbf{tr} \leq 1', \tag{13}$$

$$\breve{\pi};\mathbf{tr};\pi = \mathbf{St}, \tag{14}$$

$$\mathbf{tr} \leq \mathbf{St} \otimes \mathbf{tr}, \tag{15}$$

$$\mathbf{tr};\rho = \mathit{Ran}(\pi \nabla (\mathbf{T} \otimes \rho));\rho;\mathbf{tr}. \tag{16}$$

Equation (13) states that **tr** is a partial identity (a set). Equation (14) establishes that traces are built from states. Finally, Formulas (15) and (16) establish that traces are infinite, T-related, sequences of states.

The relations $\mathbf{V}_k$ allow us to build the $\mathcal{V}_k$-variants of a trace. They are defined as follows:

$$\mathbf{V}_k = \nu_X \left(\begin{array}{c} \mathit{Rep}_k \\ \otimes \\ X \end{array} ;\mathbf{tr} \right), \tag{17}$$

where ν is the largest fixed point operator, and Rep_i is defined as the term $\underbrace{1' \otimes \cdots \otimes 1'}_{i-1} \otimes {}_{\mathsf{U}}1_{\mathsf{U}} \otimes 1'$, (i.e. the binary relation that, when provided with a state $a_1 \star \cdots \star a_i \star \cdots$, returns all the states obtained by substituting the value of a_i).

Notice that the term $T_k(X) \widehat{=} (\mathit{Rep}_k \otimes X);\mathbf{tr}$, which defines $\mathbf{V}_k$, is monotonic (as a function of X). Moreover, $T_k(X)$ is co-continuous, i.e., it is meet-distributive. In order to prove this, let $(R_j)_{j \in J}$ be a chain. Then,

$$\begin{array}{ll}
& T_k(\prod_{j\in J} R_j) \\
= & (Rep_k \otimes \prod_{j\in J} R_j)\,;\mathbf{tr} \\
& \qquad [\text{by Definition } T_k] \\
= & ((\pi\,;Rep_k)\nabla(\rho;\prod_{j\in J} R_j))\,;\mathbf{tr} \\
& \qquad [\text{by Equation (2)}] \\
= & ((\pi\,;Rep_k\,;\breve{\pi})\cdot(\rho;\prod_{j\in J} R_j\,;\breve{\rho}))\,;\mathbf{tr} \\
& \qquad [\text{by Axiom (16)}] \\
= & ((\pi\,;Rep_k\,;\breve{\pi})\cdot(\prod_{j\in J}(\rho;R_j\,;\breve{\rho})))\,;\mathbf{tr} \\
& \qquad [\text{by } [\mathbf{CT51}, \text{Theorem 4.2}]] \\
= & (\prod_{j\in J}(\pi\,;Rep_k\,;\breve{\pi})\cdot(\rho;R_j\,;\breve{\rho}))\,;\mathbf{tr} \\
& \qquad [\text{by Boolean algebra}] \\
= & (\prod_{j\in J}(Rep_k \otimes R_j))\,;\mathbf{tr} \\
& \qquad [\text{by Axiom (16) and Equation (2)}] \\
= & \prod_{j\in J}((Rep_k \otimes R_j)\,;\mathbf{tr}) \\
& \qquad [\text{by } [\mathbf{CT51}, \text{Theorem 4.2}]] \\
= & \prod_{j\in J} T_k(R_j) \\
& \qquad [\text{by Definition } T_k]
\end{array}$$

By Knaster–Tarski's fixed point theorem [**Tar55c**], if the infimum of the chain $1, T_k(1), \ldots, T_k^j(1), \ldots$ exists, then $\nu_X\,(T_k(X)) = \Pi_{j<\omega} T_k^j(1)$. Since we are not assuming our models to be complete, so far we cannot guarantee the existence of $\Pi_{j<\omega} T_k^j(1)$. We could solve this by requiring models to be complete. From a proof–theoretical point of view, this would demand axioms and proof rules guaranteeing the existence of all infima, while we are in fact concerned about the existence of a single infimum.

A simple proof by induction on n (that essentially uses the monotonicity of $T_k(X)$) shows that $\mathbf{V}_k$ is a lower bound of the chain $1, T_k(1), \ldots, T_k^n(1), \ldots$. If we add the rule

$$\frac{y \leq T_k^j(1) \vdash y \leq T_k^{j+1}(1)}{\vdash y \leq \mathbf{V}_k}\ \mathit{VarRule}_k$$

then $\mathbf{V}_k$ is indeed the largest lower bound (the infimum) of the chain.

In order to see how the rule is used, let us prove that

$$Dom(\mathbf{V}_k) \geq \mathbf{tr}\ . \qquad (18)$$

Notice that this equation cannot follow directly from the rule because it does not have the right shape. Therefore, we must find another property implying (18) with the right shape. This is usually the hardest part. If $\mathbf{V}_k \geq \mathbf{tr}$ then $Dom(\mathbf{V}_k) \geq Dom(\mathbf{tr}) = \mathbf{tr}$. Therefore, we will concentrate on proving that $\mathbf{tr} \leq \mathbf{V}_k$. According to the rule, we must prove that

$$\mathbf{tr} \leq T_k^j(1) \vdash \mathbf{tr} \leq T_k^{j+1}(1)\ .$$

$$\begin{array}{rl}
& T_k^{j+1}(1) \\
= & (Rep_k \otimes T_k^j(1));\mathbf{tr} \\
& \quad \text{[by Definition } T_k\text{]} \\
\geq & (Rep_k \otimes \mathbf{tr});\mathbf{tr} \\
& \quad \text{[by Hypothesis]} \\
\geq & (\mathbf{St} \otimes \mathbf{tr});\mathbf{tr} \\
& \quad \text{[by Definition } Rep_k\text{]}
\end{array}$$

Notice that $\mathbf{St} \otimes \mathbf{tr} \leq 1'$. Since the composition of partial identities equals their intersection, we can continue as follows:

$$\begin{array}{rl}
& (\mathbf{St} \otimes \mathbf{tr});\mathbf{tr} \\
= & (\mathbf{St} \otimes \mathbf{tr})\cdot\mathbf{tr} \\
& \quad \text{[by previous discussion]} \\
\geq & \mathbf{tr}\cdot\mathbf{tr} \\
& \quad \text{[by Equation (15)]} \\
= & \mathbf{tr} \\
& \quad \text{[by Boolean algebra]}
\end{array}$$

Definition 5.7. We define the calculus ω-CCFAU' as the extension of ω-CCFAU obtained by adding equations Equations (5) – (16) as axioms, and the rules $\{VarRule_k\}_{k\in\mathbb{N}}$. Then the class ω-CFAU' is defined as the models of the equations derivable in ω-CCFAU'.

In Definitions 5.8 and 5.9 we present a translation of FOLTL terms and formulas to fork terms.

Definition 5.8. Let $\Sigma \in Sign_{\mathsf{FOLTL}}$, $\mathcal{V}$ be a denumerable set of variable symbols, and $\mathcal{R}$ be a set of relation variables, then we define the function $t_{\mathsf{FOLTL}} : Term_{\mathsf{FOLTL}}(\Sigma) \to RelDes(\mathcal{R})$, mapping FOLTL-terms to terms in full proper closure fork algebras with urelements, as follows:

$$\begin{array}{lcl}
t_{\mathsf{FOLTL}}(v) & = & \rho^{;(\mathcal{V}_v-1)};\pi \\
t_{\mathsf{FOLTL}}(f_i(t_1,\ldots,t_n)) & = & (t_{\mathsf{FOLTL}}(t_1)\nabla\cdots\nabla t_{\mathsf{FOLTL}}(t_n));\mathbf{F}_i
\end{array}$$

Definition 5.9. Let $\Sigma \in Sign_{\mathsf{FOLTL}}$, $\mathcal{V}$ be a denumerable set of variable symbols, and $\mathcal{R}$ be a set of relation variables, then we define the function $T_{\mathsf{FOLTL}} : Form_{\mathsf{FOLTL}}(\Sigma) \to RelDes(\mathcal{R})$, mapping formulas from FOLTL to terms in full proper closure fork algebras with urelements, as follows:

$$\begin{array}{l}
T_{\mathsf{FOLTL}}(p_i(t_1,\ldots,t_n)) = \pi;(t_{\mathsf{FOLTL}}(t_1)\nabla\cdots\nabla t_{\mathsf{FOLTL}}(t_n));\mathbf{P}_i\ , \\
T_{\mathsf{FOLTL}}(\neg\alpha) = \mathbf{tr};\overline{T_{\mathsf{FOLTL}}(\alpha)}\ , \\
T_{\mathsf{FOLTL}}(\alpha\vee\beta) = T_{\mathsf{FOLTL}}(\alpha)+T_{\mathsf{FOLTL}}(\beta)\ , \\
T_{\mathsf{FOLTL}}(\mathsf{X}\alpha) = \rho;T_{\mathsf{FOLTL}}(\alpha)\ , \\
T_{\mathsf{FOLTL}}(\alpha\mathsf{U}\beta) = (Dom(T_{\mathsf{FOLTL}}(\alpha));\rho)^*;T_{\mathsf{FOLTL}}(\beta)\ , \\
T_{\mathsf{FOLTL}}((\exists v)\alpha) = \mathbf{V}_{\mathcal{V}_v};T_{\mathsf{FOLTL}}(\alpha)\ .
\end{array}$$

It is clear that, in the translation of atomic formulas, the terms are evaluated in the current (first) state in the trace.

In the remaining part of this section we present all the necessary definitions in order to arrive to the main result on the interpretability of FOLTL.

Definition 5.10. Let S be a nonempty set, and T a binary relation on S. Let $\mathcal{T}(S,T)$ be the set of binary trees t satisfying:

- t is a binary tree with information in the leaves,
- t has infinite height,
- leaves are labeled with elements from S,
- t is right degenerate, i.e., t's shape follows the pattern exhibited in Figure 5.1, and
- given any two consecutive leaves of t holding information lv and lv', $\langle lv, lv' \rangle \in T$.

Definition 5.11. Given $\mathfrak{A} \in$ fPCFAU, we define:

- $dom(R) = \{ x \mid (\exists y)(\langle x,y \rangle \in R) \}$ for all $R \in \mathfrak{A}$,
- $\pi(x \star y) = x$ for all $x, y \in U_{\mathfrak{A}}$, and $\rho(x \star y) = y$ for all $x, y \in U_{\mathfrak{A}}$.

No confusion should arise between the relation constants π and ρ and the functions π and ρ from Definition 5.11; while the former are relational constants, the latter are functions and always appear being applied to arguments using functional notation.

In order to interpret FOLTL, it will be necessary to build full proper closure fork algebras from FOLTL-models. The domain on which relations are built must include the values for variables, states, and traces of states. Let S be the domain for variables. Thus, given an injective function $\star$, we define

$$S^{\star n} = \{ a_1 \star \cdots \star a_n \mid a_i \in S (1 \leq i \leq n) \} \ , \tag{19}$$

$$S^{\star} = \{ a_1 \star \cdots \star a_n \star \cdots \mid a_i \in S (i \in \mathbb{N}) \} \ . \tag{20}$$

Rather than using functions to represent states, we will use elements from $S^\star$, which come for free in any fork algebra.

Definition 5.12. Let S be a nonempty set, and T a binary relation on $S^\star$. Then, $\mathcal{T}(S,T)^\star$ is the smallest set of binary trees built as follows:

- $S \cup S^\star \cup \mathcal{T}(S^\star, T) \subseteq \mathcal{T}(S,T)^\star$, and
- if $t_1, t_2 \in \mathcal{T}(S,T)^\star$, then $t_1 \star t_2 \in \mathcal{T}(S,T)^\star$.

Now, by considering our representation of states (see Formula 20), and Definition 5.10, $S^\star$ will be used as states, or valuations, and $\mathcal{T}(S^\star, T)$ will be used to represent traces. Then the set $\mathcal{T}(S,T)^\star$ is the smallest set, closed by $\star$, also containing states and traces. Now, $\mathcal{T}(S,T)^\star$, will be used as the base set of a full proper closure fork algebra with urelements. This closure fork algebra will be used in further definitions and lemmas as the target to which FOLTL-models will be translated.

Definition 5.13. Let S be a nonempty set, and T a binary relation on $S^\star$. A *full proper closure fork algebra with urelements on S, T* is a full proper closure fork algebra with base set $\mathcal{T}(S,T)^\star$.

Definition 5.14. Let $\star$ be an injective function and S a non-empty set. Then, given a function $st : \mathcal{V} \to S$, we denote by $st^\star$ (the $\star$-representation of st) the object $st(\mathcal{V}_1) \star \cdots \star st(\mathcal{V}_n) \star \cdots$. Similarly, given an object $s = a_1 \star \cdots \star a_n \star \cdots$, by $s^{\langle,\rangle}$ (the $\langle,\rangle$-representation of s) we denote the function $\{ \langle \mathcal{V}_i, a_i \rangle \mid i \in \mathbb{N} \}$.

LEMMA 5.1. *Let S be a non-empty set, and $st \in [\mathcal{V} \to S]$. Then,*

$$(\forall i \in \mathbb{N})(\pi(\rho^i(st^\star)) = st(\mathcal{V}_i)) \ .$$

■

The following two definitions allow us to transform states to elements in a fork algebra, and viceversa. This will be useful in order to build algebras from linear models, as well as linear models from algebras.

DEFINITION 5.15. Let S be a non-empty set, T a binary relation on $[\mathcal{V} \to S]$ and tr a sequence of T-connected elements of $[\mathcal{V} \to S]$. We define $t_{tr} \in \mathcal{T}(S,T)^\star$ as the infinite tree satisfying:

$$(\forall i \in \mathbb{N})(\pi(\rho^i(t_{tr})) = (tr_i)^\star) \ .$$

LEMMA 5.2. *Given a nonempty set S, a binary relation T on $[\mathcal{V} \to S]$, and a sequence of T-connected elements of $[\mathcal{V} \to S]$, namely, tr,*

$$(\forall i \in \mathbb{N})(t_{tr^i} = \rho^i(t_{tr})) \ .$$

■

DEFINITION 5.16. Let S be a nonempty set. Let T be a binary relation on $S^\star$. Let $t \in \mathcal{T}(S,T)^\star$. We define tr_t as the sequence of states satisfying:

$$(\forall i \in \mathbb{N}) \left((tr_t)_i = \left(\pi(\rho^i(t))\right)^{\langle,\rangle} \right) \ .$$

LEMMA 5.3. *Given a nonempty set S, a binary relation T on $S^\star$, and $t \in \mathcal{T}(S,T)^\star$,*

$$(\forall i \in \mathbb{N})(tr_{\rho^i(t)} = (tr_t)^i) \ .$$

■

DEFINITION 5.17.
Let $\langle \mathbf{St}, \mathbf{St_0}, \mathbf{T}, \mathbf{tr}, \{\mathbf{F}_i\}_{i \in \mathcal{I}}, \{\mathbf{P}_j\}_{j \in \mathcal{J}}, \{\mathbf{V}_k\}_{k \in \mathbb{N}} \rangle \in \mathsf{Sign}_{\mathsf{fPCFAU}}$, a *full proper closure fork algebra with urelements on S, T extended with constants* is a full proper closure fork algebra with urelements on S, T in which:

- $\mathbf{St} = \{ \langle s, s \rangle \mid s \in S^\star \}$,
- $\mathbf{St_0} \subseteq \mathbf{St}$,
- $\mathbf{T} = T$,
- $\mathbf{tr} = \{ \langle t, t \rangle \mid t \in \mathcal{T}(S^\star, T) \}$,
- $\mathbf{F}_i$ is a functional relation, and there exists $k \in \mathbb{N}$ such that $\mathbf{F}_i \subseteq S^{\star k} \times S$ for all $i \in \mathcal{I}$,
- there exists $k \in \mathbb{N}$ such that $dom(\mathbf{P}_j) \subseteq S^{\star k}$ and $\mathbf{P}_j$ is right-ideal for all $j \in \mathcal{J}$, and
- $\mathbf{V}_k$ is the relation that, given a $\star$-representation of a trace, builds the $\star$-representation corresponding to the v-variants, provided that v is the $k^{th.}$ variable in the totally ordered finite set of variable symbols, for all $k \in \mathbb{N}$.

In order to fully define a full proper closure fork algebra on S, T extended with constants, it suffices to provide the meaning for $\mathbf{St_0}$, $\{\mathbf{F}_i\}_{i \in \mathcal{I}}$ and $\{\mathbf{P}_j\}_{j \in \mathcal{J}}$. The remaining constants have their values determined from these.

The class of full proper closure fork algebra with urelements on S, T extended with constants will be denoted as fPCFAU'

The following lemmas, whose proofs appear in [**FL06**, Lemma 3.16–3.18], are required in order to prove Theorem 5.2. The first one proves the completeness of ω-CCFAU' for the class of full proper closure fork algebra on S, T extended with constants.

LEMMA 5.4.
Let $\Sigma = \langle \mathbf{St}, \mathbf{St_0}, \mathbf{T}, \{\mathbf{F}_i\}_{i\in\mathcal{I}}, \{\mathbf{P}_j\}_{j\in\mathcal{J}}, \mathbf{tr}, \{\mathbf{V}_k\}_{k\in\mathbb{N}}\rangle \in \mathsf{Sign}_{\mathsf{fPCFAU}}$*, and let* $\alpha \in Sen_\Sigma$*. Then,*

$$\models_{\mathsf{fPCFAU}'} \alpha \quad\iff\quad \vdash_{\omega\text{-}\mathsf{CCFAU}'} \alpha \ .$$

■

The following two lemmas synthesize the relation between Kripke structures and full closure fork algebras with urelements.

LEMMA 5.5. *Let* $\alpha \in Form_{\mathsf{FOLTL}}$ *and* $\mathfrak{K} = \langle \mathcal{A}, St, St_0, T\rangle$ *a Kripke structure, then there exist a nonempty set* S*, a binary relation* T' *on* $S^\star$ *and a full proper closure fork algebra* $\mathfrak{A}$ *on* S*,* T'*, extended with constants such that for all* $tr \in \Delta_{\mathfrak{K}}$*,*

$$\mathfrak{K}, tr \models_{\mathsf{FOLTL}} \alpha \iff t_{tr} \in dom(T_{\mathsf{FOLTL}}(\alpha)) \ .$$

■

LEMMA 5.6. *Let* $\alpha \in Form_{\mathsf{FOLTL}}$*.Given* $\mathfrak{A}$*, a full proper closure fork algebra on* S*,* T *extended with constants, there exists a Kripke structure* $\mathfrak{K}$ *such that for all* $t \in dom(\mathbf{tr})$*,*

$$t \in dom(T_{\mathsf{FOLTL}}(\alpha)) \iff \mathfrak{K}, tr_t \models_{\mathsf{FOLTL}} \alpha \ .$$

■

The next theorem presents the interpretability result for the logic FOLTL. It shows that it is possible to replace semantic reasoning in FOLTL by equational reasoning in ω-CCFAU'. The proof can be found in [**FL06**, Theorem 3.19].

THEOREM 5.2. *Let* $\alpha \in Form_{\mathsf{FOLTL}}(\Sigma)$*. Then,*

$$\models_{\mathsf{FOLTL}} \alpha \iff \\ \vdash_{\omega\text{-}\mathsf{CCFAU}'} Dom(\pi;\mathbf{St_0});\mathbf{tr};T_{\mathsf{FOLTL}}(\alpha) = Dom(\pi;\mathbf{St_0});\mathbf{tr};1 \ .$$

■

3. Interpretability as a representation map between institutions

In this section we will show how this interpretability result can be rephrased as a representation map between the institution of FOLTL and that of fPCFAU'. In order to avoid the introduction of ambiguities, sets, categories, functors and families of relations will be identified by using the logic as a subscript.

As we mentioned at the beginning of this chapter, this section is structured following the order of requirements stated in Definition 3.29. Lemma 5.7 provides the definition of a functor mapping FOLTL-signatures to fPCFAU-signatures, then from Lemma 5.8 to Lemma 5.11 we present a natural transformation as a formalization of the translation between logical systems. In Equation (21) we extend the functor mapping FOLTL-signatures to fPCFAU-signatures to a functor mapping FOLTL-signatures to fPCFAU-theories. From Lemma 2.4 to Lemma 5.18 we present a natural transformation formalizing translation of full proper closure fork algebras with urelements to Kripke models (which are the standard models for first-order linear temporal logic). And finally in Lemma 5.21 and Theorem 5.3 we show that semantic consequence at the logical level can be replaced by semantic consequence in the algebraic setting, thus providing a complete calculus, ω-CCFAU, for the logic.

LTL has been widely studied from an institutional point of view. The reader interested in the details of the definitions and lemmas is pointed to [**Fia96**]. Its first-order version can be obtained by resorting to the constructions presented by Goguen and Burstall in [**GB84**] for classical first-order logic and those presented by Fiadeiro in [**Fia96**]. In order to make the following pages easier to read, we will use the same notation presented for $\mathscr{I}_{\mathsf{fPCFAU}}$ (defined in Chapter 4, §1).

The section is structured following the order of requirements stated in Definition 3.29.

From now on we will assume a fixed but arbitrary FOLTL-signature with shape $\Sigma = \langle \{f_i\}_{i\in\mathcal{I}}, \{p_j\}_{j\in\mathcal{J}}\rangle$, and will resort to superindexing with $'$ when more than one signature is needed. In the same way, if Σ is a FOLTL-signature, we will assume a fixed but arbitrary Kripke structure for Σ with shape $\mathfrak{K} = \langle \mathcal{A}, St, St_0, T\rangle$, where $\mathcal{A} = \langle \{f_i^{\mathcal{A}}\}_{i\in\mathcal{I}}, \{p_j^{\mathcal{A}}\}_{j\in\mathcal{J}}\rangle$. When more than one Kripke structure is needed, we will resort to subindexing using natural numbers, and superindexing with $'$. Morphisms between FOLTL-signatures will be pairs of the form $\langle \sigma^{func}, \sigma^{pred}\rangle$, where σ^{func} is an arity preserving total function mapping function symbols, σ^{pred} is an arity preserving total function mapping predicate symbols.

In §2 we presented the semantics preserving translation by assuming a local view, this means we only cared about a single FOLTL-signature. The translation we presented in Definition 5.9 highly depends on a FOLTL-signature to correctly interpret the extra-logical symbols as their corresponding algebraic objects. This means that we do not have a single translation, we have a family of translations of the form $\{T^{\Sigma}_{\mathsf{FOLTL}}\}_{\Sigma\in|\mathsf{Sign}_{\mathsf{FOLTL}}|}$.

LEMMA 5.7. *Let* $\Sigma, \Sigma' \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\sigma^{\mathsf{FOLTL}} : \Sigma \to \Sigma'$. *Then, we define* $\gamma^{Sign} : \mathsf{Sign}_{\mathsf{FOLTL}} \to \mathsf{Sign}_{\mathsf{fPCFAU}'}$ *as follows:*

- $\gamma^{Sign}(\langle \{f_i\}_{i\in\mathcal{I}}, \{p_j\}_{j\in\mathcal{J}}\rangle) =$
  $\langle \{\mathbf{St}, \mathbf{St}_0, \mathbf{T}, \mathbf{tr}\} \cup \{\mathbf{F}_i\}_{i\in\mathcal{I}} \cup \{\mathbf{P}_j\}_{j\in\mathcal{J}} \cup \{\mathbf{V}_k\}_{k\in\mathbb{N}}\rangle$,
- $\gamma^{Sign}(\sigma^{\mathsf{FOLTL}}) = \sigma^{\mathsf{fPCFAU}'}$ *if and only if:*
  - $\sigma^{\mathsf{fPCFAU}'}(\mathbf{St}) = \mathbf{St}'$,
  - $\sigma^{\mathsf{fPCFAU}'}(\mathbf{St}_0) = \mathbf{St}'_0$,
  - $\sigma^{\mathsf{fPCFAU}'}(\mathbf{T}) = \mathbf{T}'$,
  - $\sigma^{\mathsf{fPCFAU}'}(\mathbf{tr}) = \mathbf{tr}'$,
  - *for all* $i \in \mathcal{I}$,
    $\sigma^{\mathsf{fPCFAU}'}(\mathbf{F}_i) = \mathbf{F}'_{\sigma^{\mathsf{fPCFAU}'}(i)}$ *iff* $\sigma^{\mathsf{FOLTL}}(f_i) = f'_{\sigma^{\mathsf{FOLTL}\,func}(i)}$,

– *for all* $j \in \mathcal{J}$,

$$\sigma^{\mathsf{fPCFAU}'}(\mathbf{P}_j) = \mathbf{P}'_{\sigma^{\mathsf{fPCFAU}'}(j)} \textit{ iff } \sigma^{\mathsf{FOLTL}}(p_j) = p'_{\sigma^{\mathsf{FOLTL}^{pred}}(j)},$$

– *for all* $k \in \mathbb{N}$,

$$\sigma^{\mathsf{fPCFAU}'}(\mathbf{V}_k) = \mathbf{V}'_k.$$

Then, γ^{Sign} *is a functor.*

Proof. The proof that γ^{Sign} is a functor follows trivially from definition of γ^{Sign}. Given $\delta : \Sigma \to \Sigma'$ a morphism in $Sign_{\mathsf{FOLTL}}$, $\gamma^{Sign}(\delta)$ is a total function between the symbols of $\gamma^{Sign}(\Sigma)$ and the symbols of $\gamma^{Sign}(\Sigma')$. Now, by definition of $\gamma^{Sign}(\delta)$ it is easy to check that $\gamma^{Sign}(id_{\Sigma}^{\mathsf{FOLTL}}) = id_{\gamma^{Sign}(\Sigma)}^{\mathsf{fPCFAU}'}$ and that $\gamma^{Sign}(\sigma^{\mathsf{FOLTL}} \circ \sigma'^{\mathsf{FOLTL}}) = \gamma^{Sign}(\sigma^{\mathsf{FOLTL}}) \circ \gamma^{Sign}(\sigma'^{\mathsf{FOLTL}})$. ■

Lemma 5.8. *Let* $\Sigma \in \mathsf{Sign}_{\mathsf{FOLTL}}$ *and* $\alpha \in \mathbf{Sen}_{\mathsf{FOLTL}}(\Sigma)$. *Then, we define* $\gamma_{\Sigma}^{Sen} : \mathbf{Sen}_{\mathsf{FOLTL}}(\Sigma) \to \gamma^{Sign} \circ \mathbf{Sen}_{\mathsf{fPCFAU}'}(\Sigma)$ *as* $\gamma_{\Sigma}^{Sen} = T_{\mathsf{FOLTL}}$.

Then, γ_{Σ}^{Sen} *is a total function.*

Proof. The proof of this lemma follows by observing that, once we fix a signature, T_{FOLTL}, as presented in Definition 5.9, is a total function. ■

Consider $f : \Sigma \to \Sigma'$ a morphism in $Sign_{\mathsf{FOLTL}}$. Then, the definition of $\mathbf{Sen}_{\mathsf{FOLTL}}(f)$ requires f to be extended homorphically to FOLTL-formulas, which requires the homomorphic extension of f to FOLTL-terms. From now on, the former will be denoted by f_{form} and the latter by f_{term}.

Lemma 5.9. *Let* $\Sigma, \Sigma' \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\sigma : \Sigma \to \Sigma'$. *Let* $t \in Term_{\Sigma}$. *Then,* $t_{\mathsf{FOLTL}}(\sigma_{term}(t)) = (\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t))$.

Proof. The proof follows by induction on the structure of t. Let $t = v$, $v \in \mathcal{V}$, then:

$$\begin{array}{rl} & t_{\mathsf{FOLTL}}(\sigma_{term}(v)) \\ = & t_{\mathsf{FOLTL}}(v) \\ & \quad \text{[by Definition of } \sigma_{term}\text{]} \\ = & \rho^{;(\mathcal{V}_v - 1)};\pi \\ & \quad \text{[by Definition 5.8]} \\ = & (\gamma^{Sign}(\sigma))_{term}(\rho^{;(\mathcal{V}_v - 1)};\pi) \\ & \quad \text{[by Definition 4.1]} \\ = & (\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(v)) \\ & \quad \text{[by Definition 5.8]} \end{array}$$

Now, let $t = f_i(t_1, \ldots, t_{arity(f_i)})$, $i \in \mathcal{I}$, then:

$$\begin{array}{rl} & t_{\mathsf{FOLTL}}(\sigma_{term}(f_i(t_1, \ldots, t_{arity(f_i)}))) \\ = & t_{\mathsf{FOLTL}}(f'_{\sigma^{func}(i)}(\sigma_{term}(t_1), \ldots, \sigma_{term}(t_{arity(f_i)}))) \\ & \quad \text{[by Definition of } \sigma_{term}\text{]} \\ = & (t_{\mathsf{FOLTL}}(\sigma_{term}(t_1)) \nabla \ldots \nabla t_{\mathsf{FOLTL}}(\sigma_{term}(t_{arity(f_i)})));\mathbf{F}'_{\gamma^{Sign}(\sigma)(i)} \\ & \quad \text{[by Definition 5.9]} \\ = & ((\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_1)) \nabla \ldots \nabla (\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_{arity(f_i)})));\mathbf{F}'_{\gamma^{Sign}(\sigma)(i)} \\ & \quad \text{[by Inductive Hypothesis]} \\ = & ((\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_1)) \nabla \ldots \nabla (\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_{arity(f_i)})));(\gamma^{Sign}(\sigma))_{term}(\mathbf{F}_i) \\ & \quad \text{[by Lemma 5.7]} \\ = & (\gamma^{Sign}(\sigma))_{term}((t_{\mathsf{FOLTL}}(t_1) \nabla \ldots \nabla t_{\mathsf{FOLTL}}(t_{arity(f_i)}));\mathbf{F}_i) \\ & \quad \text{[by Definition 4.1]} \\ = & (\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(f_i(t_1, \ldots, t_{arity(f_i)}))) \ . \\ & \quad \text{[by Definition 5.9]} \end{array}$$

■

LEMMA 5.10. *Let* $\Sigma, \Sigma' \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\sigma : \Sigma \to \Sigma'$. *Let* $\alpha \in |\mathbf{Sen}(\Sigma)|$. *Then* $T_{\mathsf{FOLTL}}(\sigma_{form}(\alpha)) = (\gamma^{Sign}(\sigma))_{term}(T_{\mathsf{FOLTL}}(\alpha))$.

PROOF. The proof follows by induction on the structure of α.
Let $\alpha = p_j(t_1, \ldots, t_{arity(p_j)})$, $j \in \mathcal{J}$:

$$\begin{array}{rl}
& T_{\mathsf{FOLTL}}(\sigma_{form}(p_j(t_1,\ldots,t_{arity(p_j)}))) \\
= & T_{\mathsf{FOLTL}}(p'_{\sigma^{pred}(j)}(\sigma_{term}(t_1),\ldots,\sigma_{term}(t_{arity(p_j)}))) \\
& \quad \text{[by Definition of } \sigma_{form}] \\
= & \pi;(t_{\mathsf{FOLTL}}(\sigma_{term}(t_1))\nabla\cdots\nabla t_{\mathsf{FOLTL}}(\sigma_{term}(t_{arity(p_j)})));\mathbf{P'}_{\gamma^{Sign}(\sigma)(j)} \\
& \quad \text{[by Definition 5.9]} \\
= & \pi;((\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_1))\nabla\cdots\nabla(\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_{arity(p_j)})));\\
& \quad \mathbf{P'}_{\gamma^{Sign}(\sigma)(j)} \\
& \quad \text{[by Lemma 5.9]} \\
= & \pi;((\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_1))\nabla\cdots\nabla(\gamma^{Sign}(\sigma))_{term}(t_{\mathsf{FOLTL}}(t_{arity(p_j)})));\\
& \quad (\gamma^{Sign}(\sigma))_{term}(\mathbf{P}_j) \\
& \quad \text{[by Lemma 5.7]} \\
= & (\gamma^{Sign}(\sigma))_{term}(\pi;(t_{\mathsf{FOLTL}}(t_1)\nabla\cdots\nabla t_{\mathsf{FOLTL}}(t_{arity(p_j)}));\mathbf{P}_j \\
& \quad \text{[by Definition 4.1]} \\
= & (\gamma^{Sign}(\sigma))_{term}(T_{\mathsf{FOLTL}}(p_j(t_1,\ldots,t_{arity(p_j)}))) \\
& \quad \text{[by Definition 5.9]}
\end{array}$$

The proof for the inductive cases follows trivially by performing an analogous reasoning but using the inductive hypothesis. ■

LEMMA 5.11. $\{\gamma^{Sen}_{\Sigma}\}_{\Sigma\in|\mathsf{Sign}_{\mathsf{FOLTL}}|}$ *is a natural family of functions.*

PROOF. The naturality condition on $\{\gamma^{Sen}_{\Sigma}\}_{\Sigma\in|\mathsf{Sign}_{\mathsf{FOLTL}}|}$, i.e, the commutativity of the diagram

$$\begin{tikzcd}
\mathbf{Sen}_{\mathsf{FOLTL}}(\Sigma) \arrow[r, "\gamma^{Sen}_{\Sigma}"] \arrow[d, "\mathbf{Sen}_{\mathsf{FOLTL}}(f)"] & \gamma^{Sign} \circ \mathbf{Sen}_{\mathsf{fPCFAU'}}(\Sigma) \arrow[d, "\gamma^{Sign} \circ \mathbf{Sen}_{\mathsf{fPCFAU'}}(f)"] \\
\mathbf{Sen}_{\mathsf{FOLTL}}(\Sigma') \arrow[r, "\gamma^{Sen}_{\Sigma'}"] & \gamma^{Sign} \circ \mathbf{Sen}_{\mathsf{fPCFAU'}}(\Sigma')
\end{tikzcd}$$

follows by observing that $\gamma^{Sen}_{\Sigma}\circ(\gamma^{Sign}\circ\mathbf{Sen}_{\mathsf{fPCFAU'}}(f)) = \mathbf{Sen}_{\mathsf{FOLTL}}(f)\circ\gamma^{Sen}_{\Sigma'}$, which is a direct consequence of Lemma 5.10. ■

Having proved that γ^{Sen} is a natural transformation, allows the extension of the functor $\gamma^{Sign} : \mathsf{Sign}_{\mathsf{FOLTL}} \to \mathsf{Sign}_{\mathsf{fPCFAU'}}$ to another functor $\gamma^{Th_0} : \mathsf{Sign}_{\mathsf{FOLTL}} \to \mathsf{Th}_{\mathsf{fPCFAU'}_0}$, such that for all $\Sigma \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$:

(21) $$\gamma^{Th_0}(\Sigma) = \langle \gamma^{Sign}(\Sigma), \Gamma \rangle$$
$$\text{, where } \Gamma = \{\text{ Eqs. (1) – (24) of Chapter 2 }\} \cup \{\text{ Eqs. (5) – (16) of Chapter 5 }\}$$

Notice that the Equations (5) – (16) of Chapter 5 only depend on Σ, thus their appearance can be put in terms of the application of a functor $\mathbf{Ax} : \mathsf{Sign}_{\mathsf{FOLTL}} \to \mathsf{Set}$ which, provided a signature $\Sigma \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$, constructs a set containing Equations (5) – (16) of Chapter 5 by resorting to the symbols in $\gamma^{Sign}(\Sigma)$; and provided σ a morphism in $\mathsf{Sign}_{\mathsf{FOLTL}}$, $\mathbf{Ax}(\sigma) = \gamma^{Sen}_{\Sigma}$.

Formula 21 and Definition 5.7 implies that if $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$, then a model $\mathcal{M} \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$ is a full proper closure fork algebra with urelements on S, T extended with constants, for some S and T. From now on, models will be sub-indexed by natural numbers and, unless we make an explicit mention, their parameters S and T will be distinguished by resorting to the same sub-index (i.e. we will assume that $\mathcal{M}_i$ is a full proper closure fork algebra with urelements on S_i, T_i extended with constants).

Definition 5.18. Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ and $\mathcal{V}$ be a denumearable set of variable symbols. Then, we define

$$\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}} : (\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}}) \to \mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma^{\mathsf{FOLTL}})$$

as follows:

- Let $\mathcal{M} = \langle \mathcal{P}, \{\mathbf{St}^{\mathcal{M}}, \mathbf{St}_0^{\mathcal{M}}, \mathbf{T}^{\mathcal{M}}, \mathbf{tr}^{\mathcal{M}}\} \cup \{\mathbf{F}_i^{\mathcal{M}}\}_{i\in\mathcal{I}} \cup \{\mathbf{P}_j^{\mathcal{M}}\}_{j\in\mathcal{J}} \cup \{\mathbf{V}_k^{\mathcal{M}}\}_{k\in\mathbb{N}}\rangle$, then

  $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}) = \langle \mathcal{A}, St, St_0, T'\rangle$, where:
  - $\mathcal{A} = \langle A, \{f_i^{\mathcal{A}}\}_{i\in\mathcal{I}}, \{p_j^{\mathcal{A}}\}_{j\in\mathcal{J}}\rangle$ such that:
    * $A = S$,
    * $f_i^{\mathcal{A}}(a_1, \dots, a_{arity(f_i)}) = a$ iff $\langle a_1 \star \dots \star a_{arity(f_i)}, a\rangle \in \mathbf{F}_i^{\mathcal{M}}$, for all $i \in \mathcal{I}$,
    * $p_j^{\mathcal{A}}(a_1, \dots, a_{arity(p_j)})$ iff $a_1 \star \dots \star a_{arity(p_j)} \in dom(\mathbf{P}_j^{\mathcal{M}})$, for all $j \in \mathcal{J}$.
  - $St = \{\, t^{\langle,\rangle} \mid t \in dom(\mathbf{St}^{\mathcal{M}}) \,\}$,
  - $St_0 = \{\, t^{\langle,\rangle} \mid t \in dom(\mathbf{St}_0^{\mathcal{M}}) \,\}$,
  - $T' = \left\{ \left\langle t^{\langle,\rangle}, t'^{\langle,\rangle} \right\rangle \,\middle|\, \langle t, t'\rangle \in \mathbf{T}^{\mathcal{M}} \right\}$.
- Let $\mathcal{M}_1, \mathcal{M}_2 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$ and $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ be a morphism in $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$, then

  $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\delta) = \widehat{\delta^{\mathsf{FOLTL}}} : St_1 \to St_2$, satisfying that for all $st \in St_1$, $v \in \mathcal{V}$, $\widehat{\delta^{\mathsf{FOLTL}}}(st)(v) = \delta^{\mathsf{FOLTL}}(st(v))$, where $\delta^{\mathsf{FOLTL}} : A_1 \to A_2$ is defined as follows:

  $$\delta^{\mathsf{FOLTL}}(a) = b \text{ iff } \delta(\{\, \langle a, a\rangle \,\}) = \{\, \langle b, b\rangle \,\} .$$

Lemma 5.12.
Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\mathcal{M} \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$.
Then, $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}) \in |\mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma^{\mathsf{FOLTL}})|$.

Proof. Let $\Sigma^{\mathsf{FOLTL}} = \langle \{f_i\}_{I\in\mathcal{I}}, \{p_j\}_{j\in\mathcal{J}}\rangle$ be a FOLTL-signature. By Formula 21, the signature of $\gamma^{Th_0}(\Sigma^{\mathsf{FOLTL}})$ is $\langle \{\mathbf{St}, \mathbf{St}_0, \mathbf{T}, \mathbf{tr}\} \cup \{\mathbf{F}_i\}_{i\in\mathcal{I}} \cup \{\mathbf{P}_j\}_{j\in\mathcal{J}} \cup \{\mathbf{V}_k\}_{k\in\mathbb{N}}\rangle$.

As $\mathcal{M} \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$, it satisfies the axioms of Definition 5.7, thus, for every Σ^{FOLTL} function symbol f_i, the constant $\mathbf{F}_i^{\mathcal{M}}$ is a functional binary relation whose pairs are formed by the $\star$-representation of a tuple with $arity(f_i)$ elements of S in the first position of the pair and an element of S in the second one. Then, as $A = S$, by Definition 5.18, $f_i^{\mathcal{A}}$ is indeed a function on the set A with arity $arity(f_i)$. The same argument applies to Σ^{FOLTL} predicate symbols. Thus proving that $\mathcal{A} = \langle A, \{f_i^{\mathcal{A}}\}_{i\in\mathcal{I}}, \{p_j^{\mathcal{A}}\}_{j\in\mathcal{J}}\rangle$ is a Σ^{FOLTL}-structure.

Finally, from Definition 5.18, it is easy to observe that St is the set $[\mathcal{V} \to A]$, $St_0 \subseteq St$ and $T' \subseteq St \times St$ is a total binary relation.

Consequently $\langle \mathcal{A}, St, St_0, T'\rangle \in |\mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma^{\mathsf{FOLTL}})|$. ■

To prove that $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\delta)$ is a morphism in $\mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma^{\mathsf{FOLTL}})$, we will need the following lemmas.

Lemma 5.13.
Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\mathcal{M}_1, \mathcal{M}_2 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$, *such that* $\mathcal{M}_2$ *is non-trivial. Finally, let* $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ *be a morphism in* $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$.

Then, for all $st \in St_1$,

$$\delta(\{\, \langle st^\star, st^\star\rangle \,\}) = \left\{ \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star \right\rangle \right\} .$$

Proof. Assume that $\delta(\{\, \langle st^\star, st^\star\rangle \,\}) \neq \left\{ \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star \right\rangle \right\}$ then, as $st^\star$ and $\widehat{\delta^{\mathsf{FOLTL}}}(st)^\star$ are right degenerated trees, there exists $i \in \mathbb{N}$ such that:

$$Dom(\rho^i;\pi;\delta(\{\, \langle st^\star, st^\star\rangle \,\})) \neq Dom(\rho^i;\pi;\left\{ \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star \right\rangle \right\}) .$$

Then, as δ is a homomorphism, we obtain that:

$$\delta(Dom(\rho^i;\pi;\{\, \langle st^\star, st^\star\rangle \,\})) \neq Dom(\rho^i;\pi;\left\{ \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star \right\rangle \right\}) .$$

Finally, by Lemma 5.1, we obtain that

$$\delta(\{\, \langle st_i, st_i\rangle \,\}) \neq \left\{ \left\langle \delta^{\mathsf{FOLTL}}(st_i), \delta^{\mathsf{FOLTL}}(st_i) \right\rangle \right\} ,$$

which, by Definition 5.18 is a contradiction. ■

Corollary 5.1.
Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\mathcal{M}_1, \mathcal{M}_2 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$, *such that* $\mathcal{M}_2$ *is non-trivial. Finally, let* $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ *be a morphism in* $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$.

Then, for all $st, st' \in St_1$,

$$\delta(\{\, \langle st^\star, st'^\star\rangle \,\}) = \left\{ \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(st')^\star \right\rangle \right\} .$$

■

Lemma 5.14.
Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\mathcal{M}_1, \mathcal{M}_2 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$, *such that* $\mathcal{M}_2$ *is non-trivial. Finally, let* $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ *be a morphism in* $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$.

Then, for all $tr \in \Delta_{\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_1)}$,

$$\delta(\{\, \langle t_{tr}, t_{tr}\rangle \,\}) = \left\{ \left\langle t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)}, t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)} \right\rangle \right\} .$$

PROOF. Assume $\delta(\{ \langle t_{tr}, t_{tr} \rangle \}) \neq \left\{ \left\langle t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)}, t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)} \right\rangle \right\}$ then, as t_{tr} and $t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)}$ are infinite right degenerated trees, there exists $i \in \mathbb{N}$ such that:

$$Dom(\rho^i ; \pi ; \delta(\{ \langle t_{tr}, t_{tr} \rangle \})) \neq Dom\left(\rho^i ; \pi ; \left\{ \left\langle t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)}, t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)} \right\rangle \right\}\right) .$$

Notice that, as δ is a homomorphism, we get that

$$\delta(Dom(\rho^i ; \pi ; \{ \langle t_{tr}, t_{tr} \rangle \})) \neq Dom\left(\rho^i ; \pi ; \left\{ \left\langle t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)}, t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)} \right\rangle \right\}\right) .$$

It is easy to see that the relation $Dom(\rho^i ; \pi ; \delta(\{ \langle t_{tr}, t_{tr} \rangle \}))$ is the projection of the $\star$-representation of the $i^{th.}$ state in tr and the same reasoning applies to the right-hand side of the equality. Thus, we obtain that:

$$\delta(\{ \langle {tr_i}^\star, {tr_i}^\star \rangle \}) \neq \left\{ \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(tr_i)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(tr_i)^\star \right\rangle \right\} ,$$

which, by Lemma 5.13, is a contradiction. ■

LEMMA 5.15.
Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ and $\mathcal{M}_1, \mathcal{M}_2 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$, such that $\mathcal{M}_2$ is non-trivial. Finally, let $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ be a morphism in $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$.

Then, for all $l \in Term_{\Sigma^{\mathsf{FOLTL}}}$, $\alpha \in Form_{\Sigma^{\mathsf{FOLTL}}}$:

(1) $\delta(t_{\mathsf{FOLTL}}^{\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_1)}(l)) = t_{\mathsf{FOLTL}}^{\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_2)}(l)$,
(2) $\delta(T_{\mathsf{FOLTL}}^{\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_1)}(\alpha)) = T_{\mathsf{FOLTL}}^{\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_2)}(\alpha)$.

PROOF. Both proofs follows trivially by induction on the structure of the terms (in the case of (1)) and formulas (in the case of (2)), and considering that δ is a homomorphism from $\mathcal{M}_1$ to $\mathcal{M}_2$. ■

Finally, we prove that $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\delta)$ is a morphism in $\mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma^{\mathsf{FOLTL}})$.

LEMMA 5.16.
Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ and $\mathcal{M}_1, \mathcal{M}_2 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$, such that $\mathcal{M}_2$ is non-trivial. Finally, let $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ be a morphism in $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$.

Then, $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\delta)$ is a morphism in $\mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma^{\mathsf{FOLTL}})$ (i.e. a bounded morphism between $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_1)$ and $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_2)$).

PROOF. Let $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_1) = \mathfrak{K}_1$ and $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_2) = \mathfrak{K}_2$, where $\mathfrak{K}_1 = \langle \mathcal{A}_1, St_1, St_{01}, T_1 \rangle$ and $\mathfrak{K}_2 = \langle \mathcal{A}_2, St_2, St_{02}, T_2 \rangle$.

Now we prove that $\widehat{\delta^{\mathsf{FOLTL}}}$ is a bounded morphism by first proving that $\widehat{\delta^{\mathsf{FOLTL}}}$ satisfies Condition 3 of Definition 5.6. Let $tr \in \Delta_{\mathfrak{K}_1}$, p a predicate symbol of Σ^{FOLTL} such that $n = arity(p)$ and $l_1, \ldots, l_n \in Terms_{\mathsf{FOLTL}}$.

$$
\begin{array}{ll}
 & \mathfrak{K}_1, tr \models p(l_1,\ldots,l_n) \\
\text{iff} & t_{tr} \in \mathit{dom}(T^{\mathfrak{K}_1}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n))) \\
 & \qquad \text{[by Lemma 5.5]} \\
\text{iff} & \langle t_{tr}, t_{tr}\rangle \in \mathit{Dom}(T^{\mathfrak{K}_1}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n))) \\
\text{iff} & \{\,\langle t_{tr}, t_{tr}\rangle\,\} \subseteq \mathit{Dom}(T^{\mathfrak{K}_1}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n))) \\
\text{iff} & \delta(\{\,\langle t_{tr}, t_{tr}\rangle\,\}) \subseteq \delta(\mathit{Dom}(T^{\mathfrak{K}_1}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n)))) \\
 & \qquad \text{[by Lemma 2.5]} \\
\text{iff} & \delta(\{\,\langle t_{tr}, t_{tr}\rangle\,\}) \subseteq \mathit{Dom}(\delta(T^{\mathfrak{K}_1}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n)))) \\
 & \qquad \text{[because } \delta \text{ is a homomorphism]} \\
\text{iff} & \delta(\{\,\langle t_{tr}, t_{tr}\rangle\,\}) \subseteq \mathit{Dom}(T^{\mathfrak{K}_2}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n))) \\
 & \qquad \text{[by Lemma 5.15]} \\
\text{iff} & \left\{ \left\langle t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)}, t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)} \right\rangle \right\} \subseteq \mathit{Dom}(T^{\mathfrak{K}_2}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n))) \\
 & \qquad \text{[by Lemma 5.14]} \\
\text{iff} & \left\langle t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)}, t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)} \right\rangle \in \mathit{Dom}(T^{\mathfrak{K}_2}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n))) \\
\text{iff} & t_{\widehat{\delta^{\mathsf{FOLTL}}}(tr)} \in \mathit{dom}(T^{\mathfrak{K}_2}_{\mathsf{FOLTL}}(p(l_1,\ldots,l_n))) \\
\text{iff} & \mathfrak{K}_2, \widehat{\delta^{\mathsf{FOLTL}}}(tr) \models p(l_1,\ldots,l_n) \\
 & \qquad \text{[by Lemma 5.5]}
\end{array}
$$

Next we prove that $\widehat{\delta^{\mathsf{FOLTL}}}$ satisfies the forth condition (i.e. Condition 4). Let $st, st \in St_1$.

$$
\begin{array}{ll}
 & st\ T_1\ st' \\
\text{iff} & \left\langle st^\star, st'^\star \right\rangle in\mathbf{T}_1 \\
 & \qquad \text{[by Definition 5.18]} \\
\text{iff} & \{\,\left\langle st^\star, st'^\star \right\rangle\,\} \subseteq \mathbf{T}_1 \\
\text{iff} & \delta(\{\,\left\langle st^\star, st'^\star \right\rangle\,\}) \subseteq \delta(\mathbf{T}_1) \\
 & \qquad \text{[by Lemma 2.5]} \\
\text{iff} & \delta(\{\,\left\langle st^\star, st'^\star \right\rangle\,\}) \subseteq \mathbf{T}_2 \\
 & \qquad \text{[because } \delta \text{ is a homomorphism]} \\
\text{iff} & \left\{ \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(st')^\star \right\rangle \right\} \subseteq \mathbf{T}_2 \\
 & \qquad \text{[by Corollary 5.1]} \\
\text{iff} & \left\langle \widehat{\delta^{\mathsf{FOLTL}}}(st)^\star, \widehat{\delta^{\mathsf{FOLTL}}}(st')^\star \right\rangle \in \mathbf{T}_2 \\
\text{iff} & \widehat{\delta^{\mathsf{FOLTL}}}(st)\ T_2\ \widehat{\delta^{\mathsf{FOLTL}}}(st') \\
 & \qquad \text{[by Definition 5.18]}
\end{array}
$$

Finally, we prove that $\widehat{\delta^{\mathsf{FOLTL}}}$ satisfies the back condition (i.e. Condition 5). Let $st_1 \in St_1$, $st_2' \in St_2$.

$$
\begin{array}{ll}
 & \delta^{\mathsf{FOLTL}}(st_1)\ T_2\ st_2' \\
\text{iff} & \langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle \in \mathbf{T_2} \\
 & \qquad \text{[by Definition 5.18]} \\
\text{iff} & \{\,\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle\,\} \subseteq \mathbf{T_2} \\
\text{iff} & (\exists s_1, s_1' \in S_1{}^\star)(\delta(\{\,\langle s_1, s_1'\rangle\,\}) = \{\,\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle\,\} \wedge \{\,\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle\,\} \subseteq \mathbf{T_2}) \\
 & \qquad \text{[by Lemma 2.4]} \\
\text{iff} & (\exists s_1, s_1' \in S_1{}^\star)(\delta(\{\,\langle s_1, s_1'\rangle\,\}) = \{\,\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle\,\} \wedge \delta(\{\,\langle s_1, s_1'\rangle\,\}) \subseteq \mathbf{T_2}) \\
\text{iff} & (\exists s_1, s_1' \in S_1{}^\star)(\delta(\{\,\langle s_1, s_1'\rangle\,\}) = \{\,\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle\,\} \wedge \{\,\langle s_1, s_1'\rangle\,\} \subseteq \mathbf{T_1}) \\
 & \qquad \text{[by Lemma 2.5]} \\
\text{iff} & (\exists \widetilde{st_1}, st_1' \in St_1)(\delta(\left\{ \left\langle \widetilde{st_1}^\star, {st_1'}^\star \right\rangle \right\}) = \{\,\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle\,\} \wedge \left\{ \left\langle \widetilde{st_1}^\star, {st_1'}^\star \right\rangle \right\} \subseteq \mathbf{T_1}) \\
 & \qquad \text{[by Definition 5.14, } \widetilde{st_1} = s_1{}^{\langle,\rangle} \text{ and } st_1' = s_1'{}^{\langle,\rangle}] \\
\text{iff} & (\exists \widetilde{st_1}, st_1' \in St_1)(\left\{ \left\langle \delta^{\mathsf{FOLTL}}(\widetilde{st_1})^\star, \delta^{\mathsf{FOLTL}}(st_1')^\star \right\rangle \right\} = \{\,\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle\,\} \wedge \\
 & \qquad \left\{ \left\langle \widetilde{st_1}^\star, {st_1'}^\star \right\rangle \right\} \subseteq \mathbf{T_1}) \\
 & \qquad \text{[by Corollary 5.1]}
\end{array}
$$

Now, $\left\{ \left\langle \delta^{\mathsf{FOLTL}}(\widetilde{st_1})^\star, \delta^{\mathsf{FOLTL}}(st_1')^\star \right\rangle \right\} = \left\{ \left\langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \right\rangle \right\}$ holds if and only if $\delta^{\mathsf{FOLTL}}(\widetilde{st_1})^\star = \delta^{\mathsf{FOLTL}}(st_1)^\star \wedge \delta^{\mathsf{FOLTL}}(st_1')^\star = {st_2'}^\star$ holds. Then,

$$\begin{array}{ll}
 & (\exists \widetilde{st_1}, st_1' \in St_1)(\left\{ \left\langle \delta^{\mathsf{FOLTL}}(\widetilde{st_1})^\star, \delta^{\mathsf{FOLTL}}(st_1')^\star \right\rangle \right\} = \{ \langle \delta^{\mathsf{FOLTL}}(st_1)^\star, {st_2'}^\star \rangle \} \wedge \\
 & \qquad \left\{ \left\langle \widetilde{st_1}^\star, {st_1'}^\star \right\rangle \right\} \subseteq \mathbf{T_1}) \\
\text{iff} & (\exists st_1' \in St_1)(\delta^{\mathsf{FOLTL}}(st_1')^\star = {st_2'}^\star \wedge \{ \langle {st_1}^\star, {st_1'}^\star \rangle \} \subseteq \mathbf{T_1}) \\
\text{iff} & (\exists st_1' \in St_1)(\delta^{\mathsf{FOLTL}}(st_1') = st_2' \wedge \{ \langle {st_1}^\star, {st_1'}^\star \rangle \} \subseteq \mathbf{T_1}) \\
 & \qquad \text{[by Definition 5.14]} \\
\text{iff} & (\exists st_1' \in St_1)(\delta^{\mathsf{FOLTL}}(st_1') = st_2' \wedge \langle {st_1}^\star, {st_1'}^\star \rangle \in \mathbf{T_1}) \\
 & \qquad \text{[because } \delta \text{ is a homomorphism]} \\
\text{iff} & (\exists st_1' \in St_1)(\delta^{\mathsf{FOLTL}}(st_1') = st_2' \wedge st_1 \; T_1 \; st_1') \\
 & \qquad \text{[by Definition 5.18]}
\end{array}$$

The proof of Conditions 1 and 2 are analogous to the previous ones. ■

COROLLARY 5.2.
Let $\Sigma^{\mathsf{FOLTL}} \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\mathcal{M}_1, \mathcal{M}_2 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})|$, *such that* $\mathcal{M}_2$ *is non-trivial. Finally, let* $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ *be a morphism in* $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$.

Then, $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_1)$ *and* $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\mathcal{M}_2)$ *are modally equivalent.*

PROOF. The proof is obtained by observing that $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}(\delta)$ is a bounded morphism and applying Theorem 5.1. ■

On the other hand, if we consider Lemma 2.3, it is easy to prove a stronger result, which is that the Kripke models obtained by applying $\gamma^{Mod}_{\Sigma^{\mathsf{FOLTL}}}$ to two related objects in $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma^{\mathsf{FOLTL}})$ are isomorphic[2] instead of just bounded morphic.

LEMMA 5.17. *Let* $\Sigma \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$, *then* γ^{Mod}_{Σ} *is a functor.*

PROOF. Let $\mathcal{M} \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma)|$. Then, we prove that $\gamma^{Mod}_{\Sigma}(id_{\mathcal{M}}) = id_{\gamma^{Mod}_{\Sigma}(\mathcal{M})}$.

Let $\widehat{f} = \gamma^{Mod}_{\Sigma}(id_{\mathcal{M}})$, then $\widehat{f}(st)(v) = f(st(v))$ for all $st \in St$, $v \in \mathcal{V}$. As $id_{\mathcal{M}}(\{ \langle st(v), st(v) \rangle \}) = \{ \langle st(v), st(v) \rangle \}$ for all $st \in St$, $v \in \mathcal{V}$, because $id_{\mathcal{M}}(\{ \langle a, a \rangle \}) = \{ \langle a, a \rangle \}$ for all $a \in S$, then $\widehat{f}(st)(v) = st(v)$ for all $st \in St$, $v \in \mathcal{V}$ and consequently $\widehat{f}(st) = st$ for all $st \in St$. Then $\widehat{f} = id_{\gamma^{Mod}_{\Sigma}(\mathcal{M})}$.

Let $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3 \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma)|$ and $\delta : \mathcal{M}_1 \to \mathcal{M}_2$ and $\delta' : \mathcal{M}_2 \to \mathcal{M}_3$ be morphisms in $(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma)$. Now we prove that $\gamma^{Mod}_{\Sigma}(\delta \circ \delta') = \gamma^{Mod}_{\Sigma}(\delta) \circ \gamma^{Mod}_{\Sigma}(\delta')$.

Let $\widehat{g} = \gamma^{Mod}_{\Sigma}(\delta \circ \delta')$, then $\widehat{g}(st)(v) = g(st(v))$ for all $st \in St$, $v \in \mathcal{V}$. Let $st(v) = a$, then if $b \in S_3$, we get that

$$\begin{array}{ll}
 & g(a) = b \\
\text{iff} & \delta \circ \delta'(\{ \langle a, a \rangle \}) = \{ \langle b, b \rangle \} \\
 & \qquad \text{[by Definition 5.18]} \\
\text{iff} & \delta'(\delta(\{ \langle a, a \rangle \})) = \{ \langle b, b \rangle \} \\
\text{iff} & (\exists c \in S_2)(\delta(\{ \langle a, a \rangle \}) = \{ \langle c, c \rangle \} \wedge \delta'(\{ \langle c, c \rangle \}) = \{ \langle b, b \rangle \})
\end{array}$$

[2]An isomorphism is a bounded morphism in which Conditions 1, 2 and 3 hold, in Condition 4 the implication is replaced by an equivalence and Condition 5 does not necessarily hold. See [**BdV01**, Definition 2.8] for a formal definition of isomorphism for the case of modal logic.

Then, there exist $\widehat{f} : St_1 \to St_2$ and $\widehat{f'} : St_2 \to St_3$ such that $\gamma^{Mod}_\Sigma(\delta) = \widehat{f}$ and $\gamma^{Mod}_\Sigma(\delta') = \widehat{f'}$. Thus, we get

$$\begin{array}{ll} & (\exists c \in S_2)(f(a) = c \wedge f'(c) = b) \\ \text{iff} & f \circ f'(a) = b \end{array}$$

Then, by Definition 5.18, $\widehat{f \circ f'}(st)(v) = f \circ f'(st(v))$ and observing that $\widehat{f \circ f'} = \widehat{f} \circ \widehat{f'}$ we obtain $\widehat{f} \circ \widehat{f'}(st)(v) = f \circ f'(st(v))$. Thus $\gamma^{Mod}_\Sigma(\delta) \circ \gamma^{Mod}_\Sigma(\delta')(st)(v) = f \circ f'(st(v))$. Finally, we get $\gamma^{Mod}_\Sigma(\delta) \circ \gamma^{Mod}_\Sigma(\delta')(st)(v) = g(st(v))$ and, consequently $\gamma^{Mod}_\Sigma(\delta) \circ \gamma^{Mod}_\Sigma(\delta')(st)(v) = \gamma^{Mod}_\Sigma(\delta \circ \delta')(st)(v)$. ■

Lemma 5.18. $\{\gamma^{Mod}_\Sigma\}_{\Sigma \in |\mathsf{Sign}_{\mathsf{FOLTL}}|}$ *is a natural family of functors.*

Proof. The naturality condition on $\{\gamma^{Mod}_\Sigma\}_{\Sigma \in |\mathsf{Sign}_{\mathsf{FOLTL}}|}$, i.e. the commutativity of the diagram

$$\begin{tikzcd}
(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma') \arrow[r, "\gamma^{Mod}_{\Sigma'}"] \arrow[d, "(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(f)"] & \mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma') \arrow[d, "\mathbf{Mod}_{\mathsf{FOLTL}}(f)"] \\
(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma) \arrow[r, "\gamma^{Mod}_{\Sigma}"] & \mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma)
\end{tikzcd}$$

follows by observing that $((\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(f)) \circ \gamma^{Mod}_\Sigma = \gamma^{Mod}_{\Sigma'} \circ \mathbf{Mod}_{\mathsf{FOLTL}}(f)$ holds.

Let $f : \Sigma \to \Sigma'$ and $\mathcal{M}' = \langle \mathcal{P}, \{\mathbf{St}, \mathbf{St_0}, \mathbf{T}, \mathbf{tr}\} \cup \{\mathbf{F}_i\}_{i \in \mathcal{I}'} \cup \{\mathbf{P}_j\}_{j \in \mathcal{J}'} \cup \{\mathbf{V}_k\}_{k \in \mathbb{N}} \rangle \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(\Sigma')|$, then

$$\begin{array}{ll} & ((\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(f)) \circ \gamma^{Mod}_\Sigma(\mathcal{M}') \\ = & \gamma^{Mod}_\Sigma(((\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU'}}(f))(\mathcal{M}')) \\ = & \gamma^{Mod}_\Sigma(\mathbf{Mod}_{\mathsf{fPCFAU'}}((\gamma^{Th_0})^{\mathsf{op}}(f))(\mathcal{M}')) \\ = & \gamma^{Mod}_\Sigma(\mathcal{M}' \restriction_{(\gamma^{Th_0})^{\mathsf{op}}(f)}) \\ & \quad \text{[by Definition 4.6]} \\ = & \gamma^{Mod}_\Sigma(\langle \langle \mathcal{P}, \{\mathbf{St}, \mathbf{St_0}, \mathbf{T}, \mathbf{tr}\} \cup \{\mathbf{F}_i\}_{i \in \mathcal{I}} \cup \{\mathbf{P}_j\}_{j \in \mathcal{J}} \cup \{\mathbf{V}_k\}_{k \in \mathbb{N}} \rangle, \Gamma \rangle) \\ & \quad \text{[by Definition 4.5 and Formula 21]} \\ = & \langle \langle S, \{f_i\}_{i \in \mathcal{I}}, \{p_j\}_{j \in \mathcal{J}} \rangle, St, St_0, T' \rangle \end{array}$$

such that:

- $f_i(a_1, \ldots, a_{arity(f_i)}) = a$ iff $\left\langle a_1 \star \cdots \star a_{arity(f_i)}, a \right\rangle \in \mathbf{F}_i$, for all $i \in \mathcal{I}$,
- $p_j(a_1, \ldots, a_{arity(p_j)})$ iff $a_1 \star \cdots \star a_{arity(p_j)} \in dom(\mathbf{P}_j)$, for all $j \in \mathcal{J}$.
- $St = \left\{ s^{\langle,\rangle} \mid s \in dom(\mathbf{St}) \right\}$,
- $St_0 = \left\{ s^{\langle,\rangle} \mid s \in dom(\mathbf{St}_0) \right\}$,
- $T' = \left\{ \left\langle s^{\langle,\rangle}, s'^{\langle,\rangle} \right\rangle \middle| \langle s, s' \rangle \in \mathbf{T} \right\}$.

On the other hand,

$$\begin{array}{ll} & \gamma^{Mod}_{\Sigma'} \circ \mathbf{Mod}_{\mathsf{FOLTL}}(f)(\mathcal{M}') \\ = & \mathbf{Mod}_{\mathsf{FOLTL}}(f)(\gamma^{Mod}_{\Sigma'}(\mathcal{M}')) \\ = & \mathbf{Mod}_{\mathsf{FOLTL}}(f)(\langle \langle S, \{f_i\}_{i \in \mathcal{I}'}, \{p_j\}_{j \in \mathcal{J}'} \rangle, St, St_0, T' \rangle) \end{array}$$

such that:

- $f_i(a_1, \ldots, a_{arity(f_i)}) = a$ iff $\left\langle a_1 \star \cdots \star a_{arity(f_i)}, a \right\rangle \in \mathbf{F}_i$, for all $i \in \mathcal{I}'$,
- $p_j(a_1, \ldots, a_{arity(p_j)})$ iff $a_1 \star \cdots \star a_{arity(p_j)} \in dom(\mathbf{P}_j)$, for all $j \in \mathcal{J}'$.
- $St = \left\{ s^{\langle,\rangle} \mid s \in dom(\mathbf{St}) \right\}$,
- $St_0 = \left\{ s^{\langle,\rangle} \mid s \in dom(\mathbf{St}_0) \right\}$,
- $T' = \left\{ \left\langle s^{\langle,\rangle}, s'^{\langle,\rangle} \right\rangle \,\middle|\, \langle s, s' \rangle \in \mathbf{T} \right\}$.

$$= \langle\langle S, \{f_i\}_{i\in\mathcal{I}}, \{p_j\}_{j\in\mathcal{J}}\rangle, St, St_0, T'\rangle$$

Thus completing the proof. ■

Now we are able to prove that the satisfaction condition of Definition 3.28 is preserved by the representation map γ formed by the functor γ^{Th_0} defined in Lemma 5.7, the natural family of functions $\{\gamma^{Sen}_\Sigma\}_{\Sigma\in|\mathsf{Sign}_{\mathsf{FOLTL}}|}$ defined in Lemma 5.8, and the natural transformation $\{\gamma^{Mod}_\Sigma\}_{\Sigma\in|\mathsf{Sign}_{\mathsf{FOLTL}}|}$ defined in Definition 5.18.

LEMMA 5.19. *Let* $\Sigma \in |\mathsf{Sign}_{\mathsf{FOLTL}}|$ *and* $\mathcal{M}' \in |(\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU}'}(\Sigma)|$, *then*

$$\mathcal{M}' \models_{\gamma^{Th_0}(\Sigma)} \gamma^{Sen}_\Sigma(\alpha) \quad \textit{iff} \quad \gamma^{Mod}_\Sigma(\mathcal{M}') \models_\Sigma \alpha \ .$$

PROOF. Once Σ is fixed, the proof of this lemma follows by Lemma 5.6. ■

Once we proved the representability condition, by [**Tar96**, Proposition 5.2], we obtain that semantic consequence is preserved by institution representation. This result can be stated as follows.

LEMMA 5.20. *Let* $\Sigma \in \mathsf{Sign}_{\mathsf{FOLTL}}$, $\Gamma \cup \{\alpha\} \subseteq \mathbf{Sen}_{\mathsf{FOLTL}}(\Sigma)$.
Then, if $\Gamma \models^{\mathsf{FOLTL}}_\Sigma \alpha$, *then* $\gamma^{Sen}_\Sigma(\Gamma) \models^{\mathsf{fPCFAU}'}_{\gamma^{Th_0}(\Sigma)} \gamma^{Sen}_\Sigma(\alpha)$. ■

By Lemma 5.4 we know that ω-CCFAU' is a complete calculus for full proper closure fork algebras with urelements on S, T extended with constants (i.e. for the algebras we used to interpret Kripke models), then we would like to know if this is a complete calculus for FOLTL.

LEMMA 5.21. *Let* $\Sigma \in \mathsf{Sign}_{\mathsf{FOLTL}}$, $\Gamma \cup \{\alpha\} \subseteq \mathbf{Sen}_{\mathsf{FOLTL}}(\Sigma)$.
Then, for all $\mathfrak{K} \in \mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma)$ *there exists a* $(\gamma^{Th_0})^{\mathsf{op}}(\Sigma)$*-models* $\mathcal{M} \in (\gamma^{Th_0})^{\mathsf{op}} \circ \mathbf{Mod}_{\mathsf{fPCFAU}'}(\Sigma)$ *such that* $\gamma^{Mod}_\Sigma(\mathcal{M}) = \mathfrak{K}$

PROOF. The proof follows by Lemma 5.5. ■

The following theorem, which is a particular case of [**Tar96**, Corollary 5.4], proves that, as the translation of models is surjective, the proof calculus formalized in Chapter 4, §3, enriched with Axioms (5) – (16), and extended with rules $\{\,VarRule_k\}_{k\in\mathbb{N}}$, is a complete calculus for FOLTL.

THEOREM 5.3. *Let* $\Sigma \in \mathsf{Sign}_{\mathsf{FOLTL}}$, $\Gamma \cup \{\alpha\} \subseteq \mathbf{Sen}_{\mathsf{FOLTL}}(\Sigma)$.
Then, if for all $\mathfrak{K} \in \mathbf{Mod}_{\mathsf{FOLTL}}(\Sigma)$ *there exists a* $(\gamma^{Th_0})^{\mathsf{op}}(\Sigma)$*-models* $\mathcal{M} \in \mathbf{Mod}_{\mathsf{fPCFAU}'}((\gamma^{Th_0})^{\mathsf{op}}(\Sigma))$ *such that* $\gamma^{Mod}_\Sigma(\mathcal{M}) = \mathfrak{K}$*, then*

$$\Gamma \models^{\mathsf{FOLTL}}_\Sigma \alpha \quad \textit{iff} \quad \gamma^{Sen}_\Sigma(\Gamma) \models^{\mathsf{fPCFAU}'}_{\gamma^{Th_0}(\Sigma)} \gamma^{Sen}_\Sigma(\alpha) \ .$$

■

Until now, we have been able to relate two institutions by means of a representation map which, in a few words, maps signatures of the institution of the logical language to theories of the algebraic one. Notice that, as we are interested in using the logical languages to write specifications, what we really want is to translate theories of the institution of the logical language to theories of the algebraic one. This can be easily achieved by extending the functor $\gamma^{Th_0} : \mathsf{Sign}^I \to \mathsf{Th}^{I'}{}_0$ to a new functor $\widehat{\gamma^{Th_0}} : \mathsf{Th}^I{}_0 \to \mathsf{Th}^{I'}{}_0$ which have to be γ^{Sen}-sensible with respect to the entailment systems induced by the institutions I and I'. Now, if $\langle \Sigma, \Gamma \rangle \in |Sign^I|$, then $\widehat{\gamma^{Th_0}}$ is defined as follows:

$$\widehat{\gamma^{Th_0}}(\langle \Sigma, \Gamma \rangle) = \langle \mathbf{sign} \circ \gamma^{Th_0}(\Sigma), \mathbf{ax} \circ \gamma^{Th_0}(\Sigma) \cup \gamma^{Sen}_{\Sigma}(\Gamma) \rangle \ ,$$

where $\mathbf{sign} : \mathsf{Th}^I{}_0 \to \mathsf{Sign}^I$ is the functor that, provided a theory, projects its signature, and $\mathbf{ax} : \mathsf{Th}^I{}_0 \to \mathsf{Set}$ is the functor that, provided a theory, projects its axioms. Then, it is easy to prove that $\widehat{\gamma^{Th_0}}$ is γ^{Sen}-simple because it is the γ^{Sen}-extension of γ^{Th_0} to theories, thus being γ^{Sen}-sensible.

CHAPTER 6

Reasoning across logics in fork algebras

In Chapter 5 we showed, by resorting to an example, how an interpretability result can be reformulated as a representation map from the institution of the logic being interpreted into extensions of fork algebras. In the same way, it is possible to review the other existing representability results [**Fri02, FO98, FBM02, FL03, FGLR05**] but considering these categorical constructions instead of set theory. In this chapter we will show what is the advantage of this approach over the use of the original set theoretical.

This chapter is divided into three main section. In §1 we provide definitions and results presenting the machinery used to glue partial specifications. §2 address the problem of adding design decisions in order to bring into the description of the system the interactions of the partial specifications. Finally, in §3 we draw some conclusions related with the problem of expressing and adding this kind of additional information.

1. Relating partial specifications

As we mentioned at the beginning of this work, our claim is that fork algebras can be used as a framework in which it is possible to translate separate specifications written in different logics in a way that allows us to obtain a single algebraic specification of the whole system. To accomplish this task we need more than representation maps from the institutions of each logic to fork algebras; this result by itself only allows us to reuse reasoning tools developed for fork algebras in the task of verifying or validating logical specifications. This is because separate logical specifications give rise to separate algebraic specifications that offer no help in the solution of the problem of relating the concepts introduced in them.

LEMMA 6.1. *The category* $\mathsf{Sign}_{\mathsf{fPCFAU}}$ *is finitely co-complete.* ■

This proof of this lemma can be obtained by proving that the category of signatures has initial objects ($\langle\emptyset\rangle$) and pushouts for every pair of morphisms with common source object and then using Lemma 3.2.

Whether the use of pushouts is a good choice to prove the existence of co-limits for every finite diagram, it does not provide any hint of how to obtain them.

Now we show how co-limits in $\mathsf{Sign}_{\mathsf{fPCFAU}}$ are obtained in order to get a complete signature for partial ones. Then, we will extend the construction to the category of theories $\mathsf{Th}_{\mathsf{fPCFAU0}}$.

DEFINITION 6.1. Let $G = \langle V, E \rangle$ be a graph and $\delta : G \to \mathsf{Sign}_{\mathsf{fPCFAU}}$ be a finite diagram of $\mathsf{Sign}_{\mathsf{fPCFAU}}$. Let $\Sigma_v = \langle \{f_i\}_{i \in \mathcal{I}_v} \rangle \in |\mathsf{Sign}_{\mathsf{fPCFAU}}|$ for all $v \in V$. Then, we define the relations $\longrightarrow^\delta$, $\longleftrightarrow^\delta$, $\overset{n}{\longleftrightarrow}^\delta$, and $\overset{*}{\longleftrightarrow}^\delta$ as follows:

(1) $f_i \longrightarrow^\delta f_j$ if and only if there exist $v, w \in V$, $(v, w) \in E$, $\delta(v) = \Sigma_v$, $\delta(w) = \Sigma_w$, $i \in \mathcal{J}_v$, $j \in \mathcal{J}_w$ and $j = \delta((v, w))(i)$,
(2) $f_i \longleftrightarrow^\delta f_j$ if and only if $f_i \longrightarrow^\delta f_j$ or $f_j \longrightarrow^\delta f_i$,
(3) $f_i \overset{n}{\longleftrightarrow}^\delta f_j$ if and only if:
   - $n = 1$ and $f_i \longleftrightarrow^\delta f_j$, or
   - $n > 1$ and there exists f_k such that $f_i \longleftrightarrow^\delta f_k$ and $f_k \overset{n-1}{\longleftrightarrow}^\delta f_j$,

(4) $f_i \overset{*}{\longleftrightarrow}^\delta f_j$ if and only if $f_i \left(\bigcup_{n \in \mathbb{N}} \overset{n}{\longleftrightarrow}^\delta \right) f_j$.

Notice that if $G = \langle V, E \rangle$ is a graph and $\delta : G \to \mathsf{Sign}_{\mathsf{fPCFAU}}$ is a finite diagram of $\mathsf{Sign}_{\mathsf{fPCFAU}}$, then $\overset{*}{\longleftrightarrow}^\delta$ is reflexive, symmetric and transitive thus inducing a partition on the union of the sets of symbols of all the signatures involved in the diagram. The next definition will help us clarifying the notation we will use further.

DEFINITION 6.2. Let S a set, $R \subseteq S \times S$ be a reflexive, symmetric and transitive relation and $s \subseteq S$, then $s|R \subseteq S$ satisfies the following properties:

- for all $f \in s$ there exists a unique $f' \in s|R$ such that $R(f, f')$,
- for all $f' \in s|R$ there exists $f \in s$ such that $R(f, f')$.

If $f \in s$, then $[[f]]_R$ denotes the unique element $f' \in s|R$ such that $R(f, f')$.

LEMMA 6.2. *Let $G = \langle V, E \rangle$ a graph and $\delta : G \to \mathsf{Sign}_{\mathsf{fPCFAU}}$ a finite diagram in $\mathsf{Sign}_{\mathsf{fPCFAU}}$. Let $\Sigma_v = \langle \{f_i\}_{i \in \mathcal{I}_v} \rangle \in |\mathsf{Sign}_{\mathsf{fPCFAU}}|$ for all $v \in V$. Then, the co-cone $\langle \langle S \rangle, \{\gamma_v : \Sigma_v \to \langle S \rangle\}_{v \in V} \rangle$ where:*

- *$S = \left(\bigcup_{v \in V} \{f_i\}_{i \in \mathcal{I}_v} \right) | \overset{*}{\longleftrightarrow}^\delta$, and*
- *$\gamma_v(f_i) = [[f_i]]_{\overset{*}{\longleftrightarrow}^\delta}$ for all $i \in \mathcal{I}_v$,*

is a co-limit of δ.

PROOF. Let $v, v' \in V$, $(v, v') \in E$, and let $\Sigma_v = \langle \{f_i\}_{i \in \mathcal{I}} \rangle, \Sigma_{v'} = \langle \{f'_i\}_{i \in \mathcal{I}'} \rangle \in |\mathsf{Sign}_{\mathsf{fPCFAU}}|$ and $\sigma : \Sigma_v \to \Sigma_{v'}$ be a morphism in $\mathsf{Sign}_{\mathsf{fPCFAU}}$ such that $\delta(v) = \Sigma_v$, $\delta(v') = \Sigma_{v'}$ and $\delta((v, v')) = \sigma$. Let $i_0 \in \mathcal{I}$ and $i'_0 \in \mathcal{I}'$ such that $i'_0 = \sigma(i_0)$. Then $\gamma_v(f_{i_0}) = [[f_{i_0}]]_{\overset{*}{\longleftrightarrow}^\delta} = \gamma_{v'}(f'_{i'_0})$, and consequently $\langle \langle S \rangle, \{\gamma_v : \Sigma_v \to \langle S \rangle\}_{v \in V} \rangle$ is a commutative co-cone producing the following diagram

$$\begin{tikzcd}
 & \langle S \rangle & \\
 & \odot & \\
\langle \{f_i\}_{i\in\mathcal{I}} \rangle \arrow[uur, "\gamma_v"] \arrow[rr, "\sigma"] & & \langle \{f'_i\}_{i\in\mathcal{I}'} \rangle \arrow[uul, "\gamma_{v'}"']
\end{tikzcd}$$

Let $\langle\langle S'\rangle, \{\gamma'_v : \Sigma_v \to \langle S'\rangle\}_{v\in V}\rangle$ be another commutative co-cone for δ. Let $\Sigma_v = \langle\{f_i\}_{i\in\mathcal{I}}\rangle \in |\mathsf{Sign}_{\mathsf{fPCFAU}}|$ then, it is trivial to see that there exists a morphism $\tau : \langle S\rangle \to \langle S'\rangle$ satisfying $\tau(\gamma(f_i)) = \gamma'(f_i)$ for all $i \in \mathcal{I}$. Now it only rests to prove that τ is unique.

Assume there exists $\tau' : \langle S\rangle \to \langle S'\rangle$ such that $\gamma \circ \tau' = \gamma'$ and $\tau \neq \tau'$ then let f_i be a function symbol in Σ such that $\tau(\gamma(f_i)) \neq \tau'(\gamma(f_i))$. Then, $\tau'(\gamma(f_i)) \neq \tau(\gamma(f_i)) = \gamma'(f_i)$ and consequently $\gamma \circ \tau' \neq \gamma'$.

Then, $\langle\langle S\rangle, \{\gamma_v : \Sigma_v \to \langle S\rangle\}_{v\in V}\rangle$ is a co-limit for δ. ■

LEMMA 6.3. *The category* $\mathsf{Th}_{\mathsf{fPCFAU}0}$ *is finitely co-complete.*

PROOF. If we consider the functor $\mathbf{Sign} : \mathsf{Th}_{\mathsf{fPCFAU}0} \to \mathsf{Sign}_{\mathsf{fPCFAU}}$ which for any $\langle\Sigma, \Gamma\rangle \in |\mathsf{Th}_{\mathsf{fPCFAU}0}|$, $\mathbf{Sign}(\langle\Sigma, \Gamma\rangle) = \Sigma$, then the proof of this lemma follows by Lemma 6.1 and [**GB92**, Theorem 11]. ■

Now, the construction performed in Lemma 6.2 for the category of signature $\mathsf{Sign}_{\mathsf{fPCFAU}}$ can be extended to the category $\mathsf{Th}_{\mathsf{fPCFAU}0}$ in the same way is done in the proof of [**GB92**, Theorem 11].

DEFINITION 6.3. Let $G = \langle V, E\rangle$ a graph and $\delta : G \to \mathsf{Th}_{\mathsf{fPCFAU}0}$ a finite diagram in $\mathsf{Th}_{\mathsf{fPCFAU}0}$. Let $T_v = \langle\langle\{f_i\}_{i\in\mathcal{I}_v}\rangle, \Gamma_v\rangle \in |\mathsf{Th}_{\mathsf{fPCFAU}0}|$ for all $v \in V$. Then, $\langle\langle\Sigma, \Gamma\rangle, \{\gamma_v : T_v \to \langle\Sigma, \Gamma\rangle\}_{v\in V}\rangle$ where $\Sigma = \langle\left(\bigcup_{v\in V}\{f_i\}_{i\in\mathcal{I}_v}\right) \mid \overset{*}{\longleftrightarrow}^{\delta}\rangle$, $\Gamma = \bigcup_{v\in V}\gamma_{v\,eq}(\Gamma_v)$, and $\gamma_v(f_i) = [[f_i]]_{\overset{*}{\longleftrightarrow}\delta}$ for all $i \in \mathcal{I}_v$, is a co-limit of δ.

Definition 6.3 allows us to compute a fork algebra theory specifying the whole system. Even having the possibility of building a single specification of the system, we did not gain too much because there is no information about how specifications interact; for example, assume we have a system S whose main properties are temporal (formalized in a linear temporal logic theory $\langle\Sigma_S^{\mathsf{LTL}}, \Gamma_S^{\mathsf{LTL}}\rangle$) and dynamic (formalized in a propositional dynamic logic theory $\langle\Sigma_S^{\mathsf{PDL}}, \Gamma_S^{\mathsf{PDL}}\rangle$), provided that the intention is that some of the propositional symbols appearing in the signatures should be shared. Let $\rho_{\mathsf{LTL}} : \mathscr{I}_{\mathsf{LTL}} \to \mathscr{I}_{\mathsf{fPCFAU}}$ and $\rho_{\mathsf{PDL}} : \mathscr{I}_{\mathsf{PDL}} \to \mathscr{I}_{\mathsf{fPCFAU}}$ be institution representations[1]. We would like shared propositional variables appearing in Th_S^{LTL} to end up being the same as those appearing in Th_S^{PDL}. This interaction could be carried out by resorting to a morphism between $\rho_{\mathsf{LTL}}(Th_S^{\mathsf{LTL}})$ and $\rho_{\mathsf{PDL}}(Th_S^{\mathsf{PDL}})$, mapping the relational constants corresponding to propositional variables in Th_S^{LTL} to the relational constants corresponding to propositional variables in Th_S^{PDL} as desired. Unfortunately, most of the time,

[1]The definition of the representation map $\rho_{\mathsf{PDL}} : \mathscr{I}_{\mathsf{PDL}} \to \mathscr{I}_{\mathsf{fPCFAU}}$ can be given in the same way we did in Chapter 5 for FOLTL, but resorting to the set-theoretical result presented in [**FO98**] by Frias and Orlowska.

these morphisms do not exist because theories resulting from the translations may not share some symbols (for example, the relational constant **tr** in $\rho_{\mathsf{LTL}}(Th_S^{\mathsf{LTL}})$, see [**FL06**], has no dynamic counterpart and the family of relational constants $\{\mathbf{A}_i\}_{i\in\mathcal{I}}$ in $\rho_{\mathsf{PDL}}(Th_S^{\mathsf{PDL}})$, see [**FO98**], has no temporal counterpart), thus avoiding the possibility to get a total mapping from one of the signatures to the other.

To solve this mismatch in the synchronization of symbols, Fiadeiro introduced the notion of "channel" [**Fia96**]. Channels are theories that only contain symbols with no particular behavior (do not contain any axioms). Notice that these axiomless theories can be trivially related to theories that come from system description by means of morphisms in Th_{fPCFAU0}. This is because any theory morphism having a channel in its domain is axiom preserving. Continuing with the example of a system specified in LTL and PDL, consider $T_{\mathsf{LTL}} = \langle\langle\{\mathbf{St}, \mathbf{St}_0, \mathbf{T}, \mathbf{tr}\} \cup \{\mathbf{P}_i\}_{i\in\mathcal{I}}\rangle, \Gamma_{\mathsf{LTL}}\rangle$ and $T_{\mathsf{PDL}} = \langle\langle\{\mathbf{A}_k\}_{k\in\mathcal{K}} \cup \{\mathbf{Q}_j\}_{j\in\mathcal{J}}\rangle, \Gamma_{\mathsf{PDL}}\rangle$ as the partial specifications. Now assume that for all $l \in \mathcal{L}$, $\mathbf{P}_l$ and $\mathbf{Q}_l$ must agree because they reflect the same propositional variable. Then, it is possible to synchronize these symbols by resorting to the concept of channel; in this case the channel is a fork algebra theory $T_C = \langle\langle\{\mathbf{R}_l\}_{l\in\mathcal{L}}\rangle, \emptyset\rangle$, together with morphisms $\sigma_{\mathsf{LTL}} : T_C \to T_{\mathsf{LTL}}$ and $\sigma_{\mathsf{PDL}} : T_C \to T_{\mathsf{PDL}}$ such that:

- $\sigma_{\mathsf{LTL}}(\mathbf{R}_l) = \mathbf{P}_l$, for all $l \in \mathcal{L}$, and
- $\sigma_{\mathsf{PDL}}(\mathbf{R}_l) = \mathbf{Q}_l$, for all $l \in \mathcal{L}$.

Now, if we recall Lemma 6.2, the theory being the vertex of the co-limit of the diagram whose image contains the theories T_{LTL}, T_{PDL}, T_C, and the morphisms σ_{LTL} and σ_{PDL} will have the following shape:

$$\begin{aligned}&\langle\langle\{\mathbf{St}, \mathbf{St}_0, \mathbf{T}, \mathbf{tr}\} \cup \{\mathbf{P}_i\}_{i\in\mathcal{I}/\mathcal{L}} \cup \{\mathbf{R}_l\}_{l\in\mathcal{L}} \cup \{\mathbf{Q}_j\}_{j\in\mathcal{J}/\mathcal{L}}\rangle,\\ &\qquad \sigma_{T_{\mathsf{LTL}}{}_{eq}}(\Gamma_{\mathsf{LTL}}) \cup \sigma_{T_{\mathsf{PDL}}{}_{eq}}(\Gamma_{\mathsf{PDL}})\rangle\ ,\\ &\qquad\qquad \text{where } \sigma_{T_{\mathsf{LTL}}} \text{ and } \sigma_{T_{\mathsf{PDL}}} \text{ are the edges of the co-cone.}\end{aligned}$$

Notice that this construction forces $\mathbf{R}_l$ to be the unique symbol characterized by the translations of those LTL axioms mentioning $\mathbf{P}_l$ and those PDL axioms mentioning $\mathbf{Q}_l$, for all $l \in \mathcal{L}$.

Once channels have been depicted for every pair of fork algebra theories in the image of the diagram, then the computation of the vertex of its co-limit (including the channels), yields a single theory specifying the whole system in which the symbols representing the same logical objects have been glued together by the explicit introduction of a channel describing the synchronization.

2. Introducing design decisions

In §1 we described how partial specifications can be put together to get a specification of the whole system. At this point the specification we obtained reflects the behavior we originally specified in the partial descriptions of the system, plus some extra information allowing the synchronization of some symbols.

At the moment of specifying a system, it is highly recommended to split the specification into modules according to the different aspects being modeled. It has been proved in practice, since Parnas proposal of the concept of "separation of concerns" [**Par72**], that this methodology produces clearer descriptions of software systems with a vast amount of advantages. When dealing with homogeneous specifications the composition method guaranties that all the information in the design is preserved in the composition of the fragments. In our approach we not only divide the system into modules but also produce a separation between different aspects of the same module. At this point we face a difficult problem, i.e., how do we reflect the relationship between these aspects? Going back again to our example, we want to express that the LTL transition relation $\mathbf{T}$ is somehow related to the PDL atomic actions $\{\mathbf{A}_k\}_{k\in\mathcal{K}}$. This information does not appear at the top level specification. In fact, it cannot be expressed because the information provided by the temporal accessibility relation has no dynamic counterpart and viceversa. Having interpreted both pieces of specification into a common language allows us to bridge this gap permitting the addition of design decisions by resorting to an extra fork algebra theory expressing these constrains.

In the following sections we will address solutions for putting together logics that are commonly used in system design. In Tables 6.1, 6.2 and 6.3 we show which combinations we have considered as guiding examples. Locations marked with ◯ represent combinations of logics that are solved by combining the solution proposed by Fiadeiro in [**Fia96**] and, either the solution developed for some other location or some trivial considerations.

		Propositional		
		Temporal	Dynamic	Static
Propositional	Temporal	[**Fia96**]	§2.1	
	Dynamic	§2.1	◯	
	Static			◯

TABLE 6.1. Combinations of propositional logics.

		Propositional		
		Temporal	Dynamic	Static
First-order	Temporal			
	Dynamic		§2.3	
	Static			§2.2

TABLE 6.2. Combinations of propositional and first-order logics.

2.1. Synchronizing propositional temporal and propositional dynamic information. In a previous paragraph we showed how to solve the problem of gluing propositional symbols together by just adding a channel declaring which symbols should be interpreted as the same entity. In this section we will address two other problems; the first one is how to relate the temporal accessibility relation with the dynamic atomic actions, and the

		First-order		
		Temporal	Dynamic	Static
First-order	Temporal	○		§2.4
	Dynamic		○	
	Static	§2.4		○

TABLE 6.3. Combinations of first-order logics.

second one is what to do when the relation between propositional symbols is not the equality.

Consider the theories $\mathcal{T}_{\mathsf{LTL}} = \langle\langle\{\mathbf{St}, \mathbf{St}_0, \mathbf{T}, \mathbf{tr}\} \cup \{\mathbf{P}_i\}_{i\in\mathcal{I}}\rangle, \Gamma_{\mathsf{LTL}}\rangle$ and $\mathcal{T}_{\mathsf{PDL}} = \langle\langle\{\mathbf{A}_k\}_{k\in\mathcal{K}} \cup \{\mathbf{Q}_j\}_{j\in\mathcal{J}}\rangle, \Gamma_{\mathsf{PDL}}\rangle$ as the translations from LTL and PDL partial specifications to ω-closure fork algebras with urelements.

The temporal accessibility relation (its algebraic counterpart in this case) $\mathbf{T}$ can be thought of as the only one step transformation that can be carried out on the system state. This is why traces ($\mathbf{tr}$) are $\mathbf{T}$-connected infinite sequences of states $\mathbf{St}$, meaning that any possible evolution of the system can be followed by using $\mathbf{T}$ to go from the current system state to the next one.

On the other hand, the atomic actions $\{\mathbf{A}_k\}_{k\in\mathcal{K}}$ are a dynamic description of some of the system functions (operations that carry out a certain state transformation). Thus, assuming we have a dynamic description for all of the system functions, the following axiom can be added as a design decision:

(22) $$\mathbf{T} = \Sigma_{k\in\mathcal{K}}\mathbf{A}_k$$

Figure 6.1 shows the image of the diagram we should consider to express this design decision.

$$\begin{tikzcd}
 & & \langle\langle\{\mathbf{T}\} \cup \{\mathbf{A}_k\}_{k\in\mathcal{K}}\rangle, \{\mathbf{T} = \Sigma_{k\in\mathcal{K}}\mathbf{A}_k\}\rangle & & \\
 & \langle\langle\{\mathbf{T}\}\rangle, \emptyset\rangle \arrow[ur, "\sigma_3"] \arrow[dl, "\sigma_1"'] & & \langle\langle\{\mathbf{A}_k\}_{k\in\mathcal{K}}\rangle, \emptyset\rangle \arrow[ul, "\sigma_4"'] \arrow[dr, "\sigma_2"] & \\
\mathcal{T}_{\mathsf{LTL}} & & & & \mathcal{T}_{\mathsf{PDL}}
\end{tikzcd}$$

$$\begin{array}{rcl}
\sigma_1(\mathbf{T}) & = & \mathbf{T} \\
\sigma_2(\mathbf{A}_k) & = & \mathbf{A}_k \text{ ; for all } k \in \mathcal{K} \\
\sigma_3(\mathbf{T}) & = & \mathbf{T} \\
\sigma_4(\mathbf{A}_k) & = & \mathbf{A}_k \text{ ; for all } k \in \mathcal{K}
\end{array}$$

FIGURE 6.1. Relation between $\mathbf{T}$ and $\{\mathbf{A}_k\}_{k\in\mathcal{K}}$.

Depending on the information we have in our partial specifications, we can consider several variations on this diagram. In Figure 6.1 we showed how to relate $\mathbf{T}$ and $\{\mathbf{A}_k\}_{k\in\mathcal{K}}$ considering that the dynamic description was complete (this means that for every admissible functions we have a dynamic description) but we could probably have dynamic descriptions only for a proper subset of the system functions. In this case the relation between $\mathbf{T}$ and $\{\mathbf{A}_k\}_{k\in\mathcal{K}}$ is no longer $\mathbf{T} = \Sigma_{k\in\mathcal{K}}\mathbf{A}_k$ but a proper inclusion stating that the addition of the dynamic descriptions are a part of the temporal accessibility relation, then the axiom to get this relationship is[2]:

$$\Sigma_{k\in\mathcal{K}}\mathbf{A}_k < \mathbf{T}\ .$$

Notice that it is not possible to have a partial temporal description because $\mathbf{tr}$ is an abstract description of all the admissible executions of the system without taking care of how the single step transitions are carried out, then $\mathbf{T}$ is a complete description of these single step transformations of the system state.

Now consider, as we did in previous sections, $\mathcal{L} \subseteq \mathcal{I}$, $\mathcal{L} \subseteq \mathcal{J}$ and the channels $\langle\langle\{\mathbf{T}\}\cup\{\mathbf{P}_l\}_{l\in\mathcal{L}}\rangle, \emptyset\rangle$ and $\langle\langle\{\mathbf{A}_k\}_{k\in\mathcal{K}}\cup\{\mathbf{Q}_l\}_{l\in\mathcal{L}}\rangle, \emptyset\rangle$ relating not only $\mathbf{T}$ and $\{\mathbf{A}_k\}_{k\in\mathcal{K}}$ but also the propositional symbols $\{\mathbf{P}_l\}_{l\in\mathcal{L}}$ and $\{\mathbf{Q}_l\}_{l\in\mathcal{L}}$. Then we can complete the diagram of Figure 6.1 with the relations between propositional symbols obtaining the diagram in Figure 6.2.

$$\begin{tikzcd}
 & & \langle\langle\{\mathbf{T}\}\cup\{\mathbf{A}_k\}_{k\in\mathcal{K}}\cup\{\mathbf{R}_l\}_{l\in\mathcal{L}}\rangle, \{\mathbf{T} = \Sigma_{k\in\mathcal{K}}\mathbf{A}_k\}\rangle & & \\
 & \langle\langle\{\mathbf{T}\}\cup\{\mathbf{P}_l\}_{l\in\mathcal{L}}\rangle, \emptyset\rangle \arrow[ur, "\sigma_3"] \arrow[dl, "\sigma_1"'] & & \langle\langle\{\mathbf{A}_k\}_{k\in\mathcal{K}}\cup\{\mathbf{Q}_l\}_{l\in\mathcal{L}}\rangle, \emptyset\rangle \arrow[ul, "\sigma_4"'] \arrow[dr, "\sigma_2"] & \\
T_{\mathsf{LTL}} & & & & T_{\mathsf{PDL}}
\end{tikzcd}$$

$$\begin{aligned}
\sigma_1(\mathbf{T}) &= \mathbf{T}\\
\sigma_1(\mathbf{P}_l) &= \mathbf{P}_l\ ;\ \text{for all } l\in\mathcal{L}\\
\sigma_2(\mathbf{A}_k) &= \mathbf{A}_k\ ;\ \text{for all } k\in\mathcal{K}\\
\sigma_2(\mathbf{Q}_l) &= \mathbf{Q}_l\ ;\ \text{for all } l\in\mathcal{L}\\
\sigma_3(\mathbf{T}) &= \mathbf{T}\\
\sigma_3(\mathbf{P}_l) &= \mathbf{R}_l\ ;\ \text{for all } l\in\mathcal{L}\\
\sigma_4(\mathbf{A}_k) &= \mathbf{A}_k\ ;\ \text{for all } k\in\mathcal{K}\\
\sigma_4(\mathbf{Q}_l) &= \mathbf{R}_l\ ;\ \text{for all } l\in\mathcal{L}
\end{aligned}$$

FIGURE 6.2. Relation between $\mathbf{T}$ and $\{\mathbf{A}_k\}_{k\in\mathcal{K}}$, and $\{\mathbf{P}_l\}_{l\in\mathcal{L}}$ and $\{\mathbf{Q}_l\}_{l\in\mathcal{L}}$.

The diagram in Figure 6.2 is an appropriate solution for relating propositional symbols through equivalence. This means that we identified symbols which are the same logical entity but came from different partial specifications. The use of a common algebraic language allows us to introduce

[2]In the same way $x \leq y$ is essentially the equation $x{+}y = y$, $x < y$ can be rewritten as the following two axioms:

$$x{+}y = y\ , \tag{23}$$

$$1;(y\cdot\overline{x})\,;1 = 1\ . \tag{24}$$

Axiom (23) states that x is a subset of y and Axiom (24) implies that x and y are not equal.

more complex relations by the introduction of axioms in the theory appearing at the top of the diagram. The following definitions and propositions show that any Boolean combination of propositional symbols can be represented by means of algebraic operators. Operators $inl : P \to P \uplus Q$ and $inr : Q \to P \uplus Q$ are the injections of the elements of P and Q in the disjoint union of P and Q.

The following definition presents a language similar to a typical propositional language but built over two sets of variables. This definition gives the possibility of identifying the variables by recognizing where they were originated. In other words, we will use this propositional language to relate propositions coming from two different propositional partial specifications.

Definition 6.4. Let P and Q be two sets of propositional variables. Then, the set of Boolean constrains is the smallest set $BoolConst(P, Q)$ such that:

- $inl(p) \in BoolConst(P, Q)$, for all $p \in P$,
- $inr(q) \in BoolConst(P, Q)$, for all $q \in Q$,
- if $\alpha, \beta \in BoolConst(P, Q)$, then $\{\neg\alpha, \alpha \vee \beta\} \subseteq BoolConst(P, Q)$.

The rest of the propositional operators such as conjunction ($\wedge$), and implication ($\Longrightarrow$), are defined as always in terms of the negation ($\neg$) and disjunction ($\vee$) operators as $\alpha \wedge \beta \equiv \neg(\neg\alpha \vee \neg\beta)$ and $\alpha \Longrightarrow \beta \equiv \neg\alpha \vee \beta$.

Definition 6.5. Let P and Q be two sets of propositional symbols, and $\alpha \in BoolConst(P, Q)$ be a Boolean constrain. Then, we define the function $Tr_{BC} : BoolConst(P, Q) \to CCFAUForm(V)$ in the following way[3]:

$$\begin{array}{rcl}
Tr_{BC}(inl(p)) & \equiv & \mathbf{P} \\
 & & \quad ; \text{for } p \in P \text{ and } Tr_{\mathsf{PDL}\to\mathsf{fPCFAU}}(p) \equiv \mathbf{P}, \\
Tr_{BC}(inr(q)) & \equiv & \mathbf{Q} \\
 & & \quad ; \text{for } q \in Q \text{ and } Tr_{\mathsf{LTL}\to\mathsf{fPCFAU}}(q) \equiv \mathbf{Q}, \\
Tr_{BC}(\neg\alpha) & \equiv & \mathbf{St}\cdot\overline{Tr_{BC}(\alpha)}\ , \\
Tr_{BC}(\alpha \vee \beta) & \equiv & Tr_{BC}(\alpha)+Tr_{BC}(\beta)\ .
\end{array}$$

The use of **St** in the third clause is due to the need of filtering the domain of the complement of the translation of a formula to keep only those elements representing system states.

Notice that the translation of logical formulas yields relation designations, then it is not possible to add the translation of a formula (as is) as a design decision in an algebraic theory. Also notice that as formulas are translated to right ideals, and we want a certain formula $\alpha \in BoolConst(P, Q)$ to be valid on every possible state of the system, we can instead add the following formula as axiom:

$$\mathbf{St};Tr_{BC}(\alpha) = \mathbf{St};1\ .$$

[3]Notice that not only V must be chosen appropriately for this translation to be possible, but also the particular algebraic signature to which the translation is carried out. Also notice that the one corresponding to the theory that incorporates the design decisions (see Figure 6.2) can play this rôle because it contains all the symbols needed to apply Tr_{BC}.

To prove the semantic preservation of this translation of Boolean constrains we first need to characterize a class of models and a notion of validity between this class of models and Boolean constrains and then proceed with the proof by interpreting these models as full proper closure fork algebras with urelements, which are models of the co-limit theory of the diagram of Figure 6.2.

As the main topic of this thesis is not the presentation of a top level language to describe cross-logic constrains we will not go into further formalization in the framework of general logics but we believe this is one of the most important directions of further research this work leaves open. The reader is invited to either carry out this formalization, or simply use our proposal as a motivation to propose better solutions to this problem.

2.2. Synchronizing propositional and first-order static information. If our aim is to obtain some kind of interaction between a propositional and a first-order partial specification, then the notion of propositional and first-order system state must be related in an appropriate way. In this section we show how to obtain such a relationship.

Once we translated both logical theories and obtained two algebraic partial specifications (one containing the propositional information and the other containing the first-order information), we can introduce a *translation relation* representing the relationship between $\mathbf{St}^{\mathsf{Prop}}$ (the set of propositional system states) and $\mathbf{St}^{\mathsf{FOL}}$ (the set of first-order system states). Many different interactions can be established between $\mathbf{St}^{\mathsf{Prop}}$ and $\mathbf{St}^{\mathsf{FOL}}$, and all of them are obtained by the introduction of appropriate design decisions in the form of axioms (see for example Axiom (22)).

Assume that we know that both descriptions of system states are complete, this means that both $\mathbf{St}^{\mathsf{Prop}}$ and $\mathbf{St}^{\mathsf{FOL}}$ are "equal" modulo some translation from the propositional world to the first-order world. Let ${}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}$ be this translation relation, then we can add the following axioms:

$$
\begin{aligned}
&(25) & {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}} &\leq \mathbf{St}^{\mathsf{Prop}};1;\mathbf{St}^{\mathsf{FOL}} \ ,\\
&(26) & Dom\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}\right) &= \mathbf{St}^{\mathsf{Prop}} \ ,\\
&(27) & Ran\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}\right) &= \mathbf{St}^{\mathsf{FOL}} \ ,\\
&(28) & {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}} &= {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}};\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}\right)^{\smile};{}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}} \ .
\end{aligned}
$$

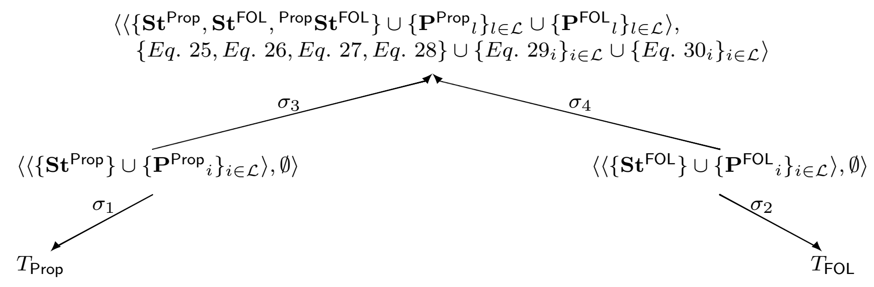
Equation (25) states that ${}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}$ is indeed a relation whose domain is a subset of the set of propositional states and whose range is a subset of the set of first-order states. Equation (26) forces ${}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}$ to be total, which means that any propositional state is related to a subset of the first-order states; and Equation (27) forces the converse relation, that any first-order state is the image of at least one propositional state. When dealing with an incomplete propositional description, we will surely have first-order states which are not the image of any propositional state, thus Equation (27) must be removed. The same happens when dealing with an incomplete first-order description but in this case we will have propositional states which are not mapped to any first-order state, thus Equation (26) must be removed. Equation (28) states that the translation is stable, which means that translating propositional states forth to first-order states and then back to propositional states, and then forth again, is exactly the same as translating the propositional states only once.

Now we can use ${}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}$ to establish further relationships between the behavior specified in the partial specifications. Let $\{\mathbf{P}^{\mathsf{Prop}}{}_i\}_{i\in\mathcal{I}^{\mathsf{Prop}}}$ be the set of propositions appearing in the propositional specifications and $\{\mathbf{P}^{\mathsf{FOL}}{}_i\}_{i\in\mathcal{I}^{\mathsf{FOL}}}$ be the set of predicates appearing in the first-order specification. Once again consider $\mathcal{L} \subseteq \mathcal{I}^{\mathsf{Prop}}$, $\mathcal{L} \subseteq \mathcal{I}^{\mathsf{FOL}}$ and the channels $\langle\langle \mathbf{St}^{\mathsf{Prop}} \cup \{\mathbf{P}^{\mathsf{Prop}}{}_i\}_{i\in\mathcal{L}}\rangle, \emptyset\rangle$ and $\langle\langle \mathbf{St}^{\mathsf{FOL}} \cup \{\mathbf{P}^{\mathsf{FOL}}{}_i\}_{i\in\mathcal{L}}\rangle, \emptyset\rangle$. In this case the symbols $\{\mathbf{P}^{\mathsf{Prop}}{}_i\}_{i\in\mathcal{L}}$ and $\{\mathbf{P}^{\mathsf{FOL}}{}_i\}_{i\in\mathcal{L}}$ will not be glued by using morphisms because we want this symbols to preserve different interpretations, those associated by the corresponding translation. They will be related by means of axioms characterizing the the way they interact. Then, if we want to identify the behavior of $\mathbf{P}^{\mathsf{Prop}}{}_i$ with the behavior of $\mathbf{P}^{\mathsf{FOL}}{}_i$ for all $i \in \mathcal{L}$, we need to add the axioms:

$$\begin{aligned}
&(29) \qquad & {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}};\mathbf{P}^{\mathsf{FOL}}{}_i &= \mathbf{P}^{\mathsf{Prop}}{}_i \text{ , for all } i \in \mathcal{L},\\
&(30) \qquad & \left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}\right)^{\smile};\mathbf{P}^{\mathsf{Prop}}{}_i &= \mathbf{P}^{\mathsf{FOL}}{}_i \text{ , for all } i \in \mathcal{L}.
\end{aligned}$$

Let $i \in \mathcal{L}$, then Axiom (29) indicates that the set of propositional state in the domain of $\mathbf{P}^{\mathsf{Prop}}{}_i$ is translated to the set of first-order state in the domain of $\mathbf{P}^{\mathsf{FOL}}{}_i$, and Axiom (30) indicates the converse relation.

The diagram involving all this information is obtained analogously to that of Figure 6.2, but considering the channels and axioms presented in this section, and appropriate morphisms mapping the symbols defined in the channels to the symbols appearing in the original theories and the theory containing the design decisions. Figure 6.3 shows the image of this diagram.



$$\begin{tikzcd}
 & \begin{array}{l}\langle\langle\{\mathbf{St}^{\mathsf{Prop}}, \mathbf{St}^{\mathsf{FOL}}, {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}\} \cup \{\mathbf{P}^{\mathsf{Prop}}{}_l\}_{l\in\mathcal{L}} \cup \{\mathbf{P}^{\mathsf{FOL}}{}_l\}_{l\in\mathcal{L}}\rangle,\\ \{Eq.\ 25, Eq.\ 26, Eq.\ 27, Eq.\ 28\} \cup \{Eq.\ 29_i\}_{i\in\mathcal{L}} \cup \{Eq.\ 30_i\}_{i\in\mathcal{L}}\rangle\end{array} & \\
\langle\langle\{\mathbf{St}^{\mathsf{Prop}}\} \cup \{\mathbf{P}^{\mathsf{Prop}}{}_i\}_{i\in\mathcal{L}}\rangle, \emptyset\rangle \arrow[ur, "\sigma_3"] \arrow[d, "\sigma_1"'] & & \langle\langle\{\mathbf{St}^{\mathsf{FOL}}\} \cup \{\mathbf{P}^{\mathsf{FOL}}{}_i\}_{i\in\mathcal{L}}\rangle, \emptyset\rangle \arrow[ul, "\sigma_4"'] \arrow[d, "\sigma_2"] \\
T_{\mathsf{Prop}} & & T_{\mathsf{FOL}}
\end{tikzcd}$$

$$\begin{aligned}
\sigma_1(\mathbf{St}^{\mathsf{Prop}}) &= \mathbf{St}^{\mathsf{Prop}}\\
\sigma_1(\mathbf{P}^{\mathsf{Prop}}{}_l) &= \mathbf{P}^{\mathsf{Prop}}{}_l \text{ ; for all } l \in \mathcal{L}\\
\sigma_2(\mathbf{St}^{\mathsf{FOL}}) &= \mathbf{St}^{\mathsf{FOL}}\\
\sigma_2(\mathbf{P}^{\mathsf{FOL}}{}_l) &= \mathbf{P}^{\mathsf{FOL}}{}_l \text{ ; for all } l \in \mathcal{L}\\
\sigma_3(\mathbf{St}^{\mathsf{Prop}}) &= \mathbf{St}^{\mathsf{Prop}}\\
\sigma_3(\mathbf{P}^{\mathsf{Prop}}{}_l) &= \mathbf{P}^{\mathsf{Prop}}{}_l \text{ ; for all } l \in \mathcal{L}\\
\sigma_4(\mathbf{St}^{\mathsf{FOL}}) &= \mathbf{St}^{\mathsf{FOL}}\\
\sigma_4(\mathbf{P}^{\mathsf{FOL}}{}_l) &= \mathbf{P}^{\mathsf{FOL}}{}_l \text{ ; for all } l \in \mathcal{L}
\end{aligned}$$

FIGURE 6.3. Relation between $\{\mathbf{P}^{\mathsf{Prop}}{}_l\}_{l\in\mathcal{L}}$ and $\{\mathbf{P}^{\mathsf{FOL}}{}_l\}_{l\in\mathcal{L}}$.

Once again, as we proposed at the end of §2.1, and thanks to the translation relation (${}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}$), it is possible to represent any Boolean constrain involving $\{\mathbf{P}^{\mathsf{Prop}}{}_i\}_{i\in\mathcal{I}^{\mathsf{Prop}}}$ and $\{\mathbf{P}^{\mathsf{FOL}}{}_i\}_{i\in\mathcal{I}^{\mathsf{FOL}}}$ by providing definitions analogous to Definitions 6.4 and 6.5. More complex constrains can be described by extending the language and translation to first-order constrains.

In the same way we did in Definition 6.4, the following definitions show how propositional variables and first-order predicates can be put together in a first-order language. Once again, Definition 6.7 uses a disjoint union of two sets of symbols to identify the partial specification where they were introduced to apply the corresponding translation to the ω-closure fork algebra signature.

Definition 6.6. Let F be a set of first-order function symbols, and $\mathcal{V}$ be a denumerable set of variable symbols. Then, the set of first-order terms is the smallest set $FOLTerms(F,\mathcal{V})$ such that:

- $\mathcal{V} \subseteq FOLTerms(F,\mathcal{V})$,
- if $\{t_1,\dots,t_n\} \subseteq FOLTerms(F,\mathcal{V})$ and $f \in F$ is an n-ary function symbol, then $f(t_1,\dots,t_n) \in FOLTerms(P,\mathcal{V})$.

Definition 6.7. Let P be a set of propositional variables, Q be a set of first-order predicate symbols, and $\mathcal{V}$ be a denumerable set of variable symbols. Then, the set of first-order constrains is the smallest set $X = FOLConst(P,Q,\mathcal{V})$ such that:

- $inl(p) \in X$, for all $p \in P$,
- if $\{t_1,\dots,t_n\} \subseteq FOLTerms(F,\mathcal{V})$ and $q \in Q$ is an n-ary predicate symbol, then $inr(q(t_1,\dots,t_n)) \in X$,
- if $\alpha,\beta \in X$, then $\{\neg\alpha, \alpha\vee\beta\} \subseteq X$.
- if $\alpha \in X$ and $v \in \mathcal{V}$, then $(\exists v)\alpha \in X$.

The rest of the propositional operators such as conjunction ($\wedge$), and implication ($\Longrightarrow$), are defined as always in terms of the negation ($\neg$) and disjunction ($\vee$) operators as $\alpha\wedge\beta \equiv \neg(\neg\alpha\vee\neg\beta)$, $\alpha \Longrightarrow \beta \equiv \neg\alpha\vee\beta$ and $(\forall x)\alpha \equiv \neg(\exists x)\neg\alpha$.

The translation from first-order constrains to algebraic axioms we will present in the following definition uses a formulation similar to the one we presented in Chapter 5 for FOLTL. Next definition presents this translation[4].

Definition 6.8. Let F be a set of first-order function symbols, $\mathcal{V}$ be a denumerable set of variable symbols, s be sequence of natural numbers, and $\mathcal{R}$ be a denumerable set of relation variables. Then, we define the translation tr_{FC}, as a function $tr_{FC} : FOLTerms(F,\mathcal{V}) \to RelDes(\mathcal{R})$ as follows:

$$\begin{array}{lcl} tr_{FC_s}(v) & \equiv & \begin{cases} \rho^{;(Ord(\mathcal{V}_v,s)-1)};\pi & \text{if } \mathcal{V}_v < length(s). \\ \rho^{;(length(s)-1)} & \text{if } \mathcal{V}_v = length(s). \end{cases} \\ tr_{FC_s}(f(t_1,\dots,t_n)) & \equiv & (tr_{FC_s}(t_1)\nabla\cdots\nabla tr_{FC_s}(t_n));\mathbf{F} \\ & & ;\text{ for all } f\in F \text{ and } Tr_{\mathsf{FOL}\to\mathsf{fPCFAU}}(f) \equiv \mathbf{F}. \end{array}$$

[4]As in Definition 6.5, the signature corresponding to the theory that incorporates the design decisions allows the application of both tr_{FC} and Tr_{FC} because it contains all the required symbols.

DEFINITION 6.9. Let P be a set of propositional variables, Q be a set of first-order predicate symbols, $\mathcal{V}$ be a denumerable set of variable symbols, s a sequence of natural numbers, and $\mathcal{R}$ be a denumerable set of relation variables. Then, we define the translation Tr_{FC} as a function $Tr_{FC} : FOLConst(P, Q, \mathcal{V}) \to RelDes(\mathcal{R})$ as follows:

$$\begin{array}{rcl} Tr_{FCs}(inr(p)) & \equiv & \left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}\right)^{\smile};\mathbf{P} \\ & & \quad ;\text{ for all } p \in P \text{ and } Tr_{\mathsf{Prop}\to\mathsf{fPCFAU}}(p) \equiv \mathbf{P}, \\ Tr_{FCs}(inr(q(t_1, \ldots, t_n))) & \equiv & \pi;(tr_{FCs}(t_1)\nabla\cdots\nabla tr_{FCs}(t_n));\mathbf{Q} \\ & & \quad ;\text{ for all } q \in Q \text{ and } Tr_{\mathsf{FOL}\to\mathsf{fPCFAU}}(q) \equiv \mathbf{Q}, \\ Tr_{FCs}(\neg\alpha) & \equiv & \mathbf{St}^{\mathsf{FOL}}\cdot\overline{Tr_{FCs}(\alpha)}\ , \\ Tr_{FCs}(\alpha \vee \beta) & \equiv & Tr_{FCs}(\alpha)+Tr_{FCs}(\beta)\ , \\ Tr_{FCs}((\exists v)\alpha) & \equiv & \Delta_{s,\mathcal{V}_v};Tr_{FC[s\oplus\mathcal{V}_v]}(\alpha)\ . \end{array}$$

where, if $l = length(s)$, $\Delta_{s,i}$ is defined by the following condition:

$$\Delta_{s,i} = \begin{cases} tr_{FCs}(\mathcal{V}_s[1])\nabla\cdots\nabla tr_{FCs}(\mathcal{V}_s[k-1])\nabla 1_{\mathsf{U}}\nabla tr_{FCs}(\mathcal{V}_s[k])\nabla\cdots\nabla tr_{FCs}(\mathcal{V}_s[l]) \\ \hfill ,\text{ if } k = Ord([s\oplus i], i) \leq l \\ tr_{FCs}(\mathcal{V}_s[1])\nabla\cdots\nabla tr_{FCs}(\mathcal{V}_s[l])\nabla 1_{\mathsf{U}} \\ \hfill ,\text{ if } Ord([s\oplus i], i) = l+1 \end{cases}$$

and can be understood as a cylindrification [**HMT71, HMT85**] on the $k^{th.}$ coordinate of a $length(s)$-dimensional space. To read about the details of the translation of first-order logic to fork algebras, the interested reader is pointed to [**Fri02**].

Let $\alpha \in FOLConst(P, Q, \mathcal{V})$. As we did in §2.1, the following formula can be added as axiom:

$$\mathbf{St}^{\mathsf{FOL}};Tr_{FC[]}(\alpha) = \mathbf{St}^{\mathsf{FOL}};1\ .$$

Once again, the use of $\mathbf{St}^{\mathsf{FOL}}$ as a filter at both sides of the equation states that the translation of α contains in its domain every first-order state, and therefore, holds in all of them.

2.3. Synchronizing propositional dynamic and first-order dynamic information. In §2.2 we showed how to relate propositional and first-order information by introducing a new relation symbol that acts as a translation between the propositional and the first-order versions of the system state. In this section we propose a methodology extending this approach to programs, to build up relationships between partial specifications written in propositional and first-order dynamic logic.

It is quite easy to see that what we presented in §2.2 can be used to relate propositional and first-order dynamic logic formulas in which no dynamic operator occurs. This is true but of no help in establishing relationships between the specification of programs.

Assume PDL-states and FODL-states are characterized by the relational constants $\mathbf{St}^{\mathsf{PDL}}$ and $\mathbf{St}^{\mathsf{FODL}}$, respectively, and consider the relational constant ${}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}$, which will play the same rôle ${}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FOL}}$ played in §2.2.

Now, to keep the example as simple as possible, we maintain the assumptions, made explicit in the previous section, that we want to relate a subset of each of the predicate symbols by means of an "equivalence" and that we have two different but complete characterizations of the set of system states. Thus, the following axioms should be added:

$$\begin{aligned}
&(31) & {}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}} &\leq \mathbf{St}^{\mathsf{PDL}};1;\mathbf{St}^{\mathsf{FODL}}\ ,\\
&(32) & \mathit{Dom}\left({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\right) &= \mathbf{St}^{\mathsf{PDL}}\ ,\\
&(33) & \mathit{Ran}\left({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\right) &= \mathbf{St}^{\mathsf{FODL}}\ ,\\
&(34) & {}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}} &= {}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}};\left({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\right)^{\smile};{}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\ ,\\
&(35) & {}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}};\mathbf{P}^{\mathsf{FODL}}{}_i &= \mathbf{P}^{\mathsf{PDL}}{}_i\ \text{, for all } i \in \mathcal{L},\\
&(36) & \left({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\right)^{\smile};\mathbf{P}^{\mathsf{PDL}}{}_i &= \mathbf{P}^{\mathsf{FODL}}{}_i\ \text{, for all } i \in \mathcal{L}.
\end{aligned}$$

Equations (31)–(36) must be interpreted in the same way as Equations (25)–(30) of §2.2.

As we did for predicate symbols, let $\{\mathbf{A}^{\mathsf{PDL}}{}_j\}_{j\in\mathcal{J}^{\mathsf{PDL}}}$ and $\{\mathbf{A}^{\mathsf{FODL}}{}_j\}_{j\in\mathcal{J}^{\mathsf{FODL}}}$ be the set of propositional and first-order atomic actions, and let $\mathcal{M} \subseteq \mathcal{J}^{\mathsf{PDL}}$ and $\mathcal{M} \subseteq \mathcal{J}^{\mathsf{FODL}}$. Now assume that for all $m \in \mathcal{M}$ we believe that the program $\mathbf{A}^{\mathsf{FODL}}{}_m$ is a first-order refinement of the program $\mathbf{A}^{\mathsf{PDL}}{}_m$, which in this case means that $\mathbf{A}^{\mathsf{FODL}}{}_m$ allows us to observe the state change with a higher level of granularity than $\mathbf{A}^{\mathsf{PDL}}{}_m$ (i.e., we are able to observe the values of the state variables before and after the application of the program). Then, if we consider that atomic actions are nothing but operations that transform the system state, we can use the relational constant ${}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}$ to establish this relationship by adding the following axioms:

$$\begin{aligned}
&(37) & {}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}};\mathbf{A}^{\mathsf{FODL}}{}_m;\left({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\right)^{\smile} &= \mathbf{A}^{\mathsf{PDL}}{}_m\ \text{, for all } m \in \mathcal{M},\\
&(38) & \left({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\right)^{\smile};\mathbf{A}^{\mathsf{PDL}}{}_m;{}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}} &= \mathbf{A}^{\mathsf{FODL}}{}_m\ \text{, for all } m \in \mathcal{M}.
\end{aligned}$$

Equations (37)–(38) state that the propositional and first-order programs take the system from the same set of source states to the same set of target states. Figure 6.4 shows the theories involving all this information as the image of a diagram.

Now, recalling §2.3 and §2.2, we can consider a language capable of expressing design decisions involving a PDL and a FODL partial specification. The following definitions formalize this language by considering, as in the case of §2.3, the disjoint union of the set of the PDL propositions and the set of the FODL predicate symbols; and the disjoint union of the set of the PDL atomic programs and the set of the FODL atomic programs. The link between propositional and first-order states is established in the same way we did in §2.2 by introducing a relation mapping the two notions of state.

Definition 6.10. Let P be a set of propositional variables, Q be a set of first-order predicate symbols, A be a set of propositional atomic actions, B be a set of first-order atomic actions, and $\mathcal{V}$ be a denumerable set of variable symbols. Then, the set of first-order dynamic constrains is the smallest set $X = \mathit{FODLConst}(P, A, Q, B, \mathcal{V})$ such that:

$$\begin{tikzcd}[column sep=small]
 & & \begin{array}{l}\langle\langle\{\mathbf{St}^{\mathsf{PDL}},\mathbf{St}^{\mathsf{FODL}},{}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}}\}\cup\{\mathbf{P}^{\mathsf{PDL}}{}_l\}_{l\in\mathcal{L}}\cup\{\mathbf{P}^{\mathsf{FODL}}{}_l\}_{l\in\mathcal{L}}\cup\\ \quad\{\mathbf{A}^{\mathsf{PDL}}{}_m\}_{m\in\mathcal{M}}\cup\{\mathbf{A}^{\mathsf{FODL}}{}_m\}_{m\in\mathcal{M}}\rangle,\\ \quad\{Eq.\ 31, Eq.\ 32, Eq.\ 33, Eq.\ 34\}\cup\{Eq.\ 35_i\}_{i\in\mathcal{L}}\cup\{Eq.\ 36_i\}_{i\in\mathcal{L}}\cup\\ \quad\{Eq.\ 37_m\}_{m\in\mathcal{M}}\cup\{Eq.\ 38_m\}_{m\in\mathcal{M}}\rangle\end{array} & & \\
 & \langle\langle\{\mathbf{St}^{\mathsf{PDL}}\}\cup\{\mathbf{P}^{\mathsf{PDL}}{}_i\}_{i\in\mathcal{L}}\cup\{\mathbf{A}^{\mathsf{PDL}}{}_m\}_{m\in\mathcal{M}}\rangle,\emptyset\rangle \arrow[ur, "\sigma_3"] & & \langle\langle\{\mathbf{St}^{\mathsf{FODL}}\}\cup\{\mathbf{P}^{\mathsf{FODL}}{}_i\}_{i\in\mathcal{L}}\cup\{\mathbf{A}^{\mathsf{FODL}}{}_m\}_{m\in\mathcal{M}}\rangle,\emptyset\rangle \arrow[ul, "\sigma_4"'] & \\
Tr_{\mathsf{PDL}} \arrow[ur, leftarrow, "\sigma_1"] & & & & Tr_{\mathsf{FODL}} \arrow[ul, leftarrow, "\sigma_2"']
\end{tikzcd}$$

$$\begin{array}{rcl}
\sigma_1(\mathbf{St}^{\mathsf{PDL}}) & = & \mathbf{St}^{\mathsf{PDL}}\\
\sigma_1(\mathbf{P}^{\mathsf{PDL}}{}_l) & = & \mathbf{P}^{\mathsf{PDL}}{}_l \text{ ; for all } l\in\mathcal{L}\\
\sigma_1(\mathbf{A}^{\mathsf{PDL}}{}_m) & = & \mathbf{P}^{\mathsf{PDL}}{}_m \text{ ; for all } m\in\mathcal{M}\\
\sigma_2(\mathbf{St}^{\mathsf{FODL}}) & = & \mathbf{St}^{\mathsf{FODL}}\\
\sigma_2(\mathbf{P}^{\mathsf{FODL}}{}_l) & = & \mathbf{P}^{\mathsf{FODL}}{}_l \text{ ; for all } l\in\mathcal{L}\\
\sigma_2(\mathbf{A}^{\mathsf{FODL}}{}_m) & = & \mathbf{A}^{\mathsf{FODL}}{}_m \text{ ; for all } m\in\mathcal{M}\\
\sigma_3(\mathbf{St}^{\mathsf{PDL}}) & = & \mathbf{St}^{\mathsf{PDL}}\\
\sigma_3(\mathbf{P}^{\mathsf{PDL}}{}_l) & = & \mathbf{P}^{\mathsf{PDL}}{}_l \text{ ; for all } l\in\mathcal{L}\\
\sigma_3(\mathbf{A}^{\mathsf{PDL}}{}_m) & = & \mathbf{A}^{\mathsf{PDL}}{}_m \text{ ; for all } m\in\mathcal{M}\\
\sigma_4(\mathbf{St}^{\mathsf{FODL}}) & = & \mathbf{St}^{\mathsf{FODL}}\\
\sigma_4(\mathbf{P}^{\mathsf{FODL}}{}_l) & = & \mathbf{P}^{\mathsf{FODL}}{}_l \text{ ; for all } l\in\mathcal{L}\\
\sigma_4(\mathbf{A}^{\mathsf{FODL}}{}_m) & = & \mathbf{A}^{\mathsf{FODL}}{}_m \text{ ; for all } m\in\mathcal{M}
\end{array}$$

FIGURE 6.4. Relation between $\{\mathbf{P}^{\mathsf{PDL}}{}_l\}_{l\in\mathcal{L}}$ and $\{\mathbf{P}^{\mathsf{FODL}}{}_l\}_{l\in\mathcal{L}}$, and $\{\mathbf{A}^{\mathsf{PDL}}{}_m\}_{m\in\mathcal{M}}$ and $\{\mathbf{A}^{\mathsf{FODL}}{}_m\}_{m\in\mathcal{M}}$.

- $inl(p)\in X$, for all $p\in P$,
- if $\{t_1,\ldots,t_n\}\subseteq FOLTerms(F,s)$ and $q\in Q$ is an n-ary predicate symbol, then $inr(q(t_1,\ldots,t_n))\in X$,
- if $\alpha,\beta\in X$, then $\{\neg\alpha,\alpha\vee\beta\}\subseteq X$.
- if $\alpha\in X$ and $v\in\mathcal{V}$, then $(\exists v)\alpha\in X$.
- if $a\in A$ and $\alpha\in X$, then $\langle inl(a)\rangle\alpha\in X$,
- if $b\in B$ and $\alpha\in X$, then $\langle inr(b)\rangle\alpha\in X$,

Next definition presents the translation of first-order dynamic constrains to algebraic axioms.

DEFINITION 6.11. Let P be a set of propositional variables, Q be a set of first-order predicate symbols, A be a set of propositional atomic actions, B be a set of first-order atomic actions, $\mathcal{V}$ be a denumerable set of variable symbols, s a sequence on natural numbers, and $\mathcal{R}$ be a set of relation variables. Then, we define the translation Tr_{FC} as a function $Tr_{FDC} : FODLConst(P,A,Q,B,\mathcal{V})\to RelDes(\mathcal{R})$ as follows:

$$\begin{array}{rcl}
Tr_{FDC\,s}(inl(p)) & \equiv & ({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}})^{\smile};\mathbf{P}\\
 & & \qquad\text{; for all } p\in P \text{ and } Tr_{\mathsf{PDL}\to\mathsf{fPCFAU}}(p)\equiv\mathbf{P},\\
Tr_{FDC\,s}(inr(q(t_1,\ldots,t_n))) & \equiv & \pi;(tr_{FC}(t_1)\nabla\cdots\nabla tr_{FC}(t_n));\mathbf{Q}\\
 & & \qquad\text{; for all } q\in Q \text{ and } Tr_{\mathsf{FODL}\to\mathsf{fPCFAU}}(q)\equiv\mathbf{Q},\\
Tr_{FDC\,s}(\neg\alpha) & \equiv & \mathbf{St}^{\mathsf{FODL}}\cdot\overline{Tr_{FDC}(\alpha)}\ ,\\
Tr_{FDC\,s}(\alpha\vee\beta) & \equiv & Tr_{FDC}(\alpha)+Tr_{FDC}(\beta)\ ,\\
Tr_{FDC\,s}((\exists v)\alpha) & \equiv & \Delta_{s,\mathcal{V}_v};Tr_{FDC}(\alpha)\ ,\\
Tr_{FDC\,s}(\langle inl(a)\rangle\alpha) & \equiv & ({}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}})^{\smile};\mathbf{A};{}^{\mathsf{PDL}}\mathbf{St}^{\mathsf{FODL}};Tr_{FDC}(\alpha)\\
 & & \qquad\text{; for all } a\in A \text{ and } Tr_{\mathsf{PDL}\to\mathsf{fPCFAU}}(a)\equiv\mathbf{A},\\
Tr_{FDC\,s}(\langle inr(b)\rangle\alpha) & \equiv & \mathbf{B};Tr_{FDC}(\alpha)\\
 & & \qquad\text{; for all } b\in B \text{ and } Tr_{\mathsf{FODL}\to\mathsf{fPCFAU}}(b)\equiv\mathbf{B}.
\end{array}$$

As in §2.2, the translation of logical formulas yields relation designations. Therefore if we want the formula $\alpha\in FODLConst(P,A,Q,B,\mathcal{V})$ to be valid on every possible state of the system, we need to add the following formula as an axiom:

$$\mathbf{St}^{\mathsf{FODL}};Tr_{FDC[]}(\alpha) = \mathbf{St}^{\mathsf{FODL}};1$$

which, again, is an equation and states that the translation of α contains in its domain every first-order dynamic state.

Once again, as in previous sections, the proof of the semantics preservation of this translation is not provided.

2.4. Synchronizing first-order static and temporal information. The case of having two partial specifications written in first-order logics is not important from the technical point of view because of the simplicity of the solution, but we considered its inclusion because it gives this chapter a sense of completeness.

Structural properties can be naturally specified by properties written in first-order logic. If we consider two partial specifications, the first one containing the structural properties of the system (written in first-order logic), and the other containing a first-order temporal description of the behavior of the system, we might like the first set of properties to hold in every state of every system execution trace. Now, if we modeled the execution traces as sequences of states taken from $\mathbf{St}^{\mathsf{FOLTL}}$ as we did in Chapter 5, and assuming we have complete characterizations of first-order states and first-order temporal states, the only equation we have to add as a design decision is:

$$\mathbf{St}^{\mathsf{FOL}} = \mathbf{St}^{\mathsf{FOLTL}} \tag{39}$$

As first-order logic properties are satisfied by every possible first-order state, by Equation (39) we can conclude that every temporal state also satisfies the first-order properties and, consequently, every state of any execution trace (which are infinite right degenerated trees of states (see Equation (15))), satisfies these properties.

The assumption of having two different but equivalent characterizations of the set of states is not the most common situation thus, turning to a more realistic example, we need to consider having partial specifications of the system state, which means that system variables will be the union of both sets of variables, even considering a non-empty set of shared variables. Let $\mathcal{V}^{\mathsf{FOL}}$ and $\mathcal{V}^{\mathsf{FOLTL}}$ be denumerable sets of variable symbols, $\mathcal{T}^{\mathsf{FOL}}, \mathcal{T}^{\mathsf{FOLTL}} \subseteq \mathbb{N}$ be finite sets and $\tau : \mathcal{T}^{\mathsf{FOL}} \to \mathcal{T}^{\mathsf{FOLTL}}$ a bijective function.

$\mathcal{T}^{\mathsf{FOL}}$ and $\mathcal{T}^{\mathsf{FOLTL}}$ can be interpreted as the positions of the shared variables in the total order of the corresponding set, Then, τ represents the correspondence between static and temporal variables.

Then, if we consider ${}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}$ a relational constant that allows the translation from first-order system states to temporal system states, and viceversa, we would like ${}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}$ (when interpreted on a full proper closure fork algebra with urelements), to behave in the following way, for all $\langle s, s\rangle \in \mathbf{St}^{\mathsf{FOL}}$:

$$Ran\left(\{\langle s,s\rangle\};{}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}\right) = \left\{ \langle s', s'\rangle \in \mathbf{St}^{\mathsf{FOLTL}} \;\middle|\; (\forall i)(i \in \mathcal{T}^{\mathsf{FOL}} \Longrightarrow s_i = s'_{\tau(i)}) \right\} . \tag{40}$$

Formula (40) states that the relational constant ${}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}$ must preserve the values of the variables shared by the FOL-specification and the FOLTL-specification. Now it only rests to provide axioms characterizing the relational constant ${}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}$.

$$\begin{aligned}
&(41) & {}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}} &\leq \mathbf{St}^{\mathsf{FOL}};1;\mathbf{St}^{\mathsf{FOLTL}}\ ,\\
&(42) & \mathit{Dom}\left({}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}\right) &= \mathbf{St}^{\mathsf{FOL}}\ ,\\
&(43) & \mathit{Ran}\left({}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}\right) &= \mathbf{St}^{\mathsf{FOLTL}}\ ,\\
&(44) & {}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}} &= {}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}};\left({}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}\right)^{\smile};{}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}\ ,\\
&(45) & {}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}} &\leq \mathbf{St}^{\mathsf{FOL}};\left(\delta_{\mathcal{T}^{\mathsf{FOL}}{}_1}\nabla\cdots\nabla\delta_{\mathcal{T}^{\mathsf{FOL}}{}_{\#(\mathcal{T}^{\mathsf{FOL}})}}\right);\\
&(46) & &\quad\left(\mathbf{St}^{\mathsf{FOLTL}};\left(\delta_{\mathcal{T}^{\mathsf{FOLTL}}{}_{\tau(1)}}\nabla\cdots\nabla\delta_{\mathcal{T}^{\mathsf{FOL}}{}_{\tau(\#(\mathcal{T}^{\mathsf{FOL}}))}}\right)\right)^{\smile}\\
& & &\quad;\text{ where } \delta_t = \rho^{;t}
\end{aligned}$$

Once again, Equations (41)–(44) must be interpreted in the same way as Equations (25)–(28) of §2.2. Finally, Equation (45) force ${}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}$ to preserve the values of the shared variables when translating system states in any direction.

Then, Equation (39) can be rephrased using ${}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}$ by the following two equations:

$$\begin{aligned}
&(47) & \mathbf{St}^{\mathsf{FOLTL}} &= \mathit{Ran}\left(\mathbf{St}^{\mathsf{FOL}};{}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}\right)\ ,\\
&(48) & \mathbf{St}^{\mathsf{FOL}} &= \mathit{Ran}\left(\mathbf{St}^{\mathsf{FOLTL}};\left({}^{\mathsf{FOL}}\mathbf{St}^{\mathsf{FOLTL}}\right)^{\smile}\right)\ .
\end{aligned}$$

Notice that the relationship expressed by Equations (47) and (48) explicitly states that every first-order state can be translated to a set of temporal states and viceversa, and this condition does not necessarily hold for any two partial specifications.

3. Concluding remarks

The reader may have noticed that developing solutions for combinations of logics, as we did in §2.1, §2.2, §2.3 and §2.4, is somehow *ad-hoc*, specially considering that the solutions we presented are useful in combining logics under particular requirements. We believe this is a key point of this approach for dealing with heterogeneous specifications. Introducing design decisions requires creativity to recover global behavior lost during specification, and the use of a common language to interpret systems properties allows us to introduce any interaction between behaviors that were described in separate theories (possibly written in different logics). On the other hand, any solution we could present is not more than an example of what this framework can do for us, being more a methodological suggestion than a rule to solve any combination of descriptions written with a certain pair of logics.

As we mentioned in §2.1, it is not our aim to provide a logical language to express design decisions, thus we simply proposed a language, a translation, which most of the times is inspired on existing ones, and did not concentrate on providing correctness proofs for our proposals.

Most of this chapter requires further formalization. We believe the field of general logics must be extended with some kind of formal support to handle this inter-logic reasoning. In this field we found very motivating the vast amount of work on Grothendieck institutions [**Dia02**] and the informal talks we had with Till Mossakowski on this topic during AMAST 2006.

CHAPTER 7

Case-study: An interactive museum guide

In this chapter we will present a small example showing the usage of our approach in the homogenization of heterogeneous partial specifications in ω-closure fork algebras with urelements. The example we present in §1 was partially introduced in [**FGLR05**] but it was never fully developed.

1. Conceptual description

At a museum of arts it is possible to use wireless enabled PDAs[1] to visualize the information of the paintings being exhibited. Sensors are located next to paintings, and whenever a visitor stands in front of a painting, information about it is transferred to the visitor's PDA. While some operations in the system are simple, and can therefore be performed almost instantaneously, retrieving information about a specific painting may take longer time because audio and video might need to be transferred.

Whenever the visitor subscribes to this service, the visitor's position begins being monitored. If information about a painting has been requested and the visitor has moved to a different location, then information should not be delivered, and information about the current painting is requested instead.

In order to help the reader in understanding the running example, we will exhibit some UML-like diagram[**OMG04b**], together with some OCL constrains[**OMG04a**]. The way this diagrams and constrains are used is rudimentary and their only purpose is to enhance the reader comprehension of the example. Typically OCL constrains are placed in text boxes within the diagram linked to objects by dashed lines but we will present them in separate tables in order to make, both the diagrams and the OCL constrains, easier to read.

In Figure 7.1, we present a class diagram describing the structure of the system. A monitor is in charge of keeping track of the visitors positions,

[1]Personal Digital Assistant, a handheld device that combines computing, Internet and networking features.

based on information acquired by sensors located next to paintings. The museum guide is in charge of sending the right information to visitors. This is done by correctly mapping a visitors position, notified by the monitor whenever it changes, to the painting located in there.

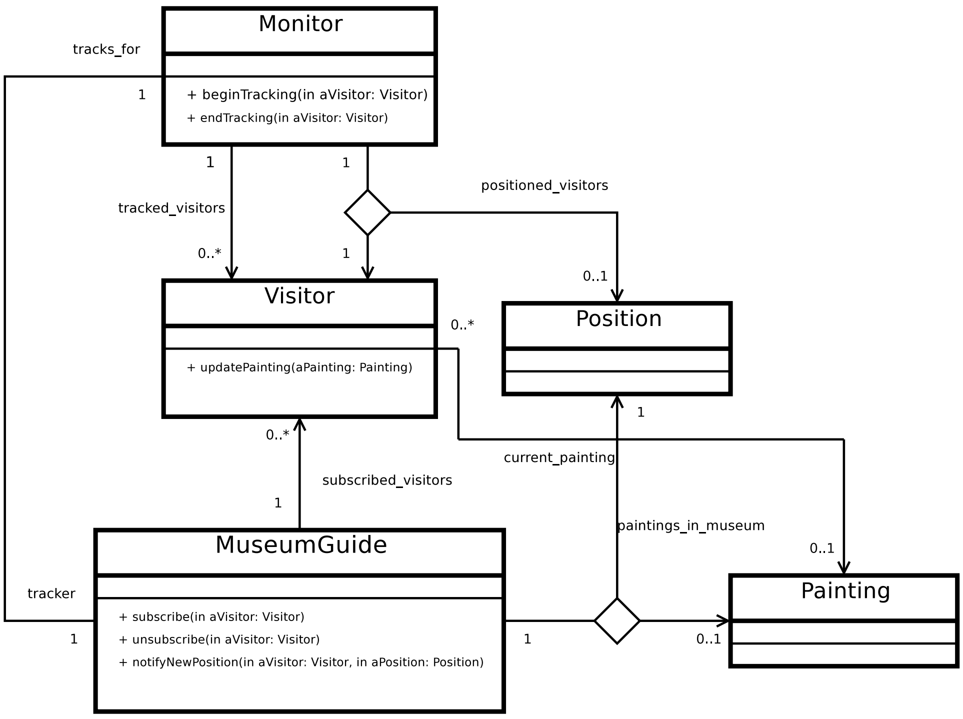

Monitor
+ beginTracking(in aVisitor: Visitor)
+ endTracking(in aVisitor: Visitor)
tracks_for
1
1
1
tracked_visitors
0..*
1
positioned_visitors
0..1
Visitor
+ updatePainting(aPainting: Painting)
0..*
Position
1
0..*
current_painting
subscribed_visitors
1
paintings_in_museum
0..1
MuseumGuide
+ subscribe(in aVisitor: Visitor)
+ unsubscribe(in aVisitor: Visitor)
+ notifyNewPosition(in aVisitor: Visitor, in aPosition: Position)
tracker
1
1
0..1
Painting

FIGURE 7.1. Class diagram for the Interactive Museum System.

As not every instance of the class diagram presented in Figure 7.1 is valid, in Table 7.1, we provide the structural restrictions formalized as OCL invariants.

Table 7.2 shows the formalization of the pre and post-condition of the methods declared in the classes appearing in Figure 7.1. Notice that the post-condition of the operation `notifyNewPosition` was declared to be `true`. This is because its application does not change neither the associations, nor the attributes accessible from the entity `MuseumGuide`.

The initial state of the system in the class diagram of Figure 7.1 is presented by resorting to the object diagram of Figure 7.2. Table 7.3 shows its formalization in OCL.

Figures 7.3–7.6 present sequence diagrams providing behavioral information about the system described before. The text box appearing within the sequence diagram denotes OCL restrictions on the message passing.

When considering the implementation of the system in a real museum, the maximum number of visitors we are able to subscribe to our guiding service is fixed and depends on the actual capacity of the museum. On the other hand, pictures and positions usually change because they depend on the art pieces being exhibited in the museum. Based on this fact, we will use propositional logics when extracting information only involving visitors, For the class *MuseumGuide* we have:

```
context MuseumGuide
    inv: paintings_in_museum→forall (t, t' | position (t) = position(t') implies
        painting(t) = painting(t'))
context MuseumGuide
    inv: paintings_in_museum→forall (t, t' | painting (t) = painting(t') implies
        position(t) = position(t'))
context MuseumGuide
    inv: subscribed_visitors→includes(tracker→tracked_visitors)
context MuseumGuide
    inv: (paintings_in_museum→collect(t | position(t)))→
        includes((tracker→positioned_visitors)→collect(t | position(t)))
context MuseumGuide
    inv: (paintings_in_museum→collect(t | painting(t)))→
        includes(subscribed_visitors→collect(v | v.current_painting))
```

For the class *Monitor* we have:

```
context Monitor
    inv: tracked_visitors→includesAll(positioned_visitors→
        collect(t | visitor(t)))
context Monitor
    inv: positioned_visitors→forall (t, t' | visitor (t) = visitor(t') implies
        position(t) = position(t'))
context Monitor
    inv: positioned_visitors→forall (t, t' | position (t) = position(t') implies
        visitor(t) = visitor(t'))
```

TABLE 7.1. List of methods of the classes of Fig. 7.1.

For the class *MuseumGuide* we have:

```
context MuseumGuide :: subscribe(aVisitor: Visitor)
    pre: self.subscribed_visitors→excludes(aVisitor)
    post: self.subscribed_visitors→includes(aVisitor)
context MuseumGuide :: unsubscribe(aVisitor: Visitor)
    pre: self.subscribed_visitors→includes(aVisitor)
    post: self.subscribed_visitors→excludes(aVisitor)
context MuseumGuide :: notifyNewPosition(aVisitor: Visitor, aPosition: Position)
    pre: self.subscribed_visitors→includes(aVisitor)
    post: true
```

For the class *Monitor*, we have:

```
context Monitor :: beginTracking(aVisitor: Visitor)
    pre: self.tracked_visitors→excludes(aVisitor)
    post: self.tracked_visitors→includes(aVisitor)
context Monitor :: endTracking(aVisitor: Visitor)
    pre: self.tracked_visitors→includes(aVisitor)
    post: self.tracked_visitors→excludes(aVisitor)
```

For the class *Visitor*, we have:

```
context Visitor :: updatePainting(aPainting: Painting)
    pre: true
    post: self.current_painting = aPainting
```

TABLE 7.2. OCL formalization of the methods the classes of Figure 7.1.

and first-order logics when extracting information involving pictures and positions.

2. Logical description

In this section we will derive logical properties from the conceptual description presented in §1. To choose the logics we will use in this process we

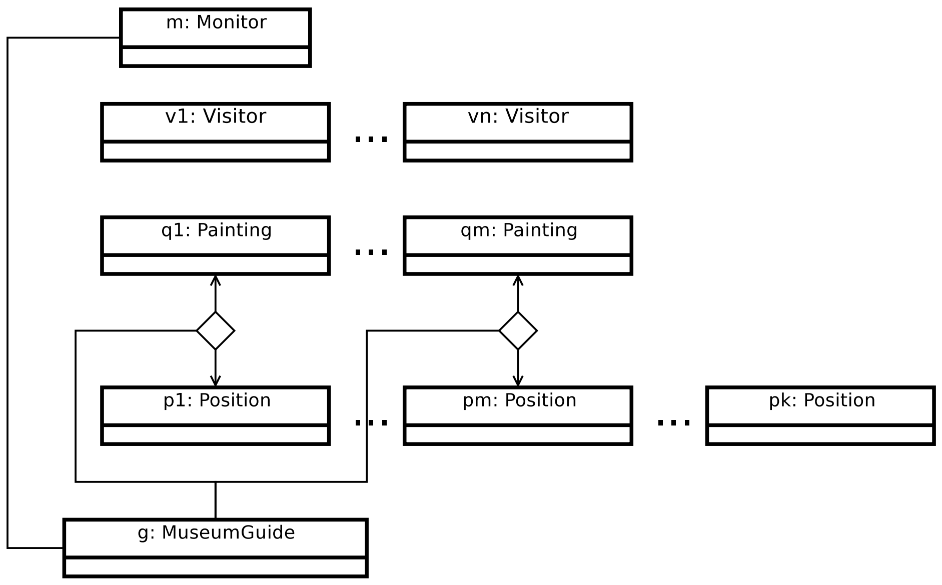


FIGURE 7.2. Object diagram for the Interactive Museum System.

For the class *MuseumGuide* we have:
context `MuseumGuide` :: `subscribed_visitors`: `Set (Visitor)`
init: `Set { }`
For the class *Monitor*, we have:
context `Monitor` :: `tracked_visitors`:
`Set (Visitor)`
init: `Set { }`
context `Monitor` :: `positioned_visitors`:
`Set (Tuple (visitor: Visitor, position: Position))`
init: `Set { }`
For the class *Visitor*, we have:
context `Visitor` :: `current_painting`: `Painting`
init: `<undefined>`

TABLE 7.3. OCL formalization of the objects diagram of Figure 7.1.

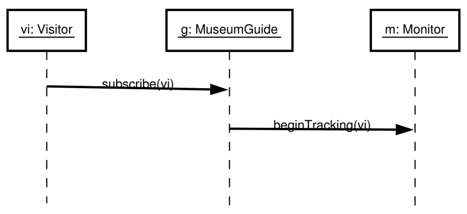


FIGURE 7.3. The subscription process.

will use both the functional and non-functional requirements. In that sense, and recalling the limit on the number of visitors to consider, operations and interactions only involving visitors will be formalized by means of propositional logics because we can represent the values for attributes and possible associations by introducing a limited number of propositions. On the other hand, values for the attributes and possible associations involving entities

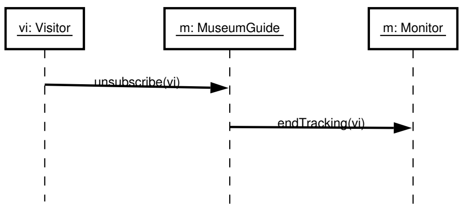


FIGURE 7.4. The unsubscription process.

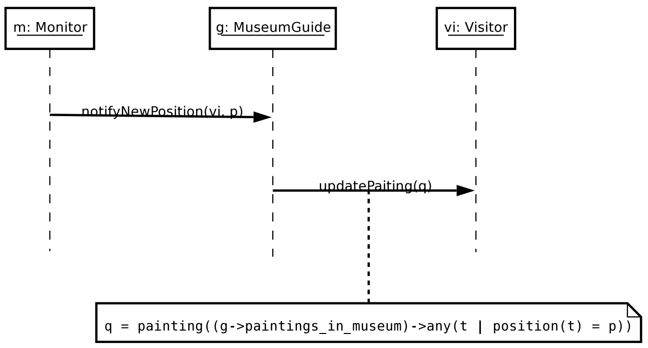


FIGURE 7.5. When the visitor moves to a new position, he may receive information about the painting being watched.

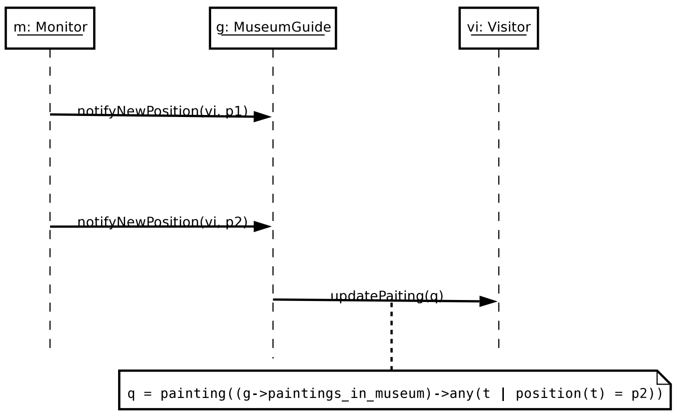

m: Monitor
g: MuseumGuide
vi: Visitor
notifyNewPosition(vi, p1)
notifyNewPosition(vi, p2)
updatePaiting(q)
q = painting((g->paintings_in_museum)->any(t | position(t) = p2))

FIGURE 7.6. When changing position, the visitor receives data about the painting in the the new position.

that are not limited can be formalized in a more natural way by resorting to first-order logics

2.1. A FOL view of the system. The class diagram of Figure 7.1 gives us the attributes and possible associations between objects. Describing the state of the system amounts to describing specific object diagrams. FOL will be used to describe the structural properties of the system. These structural properties of the design tell us which instances (object diagrams) are valid and which are not.

In order to obtain a FOL specification we need to interpret the elements occurring in the class diagram as logical objects. To do this we will interpret classes as sorts (of course this implies we will use many-sorted signatures), thus values will be elements of these sorts, and an n-association will be interpreted as an n-ary predicate symbol. For instance, the binary association `subscribed_visitors` gives rise to a binary predicate symbol $\texttt{subscribed\_visitors} \subseteq \text{Guide} \times \text{Visitor}$.

Following these guidelines, we obtain the FOL specification of Figure 7.7.

$$
\begin{array}{l}
\langle\ \langle\ \{\ \text{Guide}, \text{Visitor}, \text{Position}, \text{Painting}\ \},\\
\quad \{\ \},\\
\quad \{\ \texttt{subscribed\_visitors}, \texttt{current\_painting}, \texttt{paintings\_in\_museum}, \texttt{tracker},\\
\qquad \texttt{tracked\_visitors}, \texttt{positioned\_visitors}, \texttt{Init\_Guide}, \texttt{Init\_Monitor},\\
\qquad \texttt{Init\_Visitor}\ \}\ \rangle,\\
\{\ (\forall g : \text{Guide}, p, p' : \text{Position}, q, q' : \text{Painting})(\texttt{paintings\_in\_museum}(g,p,q) \wedge\\
\qquad \texttt{paintings\_in\_museum}(g,p',q') \Longrightarrow (p = p' \Longrightarrow q = q'))\ ,\\
\ (\forall g : \text{Guide}, p, p' : \text{Position}, q, q' : \text{Painting})(\texttt{paintings\_in\_museum}(g,p,q) \wedge\\
\qquad \texttt{paintings\_in\_museum}(g,p',q') \Longrightarrow (q = q' \Longrightarrow p = p'))\ ,\\
\ (\forall g : \text{Guide}, m : \text{Monitor})(\texttt{tracker}(g,m) \Longrightarrow\\
\qquad (\forall v : \text{Visitor}, p : \text{Position})(\texttt{positioned\_visitors}(m,v,p) \Longrightarrow\\
\qquad\quad (\exists q : \text{Painting})(\texttt{paintings\_in\_museum}(g,p,q))))\ ,\\
\ (\forall g : \text{Guide}, v : \text{Visitor})(\texttt{subscribed\_visitors}(g,v) \Longrightarrow\\
\qquad (\forall q : \text{Painting})(\texttt{current\_painting}(v,q) \Longrightarrow\\
\qquad\quad (\exists p : \text{Position})(\texttt{paintings\_in\_museum}(g,p,q))))\ ,\\
\ (\forall m : \text{Monitor}, v : \text{Visitor})((\exists p : \text{Position})(\texttt{positioned\_visitors}(m,v,p)) \Longrightarrow\\
\qquad \texttt{tracked\_visitors}(m,v)))\ ,\\
\ (\forall m : \text{Monitor}, v, v' : \text{Visitor}, p, p' : \text{Position})(\texttt{positioned\_visitors}(m,v,p) \wedge\\
\qquad \texttt{positioned\_visitors}(m,v',p') \Longrightarrow (v = v' \Longrightarrow p = p'))\ ,\\
\ (\forall m : \text{Monitor}, v, v' : \text{Visitor}, p, p' : \text{Position})(\texttt{positioned\_visitors}(m,v,p) \wedge\\
\qquad \texttt{positioned\_visitors}(m,v',p') \Longrightarrow (p = p' \Longrightarrow v = v'))\ ,\\
\ (\forall v : \text{Visitor}, q, q' : \text{Painting})((\texttt{current\_painting}(v,q) \wedge\\
\qquad \texttt{current\_painting}(v,q')) \Longrightarrow (q = q'))\ ,\\
\ (\forall g : \text{Guide})(\texttt{Init\_Guide}(g) \Leftrightarrow (\forall v : \text{Visitor})(\neg\texttt{subscribed\_visitors}(g,v)))\ ,\\
\ (\forall m : \text{Monitor})(\texttt{Init\_Monitor}(m) \Leftrightarrow\\
\qquad (\forall v : \text{Visitor}, p : \text{Position})(\neg\texttt{tracked\_visitors}(m,v,p)))\ ,\\
\ (\forall v : \text{Visitor})(\texttt{Init\_Visitor}(v) \Leftrightarrow (\forall q : \text{Painting})(\neg\texttt{current\_painting}(v,q)))\ \}\ \rangle
\end{array}
$$

FIGURE 7.7. First-order logic view of the system.

2.2. A PDL view of the system. PDL will be used to formalize the dynamic behavior of those operations which only mention attributes and possible associations involving visitors. To do that we will introduce atomic propositions corresponding to the different values an attribute may have, as well as representing particular associations among objects. In this case, the interpretation of the rôle `subscribed_visitors` produces, for each visitor v, a proposition $\texttt{subscribed}_v$.

The general procedure we will follow in order to obtain the PDL specification is to include an action for each message (with the same name, but spelled in italics), and use the atomic propositions to map OCL pre and post-conditions to appropriate PDL formulas. Partial correctness assertions using pre and post-conditions leave the behavior of a method in those states that do not satisfy the pre-condition, unspecified. We will refine these specifications by stating that in those states, methods must halt. This is done by including a PDL formula of the form $\neg\texttt{pre} \implies [\mathit{method}]\texttt{false}$. In order to economize notation, we will assume as a frame condition that actions can only alter the value of propositions explicitly modified.

Following these guidelines, we obtain the PDL specification of Figure 7.8.

$$\begin{array}{l}
\langle\ \langle\ \{\ \mathtt{subscribed\_visitors}_v, \mathtt{tracked\_visitors}_v\ \}_{v\in\text{Visitor}},\\
\quad \{\ \mathit{subscribe}_v, \mathit{unsubscribe}_v, \mathit{beginTracking}_v, \mathit{endTracking}_v\ \}_{v\in\text{Visitor}}\ \rangle,\\
\ \{\ \neg\mathtt{subscribed\_visitors}_v \implies [\mathit{subscribe}_v]\mathtt{subscribed\_visitors}_v\ ,\\
\quad \mathtt{subscribed\_visitors}_v \implies [\mathit{subscribe}_v]\mathtt{false}\ ,\\
\quad \mathtt{subscribed\_visitors}_v \implies [\mathit{unsubscribe}_v]\neg\mathtt{subscribed\_visitors}_v\ ,\\
\quad \neg\mathtt{subscribed\_visitors}_v \implies [\mathit{unsubscribe}_v]\mathtt{false}\ ,\\
\quad \neg\mathtt{tracked\_visitors}_v \implies [\mathit{beginTracking}_v]\mathtt{tracked\_visitors}_v\ ,\\
\quad \mathtt{tracked\_visitors}_v \implies [\mathit{beginTracking}_v]\mathtt{false}\ ,\\
\quad \mathtt{tracked\_visitors}_v \implies [\mathit{endTracking}_v]\neg\mathtt{tracked\_visitors}_v\ ,\\
\quad \neg\mathtt{tracked\_visitors}_v \implies [\mathit{endTracking}_v]\mathtt{false}\ \}_{v\in\text{Visitor}}\ \rangle
\end{array}$$

FIGURE 7.8. Propositional dynamic logic view of the system.

2.3. A FODL view of the system. FODL will be used to complete the dynamic description of the system. If we observe the rest of the functions described in Table 7.2, the operations we did not formalize are those involving positions and paintings.

As we did in §2.1, classes will be interpreted as sorts and a n-association will be interpreted as a n-ary predicate symbol.

In the same way we did for the construction of a PDL specification, we will map OCL pre and post-conditions to appropriate FODL formulas, which in this case will be first-order predicate logic formulas; and atomic actions will receive the same name as in the OCL description but spelled in italics. Partial correctness assertions will be treated in the same way we did in §2.2.

Following these guidelines, we obtain the FODL specification of Figure 7.9.

$$\begin{array}{l}
\langle\ \langle\ \{\ \text{Guide}, \text{Visitor}, \text{Position}\ \},\\
\quad \{\ \},\\
\quad \{\ \mathtt{subscribed\_visitors}, \mathtt{current\_painting}\ \},\\
\quad \{\ \mathit{notifyNewPosition}\ \}\ \rangle,\\
\ \{\ (\forall g : \text{Guide}, v : \text{Visitor}, p : \text{Position})(\mathtt{subscribed\_visitors}(g, v) \implies\\
\qquad\quad [\mathit{notifyNewPosition}(g, v, p)]\mathtt{true})\ ,\\
\quad (\forall g : \text{Guide}, v : \text{Visitor}, p : \text{Position})\\
\qquad\quad (\neg\mathtt{subscribed\_visitors}(g, v) \implies [\mathit{notifyNewPosition}(g, v, p)]\mathtt{false})\ ,\\
\quad (\forall m : \mathit{Monitor}, v : \text{Visitor}, q : \text{Painting})\\
\qquad\quad (\mathtt{true} \implies [\mathit{updatePainting}(v, q)]\mathtt{current\_painting}(v, q))\ \}\ \rangle
\end{array}$$

FIGURE 7.9. First-order dynamic logic view of the system.

2.4. A FOLTL view of the system. Since sequence diagrams only describe specific valid interactions between objects, it is not possible to extract general behaviors from them. Nevertheless, we extract a (partial) FOLTL specification from the description of the behavior of the system that will be consistent with the sequence diagrams of Figures 7.3–7.6. Associations will be treated in the same way we did in §2.1 and §2.3.

Figure 7.10 contains the FOLTL specification.

$$\begin{array}{l}
\langle\ \langle\ \{\ \text{Guide}, \text{Visitor}, \text{Position}, \text{Painting}\ \}, \\
\qquad \{\ \}, \\
\qquad \{\ \texttt{subscribed\_visitors}, \texttt{current\_painting}, \texttt{paintings\_in\_museum}, \texttt{tracked\_visitors}, \\
\qquad\qquad \texttt{tracker}\ \}\ \rangle, \\
\quad \{\ (\forall g : \text{Guide}, m : \text{Monitor})(\texttt{tracker}(g, m) \Longrightarrow \\
\qquad\qquad (\forall v : \text{Visitor})(\Box(\texttt{subscribed\_visitors}(g, v) \Longleftrightarrow \mathsf{X}(\texttt{tracked\_visitors}(m, v)))))\ , \\
\quad\ (\forall g : \text{Guide}, m : \text{Monitor}, p : \text{Position}, q : \text{Painting}) \\
\qquad\qquad ((\texttt{tracker}(g, m) \wedge \texttt{paintings\_in\_museum}(g, p, q)) \Longrightarrow (\forall v : \text{Visitor}) \\
\qquad\qquad\quad (\Box((\neg\texttt{positioned\_visitors}(m, v, p) \wedge \mathsf{X}(\texttt{positioned\_visitors}(m, v, p))) \Longrightarrow \\
\qquad\qquad\qquad \Diamond(\neg\texttt{positioned\_visitors}(m, v, p) \vee \texttt{current\_painting}(v, q)))))\ , \\
\quad\ (\forall g : \text{Guide}, m : \text{Monitor}, p : \text{Position}, q : \text{Painting}) \\
\qquad\qquad ((\texttt{tracker}(g, m) \wedge \texttt{paintings\_in\_museum}(g, p, q)) \Longrightarrow (\forall v : \text{Visitor}) \\
\qquad\qquad\quad (\Box((\mathsf{X}(\texttt{current\_painting}(v, q)) \wedge \neg\texttt{current\_painting}(v, q)) \Longrightarrow \\
\qquad\qquad\qquad \texttt{positioned\_visitors}(m, v, p))))\ , \\
\quad\ (\forall m : \text{Monitor}, v : \text{Visitor})(\Box(\texttt{tracked\_visitors}(m, v) \Longrightarrow \\
\qquad\qquad \Diamond(\neg\texttt{tracked\_visitors}(m, v) \vee (\exists p : \text{Position})(\texttt{positioned\_visitors}(m, v, p)))))\ , \\
\quad\ (\forall m : \text{Monitor}, v : \text{Visitor})(\Box(\texttt{tracked\_visitors}(m, v) \Longleftrightarrow \\
\qquad\qquad\quad \mathsf{X}((\exists p : \text{Position})(\texttt{positioned\_visitors}(m, v, p)))))\ \}\ \rangle
\end{array}$$

FIGURE 7.10. First-order linear temporal logic view of the system.

The first axiom reflects the sequence diagrams of Figures 7.3 and 7.4. The second axiom is consistent with the sequence diagrams of Figures 7.5 and 7.6. The rest of the axioms can not be derived from the sequence diagrams presented in §1 but describe some desired behaviors. The third axiom states that a visitor only receives the information of his current position. The fourth axiom is a progress condition, forcing visitors' behavior to evolve. And the last axiom states that once a visitor starts being tracked by the monitor, his position is immediately registered by the system.

3. Algebraic description

The algebraic description of the interactive museum guide is obtained from the logical description presented in §2 by applying the corresponding representation maps to the signatures and axioms involved in it. The translations we will use in this section were presented considering the case of mono-sorted logics, thus yielding mono-sorted algebras. Adapting them for the many-sorted case only requires to consider the existence of separate sets of urelements reflecting the sorts declared in the logical theory, and preserving the types declared for constants, functions and predicates.

3.1. Algebraic translation of the FOL view of the system. The original translation of FOL-formulas to fork algebraic equations can be found in [**Fri02**, Chapter 5]. This translation requires the FOL theory to be translated to an extension of ETBR first, and then to fork algebra. Frias and Orlowska presented in [**FO98**, Definition 3.13] a slightly different translation which does not requires this intermediate step. This approach is considerably closer to the rest the one followed in the developing of the rest of the translation we will use in this chapter. Figure 7.11 shows the translation of the FOL theory of Figure 7.7 to an ω-closure fork algebra with urelements theory.

$$
\begin{array}{l}
\langle\ \langle\ \{\ 1'_{\mathsf U}, 1_{\text{Guide}}, 1'_{\text{Guide}}, 1_{\text{Monitor}}, 1'_{\text{Monitor}}, 1_{\text{Visitor}}, 1'_{\text{Visitor}}, 1_{\text{Position}}, 1'_{\text{Position}}, \\
\qquad 1_{\text{Painting}}, 1'_{\text{Painting}}, \text{Paintings_in_museum}, \dots\ \}\ \rangle, \\
\ \{\ 1'_{\mathsf U} = \overline{(1\nabla 1)}\cdot 1'\ , \\
\quad 1'_{\text{Guide}} \le 1'_{\mathsf U}\ ,\ 1'_{\text{Monitor}} \le 1'_{\mathsf U}\ ,\ 1'_{\text{Visitor}} \le 1'_{\mathsf U}\ ,\ 1'_{\text{Position}} \le 1'_{\mathsf U}\ ,\ 1'_{\text{Painting}} \le 1'_{\mathsf U}\ , \\
\quad 1'_{\text{Guide}}\cdot 1'_{\text{Monitor}} = 0\ ,\ 1'_{\text{Guide}}\cdot 1'_{\text{Visitor}} = 0\ ,\ 1'_{\text{Guide}}\cdot 1'_{\text{Position}} = 0\ ,\ 1'_{\text{Guide}}\cdot 1'_{\text{Painting}} = 0\ , \\
\quad 1'_{\text{Monitor}}\cdot 1'_{\text{Visitor}} = 0\ ,\ 1'_{\text{Monitor}}\cdot 1'_{\text{Position}} = 0\ ,\ 1'_{\text{Monitor}}\cdot 1'_{\text{Painting}} = 0\ , \\
\quad 1'_{\text{Visitor}}\cdot 1'_{\text{Position}} = 0\ ,\ 1'_{\text{Visitor}}\cdot 1'_{\text{Painting}} = 0\ , \\
\quad 1'_{\text{Position}}\cdot 1'_{\text{Painting}} = 0\ , \\
\quad 1_{\text{Guide}} = 1'_{\text{Guide}};1;1'_{\text{Guide}}\ ,\ 1_{\text{Monitor}} = 1'_{\text{Monitor}};1;1'_{\text{Monitor}}\ ,\ 1_{\text{Visitor}} = 1'_{\text{Visitor}};1;1'_{\text{Visitor}}\ , \\
\quad 1_{\text{Position}} = 1'_{\text{Position}};1;1'_{\text{Position}}, 1_{\text{Painting}} = 1'_{\text{Painting}};1;1'_{\text{Painting}}\ , \\
\quad \left(1'_{\text{Guide}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Painting}}\right);\text{Paintings_in_museum};1 = \text{Paintings_in_museum}\ , \\
\vdots \\
\quad \Delta_1 = 1_{\text{Guide}}\ , \\
\quad \Delta_2 = \pi\nabla 1_{\text{Position}}\ , \\
\quad \Delta_3 = \pi\nabla\,(\rho;\pi)\,\nabla 1_{\text{Position}}\ , \\
\quad \Delta_4 = \pi\nabla\,(\rho;\pi)\,\nabla\,(\rho;\rho;\pi)\,\nabla 1_{\text{Painting}}\ , \\
\quad \Delta_5 = \pi\nabla\,(\rho;\pi)\,\nabla\,(\rho;\rho;\pi)\,\nabla\,(\rho;\rho;\rho;\pi)\,\nabla 1_{\text{Painting}}\ , \\
\quad {1'_{\mathsf U}}^{0} = 1'_{\mathsf U}\ , \\
\quad {1'_{\mathsf U}}^{1} = 1'_{\text{Guide}}\ , \\
\quad {1'_{\mathsf U}}^{2} = 1'_{\text{Guide}} \otimes 1'_{\text{Position}}\ , \\
\quad {1'_{\mathsf U}}^{3} = 1'_{\text{Guide}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Position}}\ , \\
\quad {1'_{\mathsf U}}^{4} = 1'_{\text{Guide}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Painting}}\ , \\
\quad {1'_{\mathsf U}}^{5} = 1'_{\text{Guide}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Painting}} \otimes 1'_{\text{Painting}}\ , \\
\quad \overline{{1'_{\mathsf U}}^{0};\Delta_1;\underline{(\,{1'_{\mathsf U}}^{1};\Delta_2;\,({1'_{\mathsf U}}^{2};\Delta_3;\,({1'_{\mathsf U}}^{3};\Delta_4;\,({1'_{\mathsf U}}^{4};\Delta_5;}} \\
\qquad\quad \overline{\overline{\left(\overline{{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};\ \begin{matrix}\pi\\ \nabla\\ \rho;\pi\\ \nabla\\ \rho;\rho;\rho;\pi\end{matrix}\ ;\text{Paintings_in_museum}}+\right.}} \\
\qquad\quad \overline{\overline{\overline{{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};\ \begin{matrix}\pi\\ \nabla\\ \rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\rho\end{matrix}\ ;\text{Paintings_in_museum}}+{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};\ \overline{\begin{matrix}\rho;\pi\\ \nabla\\ \rho;\rho;\pi\end{matrix}\ ;1'_{\text{Position}}}+}} \\
\qquad\quad \overline{\overline{{1'_{\mathsf U}}^{5};\ \begin{matrix}\rho;\rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\rho\end{matrix}\ ;1'_{\text{Painting}}}\left.\Big)\Big)\Big)\Big)\Big)\right.} = {1'_{\mathsf U}}^{0};1\ , \\
\quad \overline{{1'_{\mathsf U}}^{0};\Delta_1;\underline{(\,{1'_{\mathsf U}}^{1};\Delta_2;\,({1'_{\mathsf U}}^{2};\Delta_3;\,({1'_{\mathsf U}}^{3};\Delta_4;\,({1'_{\mathsf U}}^{4};\Delta_5;}} \\
\qquad\quad \overline{\overline{\left(\overline{{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};\ \begin{matrix}\pi\\ \nabla\\ \rho;\pi\\ \nabla\\ \rho;\rho;\rho;\pi\end{matrix}\ ;\text{Paintings_in_museum}}+\right.}} \\
\qquad\quad \overline{\overline{\overline{{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};\ \begin{matrix}\pi\\ \nabla\\ \rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\rho\end{matrix}\ ;\text{Paintings_in_museum}}+{1'_{\mathsf U}}^{5};{1'_{\mathsf U}}^{5};\ \overline{\begin{matrix}\rho;\pi\\ \nabla\\ \rho;\rho;\pi\end{matrix}\ ;1'_{\text{Painting}}}+}} \\
\qquad\quad \overline{\overline{{1'_{\mathsf U}}^{5};\ \begin{matrix}\rho;\rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\rho\end{matrix}\ ;1'_{\text{Position}}}\left.\Big)\Big)\Big)\Big)\Big)\right.} = {1'_{\mathsf U}}^{0};1\ , \\
\vdots \\
\ \}\ \rangle
\end{array}
$$

FIGURE 7.11. Algebraic translation of the FOL view of the system presented in Figure 7.7.

3.2. Algebraic translation of the PDL view of the system. The translation of PDL-formulas to fork algebraic equations was presented in [**FO98**, Definition 7.13]. Figure 7.12 shows the translation of the PDL theory of Figure 7.8 to an ω-closure fork algebra with urelements theory.

$$
\begin{array}{l}
\langle\ \langle\ \{\ {}_{\mathsf{U}}1, 1_{\mathsf{U}}, 1'_{\mathsf{U}}\ \} \cup \{\ \mathrm{Subscribed\_visitors}_v, \mathrm{Tracked\_visitors}_v, \mathrm{Subscribe}_v, \mathrm{Unsubscribe}_v, \\
\qquad \mathrm{BeginTracking}_v, \mathrm{EndTracking}_v\ \}_{v \in \mathrm{Visitor}}\ \rangle, \\
\ \{\ {}_{\mathsf{U}}1 = \overline{(1 \nabla 1)^{\smile}}\ , 1_{\mathsf{U}} = \overline{1 \nabla 1}\ , 1'_{\mathsf{U}} = ({}_{\mathsf{U}}1 ; 1_{\mathsf{U}}) \cdot 1'\ \} \cup \\
\ \{\ 1'_{\mathsf{U}} ; \mathrm{Subscribed\_visitors}_v ; 1 = \mathrm{Subscribed\_visitors}_v\ , \\
\quad 1'_{\mathsf{U}} ; \mathrm{Tracked\_visitors}_v ; 1 = \mathrm{Tracked\_visitors}_v\ , \\
\quad 1'_{\mathsf{U}} ; \mathrm{Subscribe}_v ; 1'_{\mathsf{U}} = \mathrm{Subscribe}_v\ , \\
\quad 1'_{\mathsf{U}} ; \mathrm{Unsubscribe}_v ; 1'_{\mathsf{U}} = \mathrm{Unsubscribe}_v\ , \\
\quad 1'_{\mathsf{U}} ; \mathrm{BeginTracking}_v ; 1'_{\mathsf{U}} = \mathrm{BeginTracking}_v\ , \\
\quad 1'_{\mathsf{U}} ; \mathrm{EndTracking}_v ; 1'_{\mathsf{U}} = \mathrm{EndTracking}_v\ , \\
\quad \mathrm{Subscribed\_visitors}_v + 1'_{\mathsf{U}} ; \overline{\mathrm{Subscribe}_v ; \left(1'_{\mathsf{U}} ; \overline{\mathrm{Subscribed\_visitors}_v}\right)} + \overline{{}_{\mathsf{U}}1} = 1\ , \\
\quad \left(1'_{\mathsf{U}} ; \overline{\mathrm{Subscribed\_visitors}_v}\right) + 1'_{\mathsf{U}} ; \overline{\mathrm{Subscribe}_v ; (1'_{\mathsf{U}} ; 1)} + \overline{{}_{\mathsf{U}}1} = 1\ , \\
\quad \left(1'_{\mathsf{U}} ; \overline{\mathrm{Subscribed\_visitors}_v}\right) + 1'_{\mathsf{U}} ; \overline{\mathrm{Unsubscribe}_v ; \mathrm{Subscribed\_visitors}_v} + \overline{{}_{\mathsf{U}}1} = 1\ , \\
\quad \mathrm{Subscribed\_visitors}_v + 1'_{\mathsf{U}} ; \overline{\mathrm{Unsubscribe}_v ; (1'_{\mathsf{U}} ; 1)} + \overline{{}_{\mathsf{U}}1} = 1\ , \\
\quad \mathrm{Tracked\_visitors}_v + 1'_{\mathsf{U}} ; \overline{\mathrm{BeginTracking}_v ; \left(1'_{\mathsf{U}} ; \overline{\mathrm{Tracked\_visitors}_v}\right)} + \overline{{}_{\mathsf{U}}1} = 1\ , \\
\quad \left(1'_{\mathsf{U}} ; \overline{\mathrm{Tracked\_visitors}_v}\right) + 1'_{\mathsf{U}} ; \overline{\mathrm{BeginTracking}_v ; (1'_{\mathsf{U}} ; 1)} + \overline{{}_{\mathsf{U}}1} = 1\ , \\
\quad \left(1'_{\mathsf{U}} ; \overline{\mathrm{Tracked\_visitors}_v}\right) + 1'_{\mathsf{U}} ; \overline{\mathrm{EndTracking}_v ; \mathrm{Tracked\_visitors}_v} + \overline{{}_{\mathsf{U}}1} = 1\ , \\
\quad \mathrm{Tracked\_visitors}_v + 1'_{\mathsf{U}} ; \overline{\mathrm{EndTracking}_v ; (1'_{\mathsf{U}} ; 1)} + \overline{{}_{\mathsf{U}}1} = 1\ \}_{v \in \mathrm{Visitor}}\ \rangle
\end{array}
$$

FIGURE 7.12. Algebraic translation of the PDL view of the system presented in Figure 7.8.

3.3. Algebraic translation of the FODL view of the system. The translation of FODL-formulas to fork algebraic equations was presented in [**FBM02**, Definition 15][2]. Figure 7.13 shows the translation of the FODL theory of Figure 7.9 to an ω-closure fork algebra with urelements theory.

3.4. Algebraic translation of the FOLTL view of the system. The translation of FOLTL-formulas to fork algebraic equations was presented in Definitions 5.8 and 5.9] (see also [**FL06**, Definitions 3.6 and 3.7]). Figure 7.14 shows the translation of the FOLTL theory of Figure 7.10 to an ω-closure fork algebra with urelements theory.

4. Putting the pieces together

4.1. Design decisions. In this section we will present the channels and theories introducing design decisions required to glue the partial algebraic specifications. As we showed in Chapter 6, Figure 7.15 shows the diagram scheme we use to introduce design decisions. Let $T = \langle \Sigma, \Gamma \rangle, T' = \langle \Sigma', \Gamma' \rangle \in Th_0^{\mathsf{fPCFAU}}$, then, as not all the symbols of Σ and Σ' participate in the channel

[2]If we recall [**FBM02**, Definition 15, Rule 1], the translation of an atomic action a is expressed as $M_\sigma(a) = (\Pi_{\sigma,\sigma_a} ; a \nabla \Pi_{\sigma,\sigma-\sigma_a}) ; \mathrm{Merge}_{\sigma_a,\sigma-\sigma_a}$. The intuition behind this translation rule is that the term Π_{σ,σ_a} extracts from σ those values for the free variables occurring in a, to which a will be applied; term $\Pi_{\sigma,\sigma-\sigma_a}$ retains the values of σ which will not change by the application of the action a. Finally, term $\mathrm{Merge}_{\sigma_a,\sigma-\sigma_a}$ put the values which were altered by a, and the values which did not change by the application of a, in the appropriate order. This rule overlooks the case $\sigma = \sigma_a$ because there are no values to extract by performing the ∇ with the term $\Pi_{\sigma,\sigma-\sigma_a}$ thus, no composition with the term $\mathrm{Merge}_{\sigma_a,\sigma-\sigma_a}$ is needed. The error can be corrected by the following expression:

$$
M_\sigma(a) = \begin{cases} \left(\begin{array}{c} \Pi_{\sigma,\sigma_a} ; a \\ \nabla \\ \Pi_{\sigma,\sigma-\sigma_a} \end{array} \right) ; \mathrm{Merge}_{\sigma_a,\sigma-\sigma_a} & \text{, if } \sigma \neq \sigma_a \\ \Pi_{\sigma,\sigma_a} ; a & \text{, if } \sigma = \sigma_a \end{cases} .
$$

$$
\begin{array}{l}
\langle\ \langle\ \{\ 1_{\mathrm{Guide}}, 1'_{\mathrm{Guide}}, 1_{\mathrm{Visitor}}, 1'_{\mathrm{Visitor}}, 1_{\mathrm{Position}}, 1'_{\mathrm{Position}}, 1_{\mathrm{Painting}}, 1'_{\mathrm{Painting}}, 1'_{\mathsf{U}}, \\
\qquad \mathrm{Subscribed\_visitors}, \mathrm{Current\_painting}, \mathrm{UpdatePainting}, \mathrm{NotifyNewPosition}\ \}\ \rangle, \\
\{\ 1'_{\mathsf{U}} = \overline{(1 \nabla 1)} \cdot 1'\ ,\ 1'_{\mathrm{Guide}} \leq 1'_{\mathsf{U}}, 1'_{\mathrm{Visitor}} \leq 1'_{\mathsf{U}}, 1'_{\mathrm{Position}} \leq 1'_{\mathsf{U}}, 1'_{\mathrm{Painting}} \leq 1'_{\mathsf{U}}\ , \\
1'_{\mathrm{Guide}} \cdot 1'_{\mathrm{Visitor}} = 0\ ,\ 1'_{\mathrm{Guide}} \cdot 1'_{\mathrm{Position}} = 0\ ,\ 1'_{\mathrm{Guide}} \cdot 1'_{\mathrm{Painting}} = 0\ , \\
1'_{\mathrm{Visitor}} \cdot 1'_{\mathrm{Position}} = 0\ ,\ 1'_{\mathrm{Visitor}} \cdot 1'_{\mathrm{Painting}} = 0\ ,\ 1'_{\mathrm{Position}} \cdot 1'_{\mathrm{Painting}} = 0\ , \\
1_{\mathrm{Guide}} = 1'_{\mathrm{Guide}} ; 1 ; 1'_{\mathrm{Guide}}\ ,\ 1_{\mathrm{Visitor}} = 1'_{\mathrm{Visitor}} ; 1 ; 1'_{\mathrm{Visitor}}\ ,\ 1_{\mathrm{Position}} = 1'_{\mathrm{Position}} ; 1 ; 1'_{\mathrm{Position}}\ , \\
1_{\mathrm{Painting}} = 1'_{\mathrm{Painting}} ; 1 ; 1'_{\mathrm{Painting}}\ , \\
(1'_{\mathrm{Guide}} \otimes 1'_{\mathrm{Visitors}})\ ; \mathrm{Subscribed\_visitors} ; 1 = \mathrm{Subscribed\_visitors}\ , \\
\left(1'_{\mathrm{Visitor}} \otimes 1'_{\mathrm{Painting}}\right) ; \mathrm{Current\_painting} ; 1 = \mathrm{Current\_painting}\ , \\
\left(1'_{\mathrm{Visitors}} \otimes 1'_{\mathrm{Painting}}\right) ; \mathrm{UpdatePainting} ; \left(1'_{\mathrm{Visitors}} \otimes 1'_{\mathrm{Painting}}\right) = \mathrm{UptadePainting}\ , \\
(1'_{\mathrm{Guide}} \otimes 1'_{\mathrm{Visitors}} \otimes 1'_{\mathrm{Position}})\ ; \mathrm{NotifyNewPosition} ; (1'_{\mathrm{Guide}} \otimes 1'_{\mathrm{Visitors}} \otimes 1'_{\mathrm{Position}}) = \\
\qquad \mathrm{NotifyNewPosition}\ , \\
\overline{1_{\mathrm{Guide}} ; \begin{matrix} 1'_{\mathrm{Guide}} \\ \nabla \\ 1_{\mathrm{Visitor}} \end{matrix} ; \begin{matrix} \pi \\ \nabla \\ \rho \\ \nabla \\ 1_{\mathrm{Position}} \end{matrix} ; \overline{\overline{\begin{matrix} \pi \\ \nabla \\ \rho ; \pi \end{matrix}} ; \mathrm{Subscribed\_visitors} + \overline{\begin{matrix} \pi \\ \nabla \\ \rho ; \pi \\ \nabla \\ \rho ; \rho \end{matrix} ; \mathrm{NotifyNewPosition} ; 0}}} = 1\ , \\
\overline{1_{\mathrm{Guide}} ; \begin{matrix} 1'_{\mathrm{Guide}} \\ \nabla \\ 1_{\mathrm{Visitor}} \end{matrix} ; \begin{matrix} \pi \\ \nabla \\ \rho \\ \nabla \\ 1_{\mathrm{Position}} \end{matrix} ; \overline{\begin{matrix} \pi \\ \nabla \\ \rho ; \pi \end{matrix} ; \mathrm{Subscribed\_visitors} + \overline{\begin{matrix} \pi \\ \nabla \\ \rho ; \pi \\ \nabla \\ \rho ; \rho \end{matrix} ; \mathrm{NotifyNewPosition} ; 1}}} = 1\ , \\
\overline{1_{\mathrm{Visitor}} ; \overline{\begin{matrix} 1'_{\mathrm{Visitor}} \\ \nabla \\ 1_{\mathrm{Painting}} \end{matrix} ; 0 + \overline{\begin{matrix} \pi \\ \nabla \\ \rho \end{matrix} ; \mathrm{UpdatePainting} ; \overline{\begin{matrix} \pi \\ \nabla \\ \rho \end{matrix} ; \mathrm{Current\_painting}}}}} = 1\ \}\ \rangle
\end{array}
$$

FIGURE 7.13. Algebraic translation of the FODL view of the system presented in Figure 7.9.

T_{DD}, and T_{DD} must include symbols to represent the elements of both T and T', we add the theories T_{Channel} and T'_{Channel} grouping those symbols of T and T', respectively, that will be related by axioms appearing in T_{DD}.

Whenever T_{DD} does not include any axiom, or the axioms are of the form $S^T = S^{T'}$, which means that they simply identify symbols from T and T' as the same elements, we will also use a simpler kind of channel avoiding the inclusion of superfluous theories in the diagram. Figure 7.16 shows the diagram scheme we will use for this case.

Notice that the diagram schemes presented in Figures 7.15 and 7.16 can be easily generalized to the case in which more than two theories are related throw the same channel.

From now on, the theories presented in Figure 7.11 – 7.14 will be denoted by $T_{\mathsf{FOL}} = \langle \Sigma_{\mathsf{FOL}}, \Gamma_{\mathsf{FOL}} \rangle$, $T_{\mathsf{PDL}} = \langle \Sigma_{\mathsf{PDL}}, \Gamma_{\mathsf{PDL}} \rangle$, $T_{\mathsf{FODL}} = \langle \Sigma_{\mathsf{FODL}}, \Gamma_{\mathsf{FODL}} \rangle$, and $T_{\mathsf{FOLTL}} = \langle \Sigma_{\mathsf{FOLTL}}, \Gamma_{\mathsf{FOLTL}} \rangle$, respectively. Unless we make it explicit, morphisms between signatures (theories) will be the identity function.

4.1.1. *Algebraic counterpart of domains and predicates.* If we observe the algebraic versions of the first-order partial specifications it is easy to notice that some constants of the form $1'_{\mathrm{domain}}$ were added to represents domains (sets of values) as partial identities ranging over urelements. Most of these constants are shared by all the specifications and should be interpreted as the same relation in every model of the specification of the whole system. Thus we will first introduce this relationship by resorting to two constructions like the one we showed in the diagram scheme of Figure 7.16.

Consider the two theories presented in Figure 7.17.

Then, in Figure 7.18 we show a diagram involving T_{FOL}, T_{FODL}, T_{FOLTL}, $T_{\mathsf{FOL} \leftrightarrow \mathsf{FODL}}$ and $T_{\mathsf{FOL} \leftrightarrow \mathsf{FOLTL}}$.

$$
\begin{array}{l}
\langle\ \langle\ \{\ \mathbf{St}, \mathbf{St_0}, \mathbf{T}, \mathbf{tr}, \{\mathbf{V}_k\}_{k\mathrm{N}},\\
\qquad\qquad \text{Tracker}, \text{Subscribed\_visitors}, \text{Tracked\_visitors}, \text{Paintings\_in\_museum},\\
\qquad\qquad \text{Positioned\_visitors}, \text{Current\_painting},\\
\qquad\qquad 1'_{\text{Guide}}, 1'_{\text{Monitor}}, 1'_{\text{Visitor}}, 1'_{\text{Position}}, 1'_{\text{Painting}}, 1'_{\mathsf{U}}\ \}\ \rangle,\\
\{\ 1'_{\mathsf{U}} = \overline{(1\nabla 1)}\cdot 1'\ ,\ 1'_{\text{Guide}} \leq 1'_{\mathsf{U}}, 1'_{\text{Visitor}} \leq 1'_{\mathsf{U}}, 1'_{\text{Position}} \leq 1'_{\mathsf{U}}, 1'_{\text{Painting}} \leq 1'_{\mathsf{U}}\ ,\\
\quad 1'_{\text{Guide}}\cdot 1'_{\text{Visitor}} = 0\ ,\ 1'_{\text{Guide}}\cdot 1'_{\text{Position}} = 0\ ,\ 1'_{\text{Guide}}\cdot 1'_{\text{Painting}} = 0\ ,\\
\quad 1'_{\text{Visitor}}\cdot 1'_{\text{Position}} = 0\ ,\ 1'_{\text{Visitor}}\cdot 1'_{\text{Painting}} = 0\ ,\ 1'_{\text{Position}}\cdot 1'_{\text{Painting}} = 0\ ,\\
\quad \mathbf{St} = 1'_{\text{Guide}} \otimes 1'_{\text{Monitor}} \otimes 1'_{\text{Visitor}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Painting}}\ ,\\
\quad \mathbf{St_0} \leq \mathbf{St}\ ,\\
\quad \left(\mathbf{T};1;\breve{\mathbf{T}}\right)\cdot 1' = \mathbf{St}\ ,\\
\quad \mathbf{tr} \leq 1'\ ,\\
\quad \breve{\pi};\mathbf{tr};\pi = \mathbf{St}\ ,\\
\quad \mathbf{tr} \leq \mathbf{St} \otimes \mathbf{tr}\ ,\\
\quad \mathbf{tr};\rho = \mathit{Ran}(\pi\nabla(\mathbf{T}\otimes\rho));\rho;\mathbf{tr}\ ,\\
\quad \left(1'_{\text{Guide}} \otimes 1'_{\text{Monitor}}\right);\text{Tracker};1 = \text{Tracker}\ ,\\
\quad \left(1'_{\text{Guide}} \otimes 1'_{\text{Visitors}}\right);\text{Subscribed\_visitors};1 = \text{Subscribed\_visitors}\ ,\\
\quad \left(1'_{\text{Monitor}} \otimes 1'_{\text{Visitor}}\right);\text{Tracked\_visitors};1 = \text{Tracked\_visitors}\ ,\\
\quad \left(1'_{\text{Guide}} \otimes 1'_{\text{Position}} \otimes 1'_{\text{Painting}}\right);\text{Paintings\_in\_museum};1 = \text{Paintings\_in\_museum}\ ,\\
\quad \left(1'_{\text{Monitor}} \otimes 1'_{\text{Visitor}} \otimes 1'_{\text{Position}}\right);\text{Positioned\_visitors};1 = \text{Positioned\_visitors}\ ,\\
\quad \left(1'_{\text{Visitor}} \otimes 1'_{\text{Painting}}\right);\text{Current\_painting};1 = \text{Current\_painting}\ ,
\end{array}
$$

$$
\mathbf{tr};\mathbf{V}_1;\mathbf{V}_2;\overline{\overline{\mathbf{tr};\mathbf{tr};\pi;\overline{\begin{array}{c}\pi\\ \nabla\\ \rho;\pi\end{array}};\text{Tracker}+\mathbf{tr};\overline{\mathbf{V}_3;\mathbf{tr};\overline{(\mathbf{tr};\rho)^*};}}}
$$

$$
\overline{\overline{\overline{\left(\mathbf{tr};\mathbf{tr};\mathbf{tr};\overline{\overline{\overline{\begin{array}{c}\pi\\ \nabla\\ \rho;\rho;\pi\end{array};\text{Subscribed\_visitors}}}+\rho;\begin{array}{c}\rho;\pi\\ \nabla\\ \rho;\rho;\pi\end{array};\text{Tracked\_visitors}+}\right.}}}
$$

$$
\overline{\overline{\overline{\overline{\left.\mathbf{tr};\mathbf{tr};\rho;\overline{\overline{\overline{\begin{array}{c}\rho;\pi\\ \nabla\\ \rho;\rho;\pi\end{array};\text{Tracked\_visitors}}}+\begin{array}{c}\pi\\ \nabla\\ \rho;\rho;\pi\end{array};\text{Subscribed\_visitors}}\right)}}}}
$$

$$
\overline{\mathbf{tr};\mathbf{V}_1;\mathbf{V}_2;\mathbf{V}_4;\mathbf{V}_5;\mathbf{tr};\overline{\overline{\left(\mathbf{tr};\overline{\begin{array}{c}\pi\\ \nabla\\ \rho;\pi\end{array};\text{Tracker}}\right)+\left(\mathbf{tr};\overline{\begin{array}{c}\pi\\ \nabla\\ \rho;\rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\rho\end{array};\text{Paintings\_in\_museum}}\right)+}}}
$$

$$
\overline{\overline{\overline{\overline{\mathbf{tr};\mathbf{V}_3;\mathbf{tr};(\mathbf{tr};\rho)^*;\left(\begin{array}{c}\rho;\pi\\ \nabla\\ \rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\pi\end{array};\text{Positioned\_visitors}+\mathbf{tr};\rho;\overline{\begin{array}{c}\rho;\pi\\ \nabla\\ \rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\pi\end{array};\text{Positioned\_visitors}+}\right.}}}}
$$

$$
\overline{\overline{\overline{\overline{\mathbf{tr};(\mathbf{tr};\rho)^*;\left(\mathbf{tr};\mathbf{tr};\overline{\overline{\begin{array}{c}\rho;\pi\\ \nabla\\ \rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\pi\end{array};\text{Positioned\_visitors}}+\begin{array}{c}\rho;\rho;\pi\\ \nabla\\ \rho;\rho;\rho;\rho\end{array};\text{Current\_painting}}\right)}\Bigg)}\Bigg)}\Bigg)\ ,
$$

$$
\begin{array}{l}
\vdots\\
\}\ \rangle
\end{array}
$$

FIGURE 7.14. Algebraic translation of the FOLTL view of the system presented in Figure 7.10.

4.1.2. *Algebraic counterpart of the state of the system.* Recalling the theories of Figures 7.12 and 7.14 we observe that there exist two different descriptions of the system state: the one in Figure 7.12 which is the algebraic interpretation of a propositional description of the state of the system, and the one in Figure 7.14 which corresponds to a first-order description of the state of the system.

$$\begin{tikzcd}
& T_{\mathrm{DD}} & \\
T_{\mathrm{Channel}} \arrow[ur] \arrow[d] & & T'_{\mathrm{Channel}} \arrow[ul] \arrow[d] \\
T & & T'
\end{tikzcd}$$

FIGURE 7.15. General diagram scheme used to introduce design decision.

$$\begin{tikzcd}
& T_{\mathrm{DD}} & \\
T \arrow[ur] & & T' \arrow[ul]
\end{tikzcd}$$

FIGURE 7.16. Simple diagram scheme used to introduce design decision.

$$\begin{array}{lcl}
T_{\mathsf{FOL}\leftrightarrow\mathsf{FODL}} & = & \langle\ \langle\ \{\ 1_{\mathrm{Guide}}, 1'_{\mathrm{Guide}}, 1_{\mathrm{Visitor}}, 1'_{\mathrm{Visitor}}, 1_{\mathrm{Position}}, 1'_{\mathrm{Position}}, \\
& & \qquad\qquad 1_{\mathrm{Painting}}, 1'_{\mathrm{Painting}}, 1'_{\mathsf{U}}, \\
& & \qquad\qquad \mathrm{Subscribed\_visitors}, \mathrm{Current\_painting}\ \}\ \rangle\ , \\
& & \quad \emptyset\ \rangle \\
T_{\mathsf{FOL}\leftrightarrow\mathsf{FOLTL}} & = & \langle\ \langle\ \{\ 1'_{\mathrm{Guide}}, 1'_{\mathrm{Visitor}}, 1'_{\mathrm{Position}}, 1'_{\mathrm{Painting}}, 1'_{\mathsf{U}}, \\
& & \qquad\qquad \mathrm{Tracker}, \mathrm{Subscribed\_visitors}, \mathrm{Tracked\_visitors}, \\
& & \qquad\qquad \mathrm{Paintings\_in\_museum}, \mathrm{Positioned\_visitors}, \\
& & \qquad\qquad \mathrm{Current\_painting}\ \}\ \rangle\ , \\
& & \quad \emptyset\ \rangle
\end{array}$$

FIGURE 7.17. Theories containing design decisions synchronizing T_{FOL}, T_{FODL}, and T_{FOLTL}.

$$\begin{tikzcd}
& T_{\mathsf{FOL}\leftrightarrow\mathsf{FODL}} \arrow[dl] \arrow[dr] & & T_{\mathsf{FOL}\leftrightarrow\mathsf{FOLTL}} \arrow[dl] \arrow[dr] & \\
T_{\mathsf{FODL}} & & T_{\mathsf{FOL}} & & T_{\mathsf{FOLTL}}
\end{tikzcd}$$

FIGURE 7.18. Diagram involving T_{FOL}, T_{FODL}, T_{FOLTL}, $T_{\mathsf{FOL}\leftrightarrow\mathsf{FODL}}$ and $T_{\mathsf{FOL}\leftrightarrow\mathsf{FOLTL}}$.

In §2.2 we showed how to synchronize propositional logic and first-order logic. In this case-study there is no classical propositional logic description of the system state, and the translation we used to produce the algebraic interpretation of the first-order logic partial description of the system does not produce an explicit representation of the system state. Thus, we will apply those ideas to relate the different descriptions of the system state introduced in the theories of Figures 7.12 and 7.14.

To relate these two descriptions of the system state, we shall consider the theory of Figure 7.19.

$$
\begin{aligned}
\mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}} \;=\;& \langle\, \langle\, \{\ \mathbf{St}^{\mathsf{Prop}}, \mathbf{St}^{\mathsf{FO}}, 1'_{\text{Guide}}, 1'_{\text{Visitor}}, 1'_{\text{Position}}, 1'_{\text{Painting}}, 1'_{\mathsf{U}}, {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}, \\
&\qquad \text{Subscribed\_visitors}, \text{Tracked\_visitors}\ \} \cup \\
&\qquad \{\ \text{Subscribed\_visitors}_v, \text{Tracked\_visitors}_v\ \}_{v\in\text{Visitor}}\ \rangle\ , \\
&\ \{\ \mathbf{St}^{\mathsf{Prop}} \leq 1'_{\mathsf{U}}\ , \\
&\ \ 1'_{\mathsf{U}\text{Guide}} \cdot \mathbf{St}^{\mathsf{Prop}} = 0\ , \\
&\ \ 1'_{\mathsf{U}\text{Monitor}} \cdot \mathbf{St}^{\mathsf{Prop}} = 0\ , \\
&\ \ 1'_{\mathsf{U}\text{Visitor}} \cdot \mathbf{St}^{\mathsf{Prop}} = 0\ , \\
&\ \ 1'_{\mathsf{U}\text{Position}} \cdot \mathbf{St}^{\mathsf{Prop}} = 0\ , \\
&\ \ 1'_{\mathsf{U}\text{Painting}} \cdot \mathbf{St}^{\mathsf{Prop}} = 0\ , \\
&\ \ 1'_{\mathsf{U}} = \mathbf{St}^{\mathsf{Prop}} + 1'_{\mathsf{U}\text{Guide}} + 1'_{\mathsf{U}\text{Monitor}} + 1'_{\mathsf{U}\text{Visitor}} + 1'_{\mathsf{U}\text{Position}} + 1'_{\mathsf{U}\text{Painting}}\ , \\
&\ \ {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} \leq \mathbf{St}^{\mathsf{Prop}} ; 1 ; \mathbf{St}^{\mathsf{FO}}\ , \\
&\ \ \mathit{Dom}\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right) = \mathbf{St}^{\mathsf{Prop}}\ , \\
&\ \ \mathit{Ran}\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right) = \mathbf{St}^{\mathsf{FO}}\ , \\
&\ \ {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} = {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} ; \left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right)^{\smile} ; {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\ , \\
&\ \ {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} ; \text{Subscribed\_visitors} = \Sigma_{v\in\text{Visitor}} \text{Subscribed\_visitor}_v\ , \\
&\ \ {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} ; \text{Tracked\_visitors} = \Sigma_{v\in\text{Visitor}} \text{Tracked\_visitor}_v\ \ \}\ \rangle
\end{aligned}
$$

FIGURE 7.19. Theory containing design decisions synchronizing $\mathcal{T}_{\mathsf{PDL}}$, and $\mathcal{T}_{\mathsf{FOLTL}}$.

Constant symbols $\mathbf{St}^{\mathsf{Prop}}$ and $\mathbf{St}^{\mathsf{FO}}$ were introduced to identify the two different sets of states. Being $\mathbf{St}^{\mathsf{Prop}}$ a partial identity over the set of urelements, we force a separation between those urelements representing propositional states from the ones we shall use to interpret first-order domains (also organized in disjoint subsets of $1'_{\mathsf{U}}$, see the theories of Figures 7.11, 7.13 and 7.14). Notice that the constant symbol $1'_{\mathsf{U}}$ introduced in $\mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}}$ can not longer be related to the symbol $1'_{\mathsf{U}}$ appearing in the theories of Figures 7.11, 7.13 and 7.14.

The two theories presented in Figure 7.20 will be the channels which will allow the synchronization between the theories of Figures 7.12 and 7.14 with the design decisions introduced in $\mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}}$.

$$
\begin{aligned}
\mathcal{T}^{\mathsf{PDL}}_{\text{Channel}} \;=\;& \langle\, \langle\, \{\ 1'_{\mathsf{U}}\ \} \cup \{\ \text{Subscribed\_visitors}_v, \text{Tracked\_visitors}_v\ \}_{v\in\text{Visitor}}\ \rangle\ , \\
&\ \emptyset\ \rangle \\
\mathcal{T}^{\mathsf{FOLTL}}_{\text{Channel}} \;=\;& \langle\, \langle\, \{\ 1'_{\text{Guide}}, 1'_{\text{Monitor}}, 1'_{\text{Visitor}}, 1'_{\text{Position}}, 1'_{\text{Painting}}, \mathbf{St}, \\
&\qquad \text{Subscribed\_visitors}, \text{Tracked\_visitors}\ \}\ \rangle\ , \\
&\ \emptyset\ \rangle
\end{aligned}
$$

FIGURE 7.20. Channels used to synchronize $\mathcal{T}_{\mathsf{PDL}}$, and $\mathcal{T}_{\mathsf{FOLTL}}$.

Figure 7.21 shows a diagram involving $\mathcal{T}_{\mathsf{PDL}}$, $\mathcal{T}_{\mathsf{FOLTL}}$ and $\mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}}$.

4.1.3. *Algebraic counterpart of the dynamic and temporal information.* If we consider the temporal description presented in Figure 7.10 as a complete description of the linear temporal behavior of the system, then $\mathbf{T}$ is an algebraic interpretation of the accessibility relation between all the states of the system. And, considering that all the operations were formalized as

$$\begin{tikzcd}
 & T_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}} & \\
T^{\mathsf{PDL}}_{\text{Channel}} \arrow[ur, "\sigma^{\mathsf{PDL}}"] \arrow[d] & & T^{\mathsf{FOLTL}}_{\text{Channel}} \arrow[ul, "\sigma^{\mathsf{FOLTL}}"'] \arrow[d] \\
T_{\mathsf{PDL}} & & T_{\mathsf{FOLTL}}
\end{tikzcd}$$

where:

$$\begin{array}{rcl}
\sigma^{\mathsf{PDL}} & = & \{\ 1'_{\mathsf{U}} \mapsto \mathbf{St}^{\mathsf{Prop}}\ \} \cup \{\ \text{Subscribed\_visitors}_v \mapsto \text{Subscribed\_visitors}_v\ \}_{v \in \text{Visitor}} \\
\sigma^{\mathsf{FOLTL}} & = & \{\ 1'_{\text{Guide}} \mapsto 1'_{\text{Guide}}, 1'_{\text{Monitor}} \mapsto 1'_{\text{Monitor}}, 1'_{\text{Visitor}} \mapsto 1'_{\text{Visitor}}, 1'_{\text{Position}} \mapsto 1'_{\text{Position}}, \\
 & & \quad 1'_{\text{Painting}} \mapsto 1'_{\text{Painting}}, \mathbf{St} \mapsto \mathbf{St}^{\mathsf{FOLTL}}, \\
 & & \quad \text{Subscribed\_visitors} \mapsto \text{Subscribed\_visitors}, \\
 & & \quad \text{Tracked\_visitors} \mapsto \text{Tracked\_visitors}\ \}
\end{array}$$

FIGURE 7.21. Diagram involving T_{PDL}, T_{FOLTL} and $T_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}}$.

dynamic formulas in the theories of Figures 7.8 and 7.9, we can use the approach presented in §2.1 to relate these two aspects of the system.

When trying to apply the approach presented in §2.1 two difficulties arise. The first one is that the existence of two different interpretations of the operations implies the existence of different interpretations of the system state; on one hand there is an algebraic version of the propositional interpretation of the state, and on the other hand there is an algebraic version of the first-order interpretation of the state. The relationship between these different representations of the system state will be established as in §4.1.2. The second one is that the translation used to map FODL-formulas to algebraic equations, (as the one we used to map FOL-formulas to algebraic equations), does not produce an explicit representation of the first-order state of the system. Thus, the information we have to relate is an accessibility relation between explicit first-order states ($\mathbf{T}$), atomic actions on explicit propositional states (Subscribe$_v$, Unsubscribe$_v$, BeginTracking$_v$, and EndTracking$_v$, for all $v \in$ Visitor), and atomic actions on implicit first-order states (NotifyNewPosition and UpdatePainting).

The relationship between this symbols will be established with respect to the explicit representation of the first-order states. In order to relate the representation of the accessibility relation and the representation of the operations we will use the idea presented in §2.1, but between first-order states. The representation of the operations between propositional states will be introduced in this construction by resorting to the same idea we presented in §4.1.2. For the case of the representation of the operations between first-order states, we will use the approach behind the translation from FODL-formulas to algebraic terms [**FBM02**, Definition 15]. The idea is to explicitly introduce the projections needed to apply the algebraic constants representing the FODL-atomic actions to the appropriate values of the system state, and then perform a merge between the values resulting of the application with those values that did not change.

The theory appearing in Figure 7.22 introduces these design decisions.

$$
\begin{aligned}
T_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}\leftrightarrow\mathsf{FODL}} \quad = \quad & \langle\ \langle\ \{\ \mathbf{St}^{\mathsf{Prop}}, \mathbf{St}^{\mathsf{FO}}, 1'_{\mathrm{Guide}}, 1'_{\mathrm{Visitor}}, 1'_{\mathrm{Position}}, 1'_{\mathrm{Painting}}, 1'_{\mathsf{U}}, {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}, \mathbf{T}\ \}\cup \\
& \quad \{\ \mathrm{Subscribe}_v, \mathrm{Unsubscribe}_v, \mathrm{BeginTracking}_v, \mathrm{EndTracking}_v\ \}_{v\in\mathrm{Visitor}}\cup \\
& \quad \{\ \mathrm{NotifyNewPosition}, \mathrm{UpdatePainting}\ \}\ \rangle\ , \\
& \{\ \mathbf{St}^{\mathsf{Prop}} \leq 1'_{\mathsf{U}}\ , \\
& \ 1'_{\mathsf{U}\mathrm{Guide}}\cdot\mathbf{St}^{\mathsf{Prop}} = 0\ , \\
& \ 1'_{\mathsf{U}\mathrm{Monitor}}\cdot\mathbf{St}^{\mathsf{Prop}} = 0\ , \\
& \ 1'_{\mathsf{U}\mathrm{Visitor}}\cdot\mathbf{St}^{\mathsf{Prop}} = 0\ , \\
& \ 1'_{\mathsf{U}\mathrm{Position}}\cdot\mathbf{St}^{\mathsf{Prop}} = 0\ , \\
& \ 1'_{\mathsf{U}\mathrm{Painting}}\cdot\mathbf{St}^{\mathsf{Prop}} = 0\ , \\
& \ 1'_{\mathsf{U}} = \mathbf{St}^{\mathsf{Prop}} + 1'_{\mathsf{U}\mathrm{Guide}} + 1'_{\mathsf{U}\mathrm{Monitor}} + 1'_{\mathsf{U}\mathrm{Visitor}} + 1'_{\mathsf{U}\mathrm{Position}} + 1'_{\mathsf{U}\mathrm{Painting}}\ , \\
& \ {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} \leq \mathbf{St}^{\mathsf{Prop}} ; 1 ; \mathbf{St}^{\mathsf{FO}}\ , \\
& \ Dom\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right) = \mathbf{St}^{\mathsf{Prop}}\ , \\
& \ Ran\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right) = \mathbf{St}^{\mathsf{FO}}\ , \\
& \ {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} = {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}} ; \left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right)^{\smile} ; {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\ , \\
& \ \mathbf{T} = \Big(\Sigma_{v\in\mathrm{Visitor}} \Big(\Big(\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right)^{\smile} ; \mathrm{Subscribe}_v ; {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\Big) + \\
& \qquad\qquad \Big(\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right)^{\smile} ; \mathrm{Unsubscribe}_v ; {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\Big) + \\
& \qquad\qquad \Big(\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right)^{\smile} ; \mathrm{BeginTracking}_v ; {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\Big) + \\
& \qquad\qquad \Big(\left({}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\right)^{\smile} ; \mathrm{EndTracking}_v ; {}^{\mathsf{Prop}}\mathbf{St}^{\mathsf{FO}}\Big)\Big)\Big) + \\
& \quad \left(\begin{array}{c} \left(\begin{array}{c} \pi \\ \nabla \\ \rho;\rho;\pi \\ \nabla \\ \rho;\rho;\rho;\pi \end{array}\right) ; \mathrm{NotifyNewPosition} \\ \nabla \\ \left(\begin{array}{c} \rho;\pi \\ \nabla \\ \rho;\rho;\rho;\rho \end{array}\right) \end{array}\right) ; \left(\begin{array}{c} \pi;\pi \\ \nabla \\ \rho;\pi \\ \nabla \\ \pi;\rho;\pi \\ \nabla \\ \pi;\rho;\rho \\ \nabla \\ \rho;\rho \end{array}\right) + \\
& \quad \left(\begin{array}{c} \left(\begin{array}{c} \rho;\rho;\pi \\ \nabla \\ \rho;\rho;\rho;\rho \end{array}\right) ; \mathrm{UpdatePainting} \\ \nabla \\ \left(\begin{array}{c} \pi \\ \nabla \\ \rho;\pi \\ \nabla \\ \rho;\rho;\rho;\pi \end{array}\right) \end{array}\right) ; \left(\begin{array}{c} \rho;\pi \\ \nabla \\ \rho;\rho;\pi \\ \nabla \\ \pi;\pi \\ \nabla \\ \rho;\rho;\rho \\ \nabla \\ \pi;\rho \end{array}\right)\ \}\ \rangle
\end{aligned}
$$

FIGURE 7.22. Theory containing design decisions synchronizing T_{PDL}, T_{FODL}, and T_{FOLTL}.

The theories presented in Figure 7.23 will be the channels allowing the synchronization between the theories of Figures 7.12, 7.13 and 7.14 with the design decisions introduced in $T_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}\leftrightarrow\mathsf{FODL}}$.

$$
\begin{aligned}
{T^{\mathsf{PDL}}_{\mathrm{Channel}}}' \quad &= \quad \langle\ \langle\ \{\ \mathrm{Subscribe}_v, \mathrm{Usubscribe}_v, \mathrm{BeginTracking}_v, \mathrm{EndTracking}_v\ \}_{v\in\mathrm{Visitor}} \cup \\
& \qquad\qquad \{\ 1'_{\mathsf{U}}\ \}\ \rangle\ , \\
& \qquad \emptyset\ \rangle \\
{T^{\mathsf{FOLTL}}_{\mathrm{Channel}}}' \quad &= \quad \langle\ \langle\ \{\ 1'_{\mathrm{Guide}}, 1'_{\mathrm{Monitor}}, 1'_{\mathrm{Visitor}}, 1'_{\mathrm{Position}}, 1'_{\mathrm{Painting}}, \mathbf{St}, \mathbf{T}\ \}\ \rangle\ , \\
& \qquad \emptyset\ \rangle \\
{T^{\mathsf{FODL}}_{\mathrm{Channel}}}' \quad &= \quad \langle\ \langle\ \{\ \mathrm{NotifyNewPosition}, \mathrm{UpdatePainting}\ \}\ \rangle\ , \\
& \qquad \emptyset\ \rangle
\end{aligned}
$$

FIGURE 7.23. Channels used to synchronize T_{PDL}, T_{FODL}, and T_{FOLTL}.

Figure 7.24 shows a diagram involving the theories $\mathcal{T}_{\mathsf{PDL}}$, $\mathcal{T}_{\mathsf{FOLTL}}$, $\mathcal{T}_{\mathsf{FODL}}$ and $\mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}\leftrightarrow\mathsf{FODL}}$.

$$\begin{tikzcd}
& \mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}\leftrightarrow\mathsf{FODL}} & \\
\mathcal{T}^{\mathsf{PDL}\ \prime}_{\text{Channel}} \arrow[ur, "\sigma^{\mathsf{PDL}'}"] \arrow[d] & \mathcal{T}^{\mathsf{FOLTL}\ \prime}_{\text{Channel}} \arrow[u, "\sigma^{\mathsf{FOLTL}'}"'] \arrow[d] & \mathcal{T}^{\mathsf{FODL}\ \prime}_{\text{Channel}} \arrow[ul, "\sigma^{\mathsf{FODL}'}"'] \arrow[d] \\
\mathcal{T}_{\mathsf{PDL}} & \mathcal{T}_{\mathsf{FOLTL}} & \mathcal{T}_{\mathsf{FODL}}
\end{tikzcd}$$

where:

$$\begin{aligned}
\sigma^{\mathsf{PDL}'} &= \{\ 1'_{\mathsf{U}} \mapsto \mathbf{St}^{\mathsf{Prop}}\ \} \cup \{\ \text{Subscribe}_v \mapsto \text{Subscribe}_v\ \}_{v\in\text{Visitor}} \cup \\
&\quad \{\ \text{Unsubscribe}_v \mapsto \text{Unsubscribe}_v\ \}_{v\in\text{Visitor}} \cup \\
&\quad \{\ \text{BeginTracking}_v \mapsto \text{BeginTracking}_v\ \}_{v\in\text{Visitor}} \cup \\
&\quad \{\ \text{EndTracking}_v \mapsto \text{EndTracking}_v\ \}_{v\in\text{Visitor}} \\
\sigma^{\mathsf{FODL}'} &= \{\ \text{NotifyNewPosition} \mapsto \text{NotifyNewPosition}, \\
&\quad \text{UpdatePainting} \mapsto \text{UpdatePainting}\ \} \\
\sigma^{\mathsf{FOLTL}'} &= \{\ 1'_{\text{Guide}} \mapsto 1'_{\text{Guide}}, 1'_{\text{Monitor}} \mapsto 1'_{\text{Monitor}}, 1'_{\text{Visitor}} \mapsto 1'_{\text{Visitor}}, \\
&\quad 1'_{\text{Position}} \mapsto 1'_{\text{Position}}, 1'_{\text{Painting}} \mapsto 1'_{\text{Painting}}, \mathbf{St} \mapsto \mathbf{St}^{\mathsf{FOLTL}}, \\
&\quad \mathbf{T} \mapsto \mathbf{T}\ \}
\end{aligned}$$

FIGURE 7.24. Diagram involving the theories $\mathcal{T}_{\mathsf{PDL}}$, $\mathcal{T}_{\mathsf{FOLTL}}$, $\mathcal{T}_{\mathsf{FODL}}$ and $\mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}\leftrightarrow\mathsf{FODL}}$.

Notice that the information we are adding as a design decision by introducing the theory $\mathcal{T}_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}\leftrightarrow\mathsf{FODL}}$, presented in Figure 7.22, is of great importance in the complete characterization of the system. This importance is evidenced by the partial descriptions of the operation `notifyNewPosition`. As we mentioned before, we declared its post-condition as `true` so its dynamic behavior seems to be of little interest when, in fact, is the only way the monitor is able to notify to the museum guide that a visitor has moved to a different position to observe a different art piece. Being `MuseumGuide` the only entity that can map the position on which the visitor was positioned by the monitor to the painting being exhibited at that position, it is also the only entity capable of sending the message `updatePainting` to the corresponding visitor, with the appropriate information. Now, the real characterization of the behavior of the operation `notifyNewPosition` is only obtained by the combination of its dynamic behavior (see Figure 7.9) and its interaction with the rest of the operations, provided by its temporal behavior (see Figure 7.10).

4.2. The complete description of the system. Figures 7.18, 7.21 and 7.24 show different diagrams relating different aspects of the system. In Figure 7.25 we show a complete diagram, including the theories expressing partial views of the system and the theories expressing the design decisions we discussed in §4.1.1–§4.1.3.

Then, a complete specification of the system is a co-limit of the finite diagram presented in Figure 7.25.

$$\begin{tikzcd}
 & & & T_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}\leftrightarrow\mathsf{FODL}} & & \\
 & & T^{\mathsf{PDL}}_{\text{Channel}}{}' \arrow[ur, "\sigma^{\mathsf{PDL}'}"] \arrow[d] & T^{\mathsf{FOLTL}}_{\text{Channel}}{}' \arrow[u, "\sigma^{\mathsf{FOLTL}'}"'] \arrow[ddddd] & T^{\mathsf{FODL}}_{\text{Channel}}{}' \arrow[ul, "\sigma^{\mathsf{FODL}'}"'] \arrow[d] & \\
 & T^{\mathsf{PDL}}_{\text{Channel}} \arrow[r] \arrow[dl, "\sigma^{\mathsf{PDL}}"'] & T_{\mathsf{PDL}} & & T_{\mathsf{FODL}} & \\
T_{\mathsf{PDL}\leftrightarrow\mathsf{FOLTL}} & & & & & T_{\mathsf{FOL}\leftrightarrow\mathsf{FODL}} \arrow[ul] \arrow[dl] \\
 & & & & T_{\mathsf{FOL}} & \\
 & & & & & T_{\mathsf{FOL}\leftrightarrow\mathsf{FOLTL}} \arrow[ul] \arrow[dll] \\
 & T^{\mathsf{FOLTL}}_{\text{Channel}} \arrow[uuul, "\sigma^{\mathsf{FOLTL}}"] \arrow[rr] & & T_{\mathsf{FOLTL}} & &
\end{tikzcd}$$

FIGURE 7.25. Diagram combining T_{FOL}, T_{PDL}, T_{FODL} and T_{FOLTL} with design decisions.

CHAPTER 8

Tool support

Nowadays, the success of a specification language in industrial projects does not only relay in its clarity, compactness and expressiveness but also on the existence of tool support both for the design and validation/verification process. In this chapter we present a verification tool for fork algebras based on the semi-automatic theorem-prover *PVS*. This is work carried on by the author when he visited SRI International in 2002. This joint research with Owre and Shankar was published in [**LOS02**].

We also present ReMo, a validation tool for fork algebraic specifications. While the results reported were not obtained by the author, it is worth including this description for two reasons: (a) completeness of the report on tool support, and (b) ReMo is one of the research lines to be pursued by the author as part of his post-doctoral research.

1. Theorem-proving fork algebraic specifications in *PVS*

1.1. *PVS* (Prototype Verification System). Let us first review some of the main capabilities of this theorem prover built at the Computer Science Laboratory of SRI International.

PVS (Prototype Verification System) is intended as an environment for constructing clear and precise specifications, and for developing readable proofs that have been mechanically verified [**ORS92, ORSv95, Sha01**]. A variety of examples have been verified using *PVS* [**CLM$^+$95, ORSSC98**]. The most substantial use of *PVS* has been in the verification of the microcode for selected instructions of a commercial-scale microprocessor called AAMP5 designed by Rockwell-Collins [**MS95**] and in the LOOP project [**vJ01**]. The key elements of the *PVS* design are captured by the combination of features listed below.

1.1.1. *An expressive language with powerful deductive capabilities.* The *PVS* specification language is based on classical, simply typed, higher-order logic with base types such as the Booleans `bool` and the natural numbers `nat`, and type constructors for functions `[A -> B]`, records `[# a : A, b : B #]`, and tuples `[A, B, C]`. The *PVS* type system also admits predicate subtypes, e.g., {`i : nat | i > 0`} is the subtype of positive numbers. The *PVS* type system includes dependent function, record, and tuple types, e.g., `[# size: nat, elems: [below[size] -> nat] #]` is a dependent record where the type of the `elems` component depends on the value of the `size` component. It is also possible to define recursive abstract datatypes such as lists and trees. The definition of a recursive datatype can be illustrated with the list type of the *PVS* prelude built from the constructors `cons` and `null`. Theories containing the relevant axioms, induction schemes, and useful datatype operations are generated from the datatype declaration.

```
list [T: TYPE]: DATATYPE
 BEGIN
  null: null?
  cons (car: T, cdr:list):cons?
 END list
```

PVS also has parametric theories, so that it is possible to capture, say, the notion of sorting with respect to arbitrary array sizes, types, and ordering relations. Constraints on the theory parameters can be stated by means of assumptions within the theory. When an instance of a theory is imported with concrete parameters, there are proof obligations for the corresponding instances of the parameter assumptions. A theory is a list of declarations of constants (with or without definitions) and theorems. The *PVS* typechecker checks a theory for simple type correctness and generates proof obligations (called TCCs for *type correctness conditions*) corresponding to predicate subtypes. Typechecking is undecidable for *PVS*. Therefore it requires discharging such proof obligations.

1.1.2. *Powerful decision procedures with user interaction. PVS* proofs are constructed interactively. The primitive inference steps for constructing proofs are quite powerful. They make extensive use of efficient decision procedures for equality and linear arithmetic [**Sho84, RS01, SR02**]. They also exploit the tight integration between rewriting, the decision procedures, and the use of type information [**CRSS94**]. *PVS* also uses BDD-based propositional simplification so that it can combine the capability of simplifying very large propositional expressions with equality, arithmetic, induction, and rewriting.

Higher-level inference steps can be defined by means of strategies (akin to LCF tactics [**GMW79**]) written in a simple strategy language. Typical strategies include heuristic instantiation of quantifiers, propositional and arithmetic simplification, and induction and rewriting. The *PVS* proof checker tries to strike a careful balance between an automatic theorem prover and a low-level proof checker.

1.1.3. *Model checking with theorem-proving.* Many forms of finite-state verification, such as linear temporal logic model checking, language containment, and bisimulation checking, can be expressed in the μ-calculus [**BC$^+$90, EL86**]. The higher-order logic of *PVS* is used to define the least and greatest fixpoint operators of the μ-calculus. When the state type is finite, the μ-calculus expressions are translated into the propositional μ-calculus and a propositional μ-calculus model checker can be used as a decision procedure. The finite types are those built from booleans and scalars using records, tuples, or arrays over sub-ranges. Fairness cannot be expressed in CTL, but it can be defined using the μ-calculus. BDD-based symbolic model checking is integrated into *PVS* as a decision procedure for the Boolean fragment of the μ-calculus [**RSS95**]. The model checker can be invoked as an interactive proof step together with rewriting, induction, and simplification using the ground decision procedures. Automatic predicate abstraction has been implemented as an interactive step for constructing finite-state property-preserving abstractions of infinite-state systems [**SS99**]. Though this exercise does not use the model checker and abstractor, these will play an important rôle in future work.

1.1.4. *Deduction and Execution.* A functional fragment of *PVS* has been given an execution semantics that is supported by a code generator which produces Common Lisp code. The code generator includes a destructive update optimization that translates *PVS* array updates into destructive updates in Common Lisp, when it is safe to do so [**Sha02**]. The generated code is also safe with respect to runtime errors if it is generated from a typechecked *PVS* expression where all the generated proof obligations have been discharged.

1.2. A semantic embedding of proper fork algebras in *PVS*. In [**LOS02**], a joint work with SRI International, we presented a proof checker for $\mathbf{A_g}$ [**FBL01, FLB02**] using *PVS* as a semantic framework. $\mathbf{A_g}$ is a superset of the language of fork algebras with urelements including first-order dynamic logic formulas over a domain which is a pair-dense proper fork algebra with urelements. From that we derived a semantic framework for ω-closure fork algebras with urelements. In this section we present the highlights of this theorem-proving environment.

The construction of a theorem-proving environment for fork algebras requires the encoding of the semantics of its language within the higher-order logic of *PVS*. To accomplish this task we made use of some useful and powerful features such as the abstract data-type mechanism [**OS93**].

The first step is the construction of the ω-closure fork algebras with urelements language objects in a way that allows us to define their semantics as we did in Definitions 2.50 and 2.51. Next it requires the encoding of the full proper closure fork algebra with urelements which, as we proved in Theorem 2.15, are the models of ω-CCFAU.

The syntax of ω-closure fork algebras with urelements was constructed by providing an abstract datatype whose elements are fork algebraic terms and formulas. Figures 8.1 and 8.2 show the interesting parts of the *PVS* files defining the language of ω-closure fork algebras with urelements. In Figure 8.1 we observe the basic definitions implementing fork algebraic signatures (see Definition 2.49). Notice that we extended the notion of signature allowing the introduction of new predicate symbols. In Figure 8.2 we show the definition of the language syntactical objects by means of an abstract datatype.

When the specification is typechecked the abstract datatype mechanism generates a new specification file. This new specification contains the theory of the objects defined by the datatype, including the subtypes, map function, and the recursion combinator `reduce_nat`. The recursion combinator will

```
CFAU_Signature: THEORY

  BEGIN

    Constant: TYPE = {zero, one, one_prime, pi, rho, ...}
    ⋮
    Predicate: TYPE = {Leq, Functional, OneToOne, Pair, ...}

    arityPredicate: [Predicate -> nat] =
            LAMBDA (P: Predicate): CASES P
                                    OF Leq: 2,
                                       Functional: 1,
                                       OneToOne: 1,
                                       Pair: 1
                                     ENDCASES

    Function_: TYPE = {join, meet, complement, composition,
                       converse, fork, closure, choice, ...}

    arityFunction_: [Function_ -> nat] =
            LAMBDA (F: Function_): CASES F
                                    OF join: 2,
                                       meet: 2,
                                       complement: 1,
                                       composition: 2,
                                       converse: 1,
                                       fork: 2,
closure: 1,
choice: 1,
                                       ⋮
                                    ENDCASES

  END CFAU_Signature
```

Figure 8.1. Definition of ω-closure fork algebra signatures within *PVS*.

be particularly useful in the encoding of the semantics because it will be the basis for the complexity measure that we need to define the meaning function recursively.

The definition of the language is parametric in the constant, variable, predicate and function symbols, and also in the functions that define the arity of the predicate and function symbols. This allows us to work with only one formal language, but using different instances of it depending on the sets of symbols used in particular problems.

In Figure 8.3 we show the important parts of the theory that define the meaning function for ω-closure fork algebras with urelements.

The parameters `mConstant`, `mPredicate`, and `mFunction_` are functions that map every constant, predicate and function symbol to an object, predicate and function over the carrier of the algebra.

In Figure 8.4 we show the meaning function for terms. The most interesting part of this function is the use of the map function of lists to compute the meaning of a function symbol application to a list of terms.

```
CFAU_Language[Constant: TYPE,
              Variable: TYPE,
              Predicate: TYPE, arityPredicate: [Predicate -> nat],
              Function_: TYPE, arityFunction_: [Function_ -> nat]]:
        DATATYPE WITH SUBTYPES Term_, Formula_

  BEGIN

    c(c: Constant): c?: Term_
    v(v: Variable): v?: Term_
    F(f: Function_, lF:
      lPrime: list[Term_] | arityFunction_(f) = length(lPrime)):
           F?: Term_
    =(t1, t2: Term_): Eq?: Formula_
    P(p: Predicate, lP:
      lPrime: list[Term_] | arityPredicate(p) = length(lPrime)):
           P?: Formula_

  END CFAU_Language
```

FIGURE 8.2. Definition of ω-closure fork algebra signatures within *PVS*.

```
CFAU_semantic[Constant: TYPE,
              Variable: TYPE,
              Predicate: TYPE, arityPredicate: [Predicate -> nat],
              Function_: TYPE, arityFunction_: [Function_ -> nat],
              Carrier: TYPE+,
              mConstant: [Constant -> Carrier],
              mPredicate: [P: Predicate ->
                             [{l: list[Carrier] | arityPredicate(P) = length(l)} ->
                                bool]],
              mFunction_: [F: Function_ ->
                             [{l: list[Carrier] | arityFunction_(F) = length(l)} ->
                                Carrier]]]: THEORY

  BEGIN

    IMPORTING CFAU_Language[Constant,
                            Variable,
                            Predicate, arityPredicate,
                            Function_, arityFunction_],
                 list_max

    Valuation: TYPE = [v: Variable -> Carrier]
    ⋮
    mTerm(val: Valuation)(t: Term_):
        RECURSIVE Carrier = ...

    m(f: Formula_): RECURSIVE PRED[Valuation] = ...

  END CFAU_semantic
```

FIGURE 8.3. Theory encoding the meaning function for ω-closure fork algebras with urelements.

```
mTerm(val: Valuation)(t: Term_):
    RECURSIVE Carrier =
  CASES t
    OF c(c_var): mConstant(c_var),
            v(v_var): w(v_var),
            F(f_var, list_var):
                 mFunction_(f_var)(map(mTerm(val))(list_var))
    ENDCASES
  MEASURE complexity(t)
```

FIGURE 8.4. Meaning function for terms.

Notice that until now we only provided the mechanism assigning semantics to a formula which is of no help without the algebra on which terms must be interpreted.

In order to construct the algebra on which we will interpret the formulas, we must first construct the structured universe on which binary relations will range. The definition of this structured universe, appearing in Figure 8.5, is given by resorting to an abstract datatype

```
CFAU_Element[T: TYPE]: DATATYPE
  BEGIN
    urelement(t: T): urelement?
    star(el0, el1: CFAU_Element): star?
  END CFAU_Element
```

FIGURE 8.5. Definition of the structured universe within *PVS*.

Recall that the abstract datatype mechanism generates inductive types, so the universe is finitely generated. This restriction means that this construction only enables to represent a subclass of the full proper closure fork algebras. Fixing this requires removing all inductive declarations from the generated theories, which would lead to the theory presented in Figure 8.6.[1]

In Figures 8.7 and 8.8 we show how full proper closure fork algebra terms are defined using functions (`mConstant`, `mPredicate` and `mFunction_`) that map constant, predicate and function symbols to objects, predicates and functions defined over binary relations. In Figure 8.7 we show how the theory that represents the structured universe is imported to build the carrier of the algebra.

Finally in Figure 8.8 we show how this carrier is used to give semantics to the symbols declared in Figure 8.1. The most interesting part of this file is the use of the μ-calculus, defined in PVS's prelude file, to construct the least fix point of a binary relation with respect to the composition operation in order to give the semantics of a closure.

[1]The generated file has advantages aside from providing induction; in particular, the inclusive and disjoint axioms are automatically discharged by the prover. For expediency the actual specifications include the generated files, and we were careful not to make use of induction. We plan to correct this for the next application of this semantics.

```
CFAU_Element_adt[T: TYPE]: THEORY
 BEGIN

  CFAU_Element: TYPE

  urelement?, star?: [CFAU_Element -> boolean]

  urelement: [T -> (urelement?)]

  star: [[CFAU_Element, CFAU_Element] -> (star?)]

  t: [(urelement?) -> T]

  el0: [(star?) -> CFAU_Element]

  el1: [(star?) -> CFAU_Element]

  ord(x: CFAU_Element): upto(1) = ...

  CFAU_Element_urelement_extensionality: AXIOM ...
  CFAU_Element_urelement_eta: AXIOM ...
  CFAU_Element_star_extensionality: AXIOM ...
  CFAU_Element_star_eta: AXIOM ...
  CFAU_Element_t_urelement: AXIOM ...
  CFAU_Element_el0_star: AXIOM ...
  CFAU_Element_el1_star: AXIOM ...
  CFAU_Element_inclusive: AXIOM ...
  CFAU_Element_disjoint: AXIOM ...

 END CFAU_Element_adt
```

FIGURE 8.6. Theory resulting of the *PVS* abstract datatype mechanism on the definition of the structured universes.

```
CFAU_semantic: THEORY

  BEGIN

    IMPORTING CFAU_Language

    Element: TYPE+

    IMPORTING CFAU_Element_adt[Element]
    Carrier: TYPE FROM PRED[[CFAU_Element, CFAU_Element]]

    carrier_full: AXIOM
       FORALL (r: PRED[CFAU_Element, CFAU_Element]):
          Carrier_pred(r)

  END CFAU_semantic
```

FIGURE 8.7. Construction of the carrier of the full proper closure fork algebra with urelements.

The definitions presented in previous sections suffice to build a framework in which it is possible to prove properties written in ω-closure fork algebras with urelements. Notice that the semantics presented so far just

```
CFAU_semantic: THEORY

  BEGIN
    .
    .
    .
    zero: Carrier = LAMBDA (wp: [CFAU_Element, CFAU_Element]): FALSE
    one: Carrier = LAMBDA (wp: [CFAU_Element, CFAU_Element]): TRUE
    .
    .
    .
    join(c0, c1: Carrier): Carrier = LAMBDA (wp: [CFAU_Element, CFAU_Element]): c0(wp) OR c1(wp)
    .
    .
    .
    fork(c0, c1: Carrier): Carrier = LAMBDA (wp: [CFAU_Element, (star?)]):
                                   c0((wp`1, el0(wp`2))) AND c1((wp`1, el1(wp`2)))
    closure(c0: Carrier): Carrier =
       LAMBDA (wp: [CFAU_Element, CFAU_Element]):
           mu[[CFAU_Element, CFAU_Element]] (LAMBDA (r: PRED[[CFAU_Element, CFAU_Element]]):
               {p: [CFAU_Element, CFAU_Element] | one_prime(p) OR composition(c0, r)(p)})

    .
    .
    .
    Leq(c0, c1: Carrier): bool = sum(c0, c1) = c1
    Functional(c: Carrier): bool = Leq(composition(converse(c), c), one_prime)
    .
    .
    .
    mConstant: [Constant -> Carrier] =
      LAMBDA (c: Constant): CASES c
                          OF zero: zero,
                                   one: one,
                             .
                             .
                             .
                          ENDCASES

    mPredicate: [P: Predicate -> [{l: list[Carrier] | arityPredicate(P) = length(l)} -> bool]] =
      LAMBDA (P: Predicate): CASES P
                          OF Leq:
                                LAMBDA (l: {lPrime: list[Carrier] | length(lPrime) = 2}):
                                  Leq(nth(l, 0), nth(l, 1)),
                              Functional:
                                LAMBDA (l: {lPrime: list[Carrier] | length(lPrime) = 1}):
                                  Functional(nth(l, 0)),
                              .
                              .
                              .
                            ENDCASES

    mFunction_: [F: Function_ -> [{l: list[Carrier] | arityFunction_(F) = length(l)} ->
                                   Carrier]] =
      LAMBDA (F: Function_): CASES F
                           OF join:
                                LAMBDA (l: {lPrime: list[Carrier] | length(lPrime) = 2}):
                                  join(nth(l, 0), nth(l, 1)),
                              .
                              .
                              .
                              fork:
                                LAMBDA (l: {lPrime: list[Carrier] | length(lPrime) = 2}):
                                  fork(nth(l, 0), nth(l, 1)),

                              closure:
                                LAMBDA (l: {lPrime: list[Carrier] | length(lPrime) = 1}):
                                  closure(nth(l, 0))
                              .
                              .
                              .
                           ENDCASES

  END CFAU_semantic
```

FIGURE 8.8. Usage of the structured universe to give semantics to the ω-closure fork algebra with urelements constants, predicates and functions.

uses standard PVS features and provides the means to prove every property written in this language.

2. ReMo: a model-checker for fork algebraic specifications

ReMo was presented in [**GS05, FGSB05**]. It is a tool that aims at providing automatic analysis of ω-closure fork algebra specifications through the use of model-checking techniques. It is easy to see that the language of ω-closure fork algebra with urelements is not decidable, thus imposing limitations on the conclusions we can draw from the analysis process. In the following section we present Rel, the language implemented in ReMo, which is an extension of the language of ω-closure fork algebra with urelements; and ReMo.

2.1. The Rel modeling language. In this section we introduce Rel, a purely relational specification language. Rel's syntax and semantics are introduced in Figure 8.9.

$problem$ ::= decl*form	M : form $\rightarrow env \rightarrow Boolean$
decl ::= var : $typexpr$	X : expr $\rightarrow env \rightarrow value$
$typexpr$ ::=	$\mid env = (var + type) \rightarrow value$
$\mid type \times type$	$\mid value = atom +$
$\mid typexpr \times type$	$\mid atom \times value +$
$\mid type \times typexpr$	$\mid value \times atom$
form ::=	
expr <= expr (subset)	$M[a \ \texttt{<=} \ b]e = X[a]e \subseteq X[b]e$
| !form (neg)	$M[!F]e = \neg M[F]e$
| form && form (conj)	$M[F\&\&G]e = M[F]e \wedge M[G]e$
| form || form (disj)	$M[F \mid\mid G]e = M[F]e \vee M[G]e$
expr ::=	
0_t (empty with type t)	$X[0_t] = \emptyset$
| 1_t (universal with type t)	$X[1_t] =$ largest relation of type t
| id_t (identity with type t)	$X[id_t] =$ the identity relation of type t
| expr + expr (union)	$X[a + b]e = X[a]e \cup X[b]e$
| expr & expr (intersection)	$X[a\&b]e = X[a]e \cap X[b]e$
| [expr, expr] (fork)	$X[\ [a, b]\ ]e = \{\ \langle x, \langle y, z\rangle\rangle \mid \langle x, y\rangle \in X[a]e \wedge \langle x, z\rangle \in X[b]e\ \}$
| −expr (complement)	$X[-a] = X[1_t]e \setminus X[a]e$
|∼ expr(transpose)	$X[\sim a]e = \{\ \langle x, y\rangle \mid \langle y, x\rangle \in X[a]e\ \}$
| expr · expr (navigation)	$X[a \cdot b]e = X[a]e;X[b]e$
| +expr (transitive closure)	$X[+a]e =$ the smallest r such that $r;r \subseteq r$ and $X[a]e \subseteq r$
| Var (variable)	$X[v]e = e(v)$

FIGURE 8.9. Grammar and semantics of Rel.

Notice that Rel formulas are Boolean combinations of equations. In the forthcoming sections, we will need the following result proved in [**TG87**, p. 26].

THEOREM 8.1. *(*[**FGSB05**, Theorem 2.1]*)*
Every Rel *formula is equivalent to a* Rel *formula of the form $T = 1$, for an appropriate term T.*

The proof of Theorem 8.1 uses the following procedure in order to reduce Boolean combinations of equations to a single equation:

$$\begin{array}{lcl} T_1 <= T_2 & \rightsquigarrow & (-T_1) + T_2 = 1 \\ T_1 = T_2 & \rightsquigarrow & T_1 \& T_2 \ + \ (-T_1) \,\&\, (-T_2) = 1 \\ !(T = 1) & \rightsquigarrow & 1 \cdot (-T) \cdot 1 = 1 \\ T_1 = 1 \ \wedge \ T_2 = 1 & \rightsquigarrow & T_1 \& T_2 = 1 \ . \end{array}$$

3. Monotonicity Analysis for Verification of Relational Specifications

Let us consider a relational specification *Spec* in a suitable relational language (as for instance Rel). In order to validate this specification, we want to automatically analyze whether a given property α follows from *Spec*. Expressive enough relational languages are undecidable. Thus, in order to make automatic analysis feasible we will impose bounds on the size of domains. The analysis procedure reduces then to finding instances (concrete relations among elements from the bounded domains) for the relational variables that satisfy *Spec*, yet falsify α.

Notice that given a family of domains $D_1, \ldots, D_k$ with bounds $b_1, \ldots, b_k$, respectively, and a relational variable R, there are $2^{b_1 \times \cdots \times b_k}$ possible values (relations) for R on $D_1 \times \cdots \times D_k$. Even for small values of $b_1, \ldots, b_k$, exhaustive search of appropriate instances is in general unfeasible (even more if we consider that the previous number scales up exponentially when more relational variables $R_1, \ldots, R_k$ are considered). Therefore, strategies that allow us to prune the state space are mandatory in order to make automatic analysis feasible. Some strategies are *general*, in the sense that are either specification independent, or in general improve analysis performance. An example of such technique is isomorphisms elimination [**JJD98**]. Other strategies, as the one presented in [**FGSB05**], are specification dependent.

Definition 8.1. Given a relational variable R and a term $t(R)$, we say that R is *positive* (in t) if all the occurrences of R lay under an even number of complements. It is *negative* (in t) if all the occurrences of R lay under an odd number of complements. If R is neither positive nor negative in t, it is then said to be *undefined* in t.

As an example, let us consider the terms

$$-((-R) \ \cdot \ S)\,, \tag{49}$$

$$-((\sim R) \ \cdot \ R)\,, \tag{50}$$

$$(R \cdot R) \ \& \ (-R) \ . \tag{51}$$

In term (49), variable R is positive, while variable S is negative. In (50), R is negative. Finally, in (51), R is undefined.

Definition 8.2. Given a relational variable R and a term $t(R)$, t is isotonic with respect to R if for all concrete relations r, s, $r \subseteq s \ \Rightarrow \ t(r) \subseteq t(s)$. Similarly, t is antitonic with respect to R if for all concrete relations r, s, $r \subseteq s \ \Rightarrow \ t(r) \supseteq t(s)$.

Lemma 8.1. *(*[**FGSB05**, Proposition 3.3]*)*
Let $t(R)$ be a relational term on the variable R. If R is positive in t, then t is isotonic with respect to R. Similarly, if R is negative in t, then t is antitonic with respect to R. ■

In order to introduce our strategy, we will assume that the specification *Spec* consists of a sequence of formulas on a single variable R. We will later drop this assumption and generalize to variables $R_1, \ldots, R_n$. Moreover, we will assume that formulas are equations of the form $T = 1$. Notice that from Theorem 8.1, there is no loss of generality in adopting this assumption. We will also assume that R is a binary relation. Since fork allows us to simulate relations of arity greater than 2 [**FLB$^+$05**], there is no loss of generality in this assumption either.

The set of all relations on the domain $A \times B$, ordered by inclusion, is a lattice. Since in the worst case it will be necessary (according to our strategy) to explore the whole lattice, it is essential to explore it in a way such that each relation is visited (at most) once.

We will traverse the lattice in a depth-first search (DFS) manner. Actually, we will present two DFS traversals of the lattice. One from the bottom, and one from the top. If A contains elements $a_1, \ldots, a_n$ and B contains elements $b_1, \ldots, b_m$, we can impose on A and B the total orderings $a_1 < \cdots < a_n$ and $b_1 < \cdots < b_m$. Then, the relation on $A \times B$ defined by

$$\langle a_1, b_1 \rangle < \langle a_2, b_2 \rangle \quad \Longleftrightarrow \quad a_1 < a_2 \vee (a_1 = a_2 \ \wedge \ b_1 < b_2)$$

is a total ordering, called the *lexicographic* ordering. For the traversal from the bottom, notice that any given relation R has, as immediate *successors*, relations of the form $R \cup \{ \langle a, b \rangle \}$ ($a \in A$, $b \in B$), where for every $\langle a', b' \rangle \in R$, $\langle a, b \rangle > \langle a', b' \rangle$. If we soften the last requirement and just require immediate successors to add a new pair, the following situation is possible:

$$\{ \langle a, b \rangle \} \subseteq \{ \langle a, b \rangle , \langle c, d \rangle \} \supseteq \{ \langle c, d \rangle \} ,$$

i.e., there are two different relations ($\{ \langle a, b \rangle \}$ and $\{ \langle c, d \rangle \}$) that have a common successor, which might be visited twice.

Once the successors of a given a relation are defined, if we are given two different successors of R, namely $R \cup \{ \langle a, b \rangle \}$ and $R \cup \{ \langle c, d \rangle \}$, an ordering between them is induced by the ordering between $\langle a, b \rangle$ and $\langle c, d \rangle$. Therefore, in order to traverse the lattice, the successors of R will be visited according to this ordering. Figure 8.10 shows an example. Each matrix represents a relation contained in the set $\{ 0, 1 \} \times \{ 0, 1 \}$. A dark square in position $\langle i, j \rangle$ means pair $\langle i, j \rangle$ belongs to the relation modeled by the matrix. The number attached to each matrix gives the traversal ordering.

In order to traverse the lattice in a descending order, we define the *predecessors* of a relation R as the set

$$\{ -P \mid P \text{ is a successor of } -R \} \ .$$

Notice that since P is a successor of $-R$, $-R \subseteq P$, and therefore, $-P \subseteq R$. Also, predecessors differ from the parent relation in that the latter has one extra pair. The ordering in which relations are visited in the descending traversal follows from the ordering in the ascending traversal. Figure 8.11 shows an example of a descending traversal.

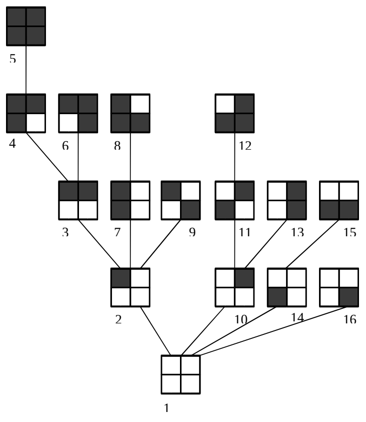


FIGURE 8.10. An ascending traversal.

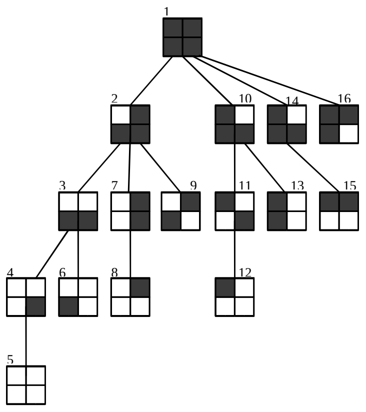


FIGURE 8.11. A descending traversal.

Let us consider now an equation of the form $t(R) = 1$ in which variable R is negative in t. As a running example, consider the following equations stating that R is a total (cf. (53)) functional (cf. (52)) relation. Notice that R is negative in (52).

$$-((\sim R) \cdot R) + Id = 1 \tag{52}$$

$$R \cdot 1 = 1 \tag{53}$$

Since we want to satisfy an equation of the form $t(R) = 1$, we want to maximize the value of $t(R)$ (notice that we are strongly using the assumption on the shape of the equation). Since R is negative in t, $t(R)$ reaches a maximum when $R = 0$. Notice that in the example, while (52) is satisfied, (53) is not. Therefore, it is necessary to search for another model. It is clear at this point that values of R near the bottom of the lattice are more likely to satisfy (52).

LEMMA 8.2. *(*[**FGSB05**, Proposition 3.4]*)*
Let $t(R) = 1$ be an equation on the variable R. Assume R is negative in t. Let r be a concrete relation such that:

1. *$t(r) = 1$,*
2. *for a successor r' of r, $t(r') \neq 1$.*

Then, for every relation $x \supseteq r'$, $t(x) \neq 1$. ■

Lemma 8.2 provides us with a sufficient criterion for pruning part of the lattice. If in an ascending traversal of the lattice we reach a relation r' for which $t(r') \neq 1$, the branch with origin in r' does not need to be traversed because it cannot produce a model.

Thus, we can conclude that:

1. if we are given an equation of the form $t(R) = 1$,
2. variable R is negative in t,
3. we are performing an ascending traversal of the lattice,
4. we have reached a relation r' in the lattice for which $t(r') \neq 1$,

then the branch of the lattice with origin in r' can be pruned. See Figure 8.12 for a graphical description.

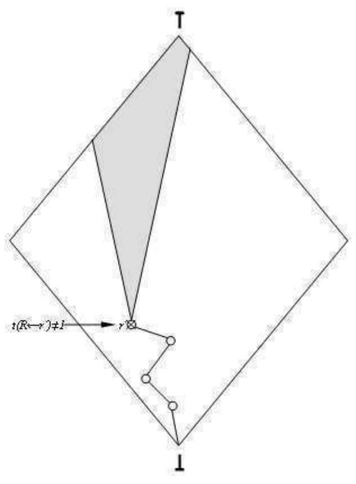


FIGURE 8.12. Using monotonicity information for negative variables.

A proof similar to that of Lemma 8.2 allows us to prove the following proposition.

LEMMA 8.3. *(*[**FGSB05**, Proposition 3.5]*)*
Let $t(R) = 1$ be an equation on the variable R. Assume R is positive in t. Let r be a concrete relation such that:

1. *$t(r) = 1$,*
2. *for a predecessor r' of r, $t(r') \neq 1$.*

Then, for every relation $x \subseteq r'$, $t(x) \neq 1$. ■

At this point we can also conclude that:

1. if we are given an equation of the form $t(R) = 1$,
2. variable R is positive in t,

(3) we are performing a descending traversal of the lattice,
(4) we have reached a relation r' in the lattice for which $t(r') \neq 1$,

then the branch of the lattice with origin in r' can be pruned. See Figure 8.13 for a graphical description.

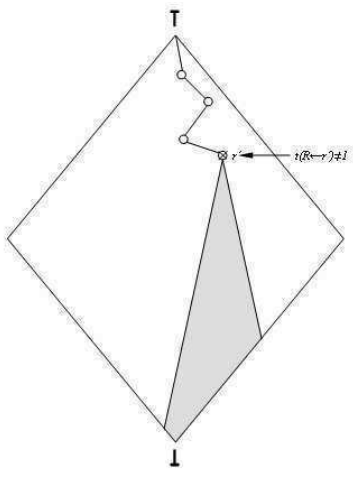


FIGURE 8.13. Using monotonicity information for positive variables.

Let us analyze now how general or restrictive are the hypothesis we are assuming. Notice that so far we have only discussed the situation where a single equation is being analyzed. If we are given equations $E_1, \ldots, E_k$, from Theorem 8.1 we can assume they are all of the form $T_i(R) = 1$ $(1 \leq i \leq k)$. At this point we are still considering the case in which there is a single relational variable R. In each equation, R may be positive, negative or undefined. Let us assume, without loss of generality, that there are more equations in which R is negative. Then, an ascending traversal of the lattice will allow us to prune a branch when one of the negative equations fails. Notice that the traversal ordering is chosen upon establishing what is the prevailing monotonicity. Therefore, the only real assumption we are making, is that variable R has a defined monotonicity in some of the equations. Thus, this is the context in which our pruning strategy can be applied.

Let us remove now the remaining assumption, namely, the restriction to a single variable R. Let us consider now relational variables $R_1, \ldots, R_n$; and equations $E_1, \ldots, E_k$, which, from Theorem 8.1, we can assume are all of the form $T_i(R_1, \ldots, R_n) = 1$ $(1 \leq i \leq k)$. We compute for each variable R_i $(1 \leq i \leq n)$ the amount of equations in which it is positive or negative, and call R_i positive (negative) if it appears positive (negative) in more equations. We now define for each variable a traversal ordering of the lattice as follows: if R_i is positive then the lattice is traversed from the top, and if it is negative the lattice is traversed from the bottom. Under these conditions we can prove the following theorem.

THEOREM 8.2. *(*[**FGSB05**, Theorem 3.6]*)*
Let $T_i(R_1, \ldots, R_n) = 1$ $(1 \leq i \leq k)$ be an equation from Spec. Let the sign of each variable in T agree with the sign in the specification (that is, if R_i

is positive (negative) in more equations, then it is also positive (negative) in T). If

(1) $r_1, \ldots, r_n$ *are concrete relations such that*

$$T_i(r_1, \ldots, r_n) \neq 1,$$

and

(2) $r'_1, \ldots, r'_n$ *are concrete relations such that* $r'_j \supseteq r_j$ ($r'_j \subseteq r_j$) *if* R_j *is negative (positive), then*

$T_i(r'_1, \ldots r'_n) \neq 1.$ ■

Theorem 8.2 allows us to prune the lattice as soon as a configuration as the one described in the hypotheses is reached. In §4 we present ReMo, a tool implementing this strategy, and evaluate its performance.

4. ReMo: RElational verification through MOnotonicity analysis

ReMo is an application that implements the analysis strategy described in §3. The structure of the relational specifications that ReMo analyzes is shown in Figure 8.14.

```
\domains
   D1 [m1:n1]
   :
   Dk [mk:nk]

\constants
   C1 < D1*D2
   :
   Cr < Dk*D1

\identities
   Id1 D1
   :
   Ids Dk

\empties
   Zero1 D1*D2
   :
   Zerot Dk*Dk

\universals
   Unit1 D1*D1
   :
   Unitu Dk*D3

\axioms
   Formula1
   :
   Formulai

\properties
   Formula1
   :
   Formulaj
```

FIGURE 8.14. Structure of a ReMo Specification.

After the `\domains` keyword, we list the domains in the specification, as well as a range (lower and upper bound) for their size. After the `\constants` keyword, we list the relational variables in the specification, as well as their type (in Figure 8.14 all the relational variables are to be interpreted as binary relations on the corresponding domains). Under the `\identities` keyword, we list those identity relations that will be required in the specification, together with their type. Similarly, we declare empty and universal relations under the appropriate keywords. Finally, the specification contains the axioms and the assertions to be verified.

ReMo receives a specification as input and transforms, using the translation defined by Theorem 8.1, each axiom and assertion to an equation of the form $T = 1$. It then computes the monotonicity of each relational variable, and determines a traversal order for each one. Values for the variables are then generated for the variables according to the traversal order, and the pruning strategy is applied whenever possible. ReMo returns counterexamples in different modalities:

- the *first* counterexample,
- the *first* k counterexamples, or
- the *next* counterexample.

Since ReMo deals with binary relations, these were implemented using *Reduced Ordered Binary Decision Diagrams* (ROBDDs) [**SW93**].

5. Concluding remarks

ReMo[**GS05, FGSB05**] is a tool which allows one to analyze ω-closure fork algebra specifications in a fully automatic way. Considering the undecidability of the language of ω-closure fork algebras with urelements, accomplishing this level of automatization requires bounding the size of the domains on which binary relations range. Bounding the size of domains has a direct impact on the conclusions we can draw from the analysis process. If a counterexample for a given assertion is found, then the model is for sure flawed. On the other hand, if no counterexample is found, we can only conclude that no counterexamples exist when domain sizes are constrained to the given bounds. Choosing larger bounds may show the existence of previously unforeseen errors. This limited analyzability offered by ReMo is essential in order to analyze ω-closure fork algebra specifications and get rid of most errors introduced in the modeling process. At the same time, models for critical applications can also benefit from usage of ReMo, but one cannot entirely rely on that.

An alternative is the use of semi-automatic theorem provers, and among these, *PVS* [**ORS92**]. Theorem provers have limitations too. First, they require an expertise from the user that many times discourages their use. And second, minor errors in a model may require to redo proofs that were using wrong hypotheses. Much the same as errors overlooked during software requirement elicitation have a greater impact the more advanced the development stage, model errors have greater impact the more auxiliary lemmas have been proved. Therefore, getting rid of as many errors as possible from the model before starting the theorem-proving process is a must.

The limits of analysis using ReMo is shared by all the tools in the field of lightweight formal methods. In [**FLM07**] we discussed how the use of the *Alloy Analyzer* [**Jac02**], one of the most prominent tools in this field, can be complemented by the use of the theorem prover *PVS* in order to conclude the absence of errors in a model. This marriage is carried out by allowing the user of *PVS* to validate the hypothesis by calling the execution of the *Alloy Analyzer* to search for finite models violating them. Also in [**FLM06, FLM07**], we presented the tool *Dynamite* which implements this interaction between *Alloy* and *PVS*.

The same improvement of the proving process can be implemented between the automatic validation technique implemented in ReMo and the semi-automatic verification capability provided by the semantic embedding of ω-closure fork algebra with urelements in *PVS*. This interaction between ReMo and the semantic embedding of ω-closure fork algebra with urelements in *PVS* has not been implemented yet, but in the presence of the experience we gained during the development of *Dynamite*, we believe it can easily be done.

CHAPTER 9

Concluding remarks and further work

1. Concluding remarks

The existing experience on building algebraizations of logics in fork algebras shows that reasonable extensions of fork algebras inherit the property of being representable on a more "concrete" class of algebras (i.e. extensions of the full proper fork algebras); even in the presence of complex interpretability results, like the one we showed in Chapter 5 of this thesis. This fact let us think that extensions of fork algebras can provide a complete calculus for most of the logics used in system specification, thus giving the possibility of using tools built for fork algebras to verify or validate specifications written in a logical language.

In Chapter 4 we presented ω-closure fork algebras with urelements within the framework of institutions and complemented this model-theoretical point of view by formalizing its entailment system and proof calculus by resorting to the concepts of general logics. This formalization exposed the relationship existing between the classes of models of two extensions of the signature of the ω-closure fork algebras with urelements. A direct consequence of this formalization of ω-closure fork algebras with urelements is the reformulation of the set theoretical interpretability results as a representation maps between the corresponding institutions. An interesting example of this reformulation was presented in Chapter 5 for the interpretability of FOLTL in an extension of ω-closure fork algebras with urelements.

If we consider the existing representability results from different logics in extensions of fork algebras ([**Fri02, FO98, FBM02, FL03, FL06, FGLR05**]), presented as representation maps between institutions, Chapter 6 shows how partial specifications written in different logics can be put together in a single algebraic description of the system. The process of gluing partial specifications is enabled by the addition of *channels*, to synchronize symbols and new theories declaring the way these symbols are related.

The complete picture defines the formal foundations of the **A**$r_{\mathbf{g}}$*entum* project. It enables the possibility of building, validating and verifying specifications from partial descriptions of a system. These partial descriptions can be written in the best suited language. This is a clear advantage with respect to monolithic languages which require all the aspects of a system to be formalized in a possibly unnatural way, reducing the clarity and compactness of the specifications.

Some disadvantages must be singled out. During the specification process, for the methodology presented in Chapter 6 to be effective, design decisions must be added by means of fork-algebraic axioms. We consider that the possibility of adding these design decisions is a contribution of this thesis, but we believe fork-algebraic axioms will probably not look "natural" for those involved in the process of building logical descriptions of the system. This is why we made an effort to present, also in Chapter 6, some small examples of languages which are closer to those we present as better suited for building specifications, and allow to bridge the gap existing between two different logics without the need of using the language of fork algebras.

In Chapter 8 we described existing tool support for automatic validation and semi-automatic theorem-proving of fork algebraic specifications. Even considering the great advantage of having tools providing certain level of automation of the validation/verification process, we do not escape from the problems affecting the fields of (bounded) model-checking and theorem proving. In the case of bounded model-checking fork algebraic specifications, which of course is an *NP-hard* problem, we made several advances on optimizing the algorithms and structures used to generate models but the size of the specifications we are able to verify is still far from being close to an industrial size problem. In the case of theorem-proving there is no problem with the complexity of the algorithms but building proofs for interesting properties still requires a deep insight of the problem domain, and of the language of fork algebras. Additionally, if we observe the shape of the formulas obtained by the translations of logical partial specifications (see Chapter 7, §3), the complexity of the algebraic specification grows very rapidly. This certainly is a disadvantage because using **A**$r_{\mathbf{g}}$*entum* in industrial size problems highly depend on the real possibility of carrying on proofs in this setting.

Partial solutions to this problem were addressed. In [**Fri03**] Frias proposed a relational notation which, being more close to the original structure of the formulas, help in overcoming the problem of the introduction of complex relational terms interpreting certain logical structures like quantification. Also in [**Fri03**] a proof system was proved sound and complete for this relational notation. Following this approach it could be possible to obtain relational notation and a set of proof rules associated to each logic, thus producing simpler specifications, and allowing the implementation of these proof rules as strategies which could simplify some parts of the proving process.

In [**FLM06, FLM07**] Frias et al. presented a complete calculus for *Alloy* [**Jac02**] based on a fork algebraic interpretation of *Alloy* specifications. In this work a similar approach was followed in order to obtain a proof calculus as closer to the *Alloy* formulas as possible, in order to enhance its usability by *Alloy* users. This calculus was successfully implemented within *PVS* [**ORS92, ORSv95, Sha01**] and used to verify a medium size case-study presented by Pamela Zave in [**Zav05**].

Finally, if we compare our approach with those relaying on Grothendiek constructions, like CafeOBJ, then it is easy to observe certain lack of generality in using fork algebras as a common language in which partial descriptions are interpreted. Of course we understand this as a disadvantage, but we believe it is relativized by the fact that our approach pretends, not only to provide a framework for dealing with heterogeneous specifications, but also to preserve the existence of a complete calculus in which to prove properties across the partial specifications.

2. Further work

We identify three areas on which our approach to deal with heterogeneous specifications presents further directions of research. These areas are:

- the study of the class of full proper closure fork algebras with urelements from the point of view of general logics,
- the need for a high-level language in which to describe heterogeneous properties, and
- the tool $\mathbf{A}r_{\mathbf{g}}entum$.

In this section we present some topics in these areas that yet need to be developed.

2.1. The study of the full proper closure fork algebras with urelements from the point of view of general logics. In the field of general logics there have been defined several properties which may or may not be satisfied by a given institution, entailment system, logic, etc. Some of these properties are natural extensions of properties that come from the field of category theory, and others appear in order to gain better comprehension of the behavior of these new structures. Among them we find *compactness* of an entailment system, *liberality*, *admittance of initial models* and *exactness* of an institution; as well as *amalgamation*, *strong amalgamation* or *interpolation*. All these properties need to be studied because they provide important information about the limits of applicability of our approach to deal with heterogeneous specifications.

We already mentioned that our approach is very close to those using Grothendiek constructions. Being so close, it emerges the question on the relation between them. If we consider the diagram formed by the institutions of those logics used as specification languages and the institution of full proper closure fork algebras with urelements (i.e. the institution on which we write design decisions), it is possible to consider the Grothendiek institution to which this diagram is mapped. The *class of models* to which this construction leads is different from the full proper closure fork algebras with urelements, which, given an heterogeneous specification, are the models of the vertex of the co-limit co-cone constructed from the translation to fork algebra of the partial specifications. Thus, the later question could have an answer in the relationship between these two classes of models when in presence of the representation maps appearing in the diagram.

2.2. The need for a high-level language in which to describe heterogeneous properties. We already mentioned in §3 the importance of having a high-level language in which to provide design decisions that must be part of the global specification. Also in Chapter 6 we gave some ideas of what can be done in this direction. This field presents several problems that must be solved. A desirable property is that the "language" should admit the inclusion of a new logic without requiring its complete re-engineering. A direct consequence of this property is that the semantics of the language must be put in terms of a translation to full proper closure fork algebras with urelements. Any attempt to characterize the class of models of a monolithic language combining all the high-level logical languages will lead to an *ad-hoc* class of models like the one introduced by Henriksen and Thiagarajan for DLTL in [**HT99**].

Recalling the proposal we made in Chapter 6 for managing heterogeneous properties, there exist several works on synchronizing partial specifications through formulas. Some of them are due to Sernadas et al. [**SSC97a, SSC97b**]. It may be important to make a study characterizing the relationship between this approach, and the use of a language like those we presented in Chapter 6.

A totally different approach is the extraction of logical specifications and properties from UML diagrams. This can be done by assigning, to each class of diagrams, a precise semantics in terms of a collection of logical formulas, probably coming from different logics, like we did in a very informal way in Chapter 7.

2.3. The tool A$r_{\mathbf{g}}$*entum*. Finally, a lot of work must be done in the field of tool support for designing, validating and verifying systems specified through partial logical descriptions. As we mentioned before, we have done some work but **A**$r_{\mathbf{g}}$*entum* is far from being ready to be used.

First of all, a design tool is required. Designing in our proposal requires identifying symbols that must be glued together in design decisions which were not included in the partial specifications. If we consider a medium scale system, this job is almost impossible without a graphical tool enabling user-friendly options like showing/defining the relationships between symbols coming from different specifications, addition of channels and design decisions, etc.

Almost all of the translations are not implemented yet, thus requiring a lot of tedious work on programming them for the integration with the design tool.

In its origin, ReMo was a prototype produced in a master thesis. Thus, in comparison with its correctness, its extensiveness, or compactness were less important. Nowadays, releasing it as a tool for validating software requires a complete re-engineering. This job has been already started by Lorena Bourg, under the supervision of Marcelo Frias.

We have been working for a while on some optimizations for ReMo. The most important one is a generalization of the elimination of isomorphisms presented by Jackson et al. in [**JJD98**] for *Nitpick* [**Jac96, Dam97**]. This work is being carried on by Fernando Miranda and is in its final stage, but must be included in the new version of ReMo we are planning to develop.

Alloy[**Jac02**] is a sat-solver based bounded model-checker for a first-order relational language. The validation process is the following:

- A specification is written in relational logic (a first-order relational language),
- a property about the specification is stated to be validated,
- a command requiring the validation of the property and providing bounds for the domains is added,
- the specification and the negation of the property are translated to a single formula in propositional logic by assuming that all the domains are finite [**Jac00**],
- an off-the-shelf sat-solver searches for counterexamples of the formula.

The translation presented in [**Jac00**] for relational logic can easily be extended by providing a translation of the fork and choice operators in order to validate fork algebraic specifications. Thus, we are planning to implement a sat-solver based bounded model-checker for ω-closure fork algebras with urelements in order to get a fair comparison of the performance of ReMo.

Recalling our current theorem proving capabilities, even when we have successfully built a semantic embedding of ω-closure fork algebras with urelements in *PVS*, as we mentioned in §1, the verification of an industrial size problem is still out of reach because of the complexity of the algebraic specifications and the skills required to build proofs. To overcome this there are several things to be done such as the development of a library of already proved lemmas, the development of strategies to solve well known proof schemes, refinement of the translations by the addition of relational notation, etc.

Finally, some minor work must be done on integrating the semantic embedding of full proper closure fork algebras with urelements in *PVS* with ReMo. This will allow a nice interaction between model-checking validation and theorem-proving verification of models, providing the advantages we mentioned at the end of Chapter 8.

Epilogue

Science is capable of conferring enormous boons: it can lighten labour, abolish poverty, and enormously diminish disease. But if science is to bring benefits instead of death, we must bring to bear upon social, and especially international, organization, intelligence of the same high order that has enabled us to discover the structure of the atom. To do this effectively we must free ourselves from the domination of ancient shibboleths, and think freely, fearlessly and rationally about the new and appalling problems with which the human race is confronted by its conquest of scientific power.

Bertrand Arthur William Russell
"The Bomb and Civilization", 1945.

I believe science (from the latin *scientia*, "knowledge") is the art of explaining the world in its many different manifestations. Thus, we scientists are responsible for the knowledge we produce, because our creations may have a strong impact on the world by producing tools to transform it. Consequently, the social value of this knowledge can not be measured in terms of money or goods, metrics typically used for private property. Doing so implies the tacit acceptance that the power resulting from the knowledge we produce will remain in the hands of those who have the money to buy it, and use it for their own privilege, without caring about those who may suffer the consequences of its use.

Science should be a social and democratic construction in which the society is responsible, as much as ourselves, for keeping a close control of what we do, because it is the whole society that pays for the consequences of what we do.

Alexander Grothendieck refused to accept the 1988 Crafoord price, and in his letter announcing his decision he wrote:

> I do not doubt that before the end of the century, totally unforeseen events will completely change our notions about "science" and its goals and the spirit in which scientific work is done.

He was right in predicting that the new century would bring dramatic changes in the way we do our work, but he was wrong in being optimistic about these changes. The new century is deepening the scientific policies that, during the second half of the $20^{th.}$ century, established the idea that the knowledge is a merchandise and the scientists only have to care for expanding the horizons of “science”. States retreated and left the place to “the market” and to big companies that are wiling to pay, in prestige or money, for the knowledge we produce to turn it into power. Their power.

It is our responsibility as scientists to stop these changes by assuming clear political positions and making ethical decisions in order to socialize the knowledge, and put it to the service of each and every human being.

Bibliography

[Bar77] Jon Barwise (ed.), *Handbook of mathematical logic*, North Holland, 1977.

[BC$^+$90] J. R. Burch, Edmund M. Clarke, , Kenneth L. McMillan, D. L. Dill, and L. J. Hwang, *Symbolic model checking:* 10^{20} *states and beyond*, Proceedings of Symposium on Logic in Computer Science '90 (Philadelphia, PA), IEEE Computer Society, June 1990, pp. 428–439.

[BdV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema, *Modal logic*, Cambridge Tracts in Theoretical Computer Science, no. 53, Cambridge University Press, 2001.

[BFM98] Gabriel A. Baum, Marcelo F. Frias, and Tomas S. E. Maibaum, *A logic for real-time systems specification, its algebraic semantics, and equational calculus*, Proceedings of Algebraic Methodology And Software Technology (Amazonia, Brasil) (Armando M. Haeberer, ed.), Lecture Notes in Computer Science, vol. 1548, Springer-Verlag, January 1998, pp. 91–105.

[Bir35] Garret Birkhoff, *On the structure of abstract algebra*, Mathematical proceedings of the Cambridge philosophical society **31** (1935), 433–454.

[BM03] Rudolf Berghammer and Bernhard Möller (eds.), *7th. conference on relational methods in computer science (RelMiCS) - 2nd. international workshop on applications of kleene algebra*, Malente, Germany, May 2003.

[Boo47] George Boole, *The mathematical analysis of logic. being an essay towards a calculus of deductive reasoning*, Barclay & Macmillan and George Bell, Cambridge, UK and London, UK, 1847.

[BS81] Stanley Burris and H. P. Sankappanavar, *A course in universal algebra*, Graduate Texts in Mathematics, Springer-Verlag, Berlin, Germany, 1981.

[BW90] M. Barr and C. Wells, *Category theory for computer science*, Prentice Hall, London, 1990.

[CLM$^+$95] D. Cyrluk, Patrick Lincoln, Steven P. Miller, Paliath Narendran, Sam Owre, Sreeranga Rajan, John M. Rushby, Natarajan Shankar, Jens Ulrik, Jens U. Skakkebæk, Mandayam Srivas, and Friedrich von Henke, *Seven papers on mechanized formal verification*, Tech. Report SRI-CSL-95-3, Computer Science Laboratory, SRI International, January 1995.

[Cro93] R. Crole, *Categories for types*, Cambridge University Press, Cambridge, 1993.

[CRSS94] D. Cyrluk, Sreeranga Rajan, Natarajan Shankar, and Mandayam Srivas, *Effective theorem proving for hardware verification*, Proceedings of Theorem Provers in Circuit Design (TPCD) '94) (Bad Herrenalb, Germany) (Ramayya Kumar and Thomas Kropf, eds.), Lecture Notes in Computer Science, vol. 901, Springer-Verlag, 1994, pp. 203–222.

[CT51] Louise H. Chin and Alfred Tarski, *Distributive and modular laws in the arithmetic of relation algebras*, University of California Publications in Mathematics **New Series** (1951), no. 1, 341–384.

[Dam97] Craig A. Damon, *Nitpick: a tool for interactive design analysis*, Proceedings of the 19th. International Conference on Software Engineering (Boston, Massachusetts, USA) (W. Richards Adrion, ed.), Association for the Computer Machinery and IEEE Computer Society, ACM Press, May 1997, pp. 596–597.

[DF02] Răzvan Diaconescu and Kokichi Futatsugi, *Logical foundations of CafeOBJ*, Theoretical Computer Science **285** (2002), no. 2, 289–318.

[Dia02] Răzvan Diaconescu, *Grothendieck institutions*, Applied Categorical Structures **10** (2002), no. 4, 383–402.

[dM64] Augustus de Morgan, *On the syllogism: IV, and on logic of relations*, Transactions of the Cambridge Philosophical Society **10** (1864), 331–358, Reprinted in [**dM66**].

[dM66] ———, *On the syllogism, and other logical writings*, Yale University Press, 1966.

[EL86] E. Allen Emerson and Chin-Laung Lei, *Efficient model checking in fragments of the propositional μ-calculus (extended abstract)*, Proceedings of Symposium on Logic in Computer Science '86 (Cambridge, MA, USA) (Albert Meyer, ed.), IEEE Computer Society, June 1986, pp. 267–278.

[Eme90] E. Allen Emerson, *Temporal and modal logic*, Handbook of Theoretical Computer Science (Jan van Leeuwen, ed.), vol. B, Elsevier, Amsterdam, 1990.

[End72] Herbert B. Enderton, *A mathematical introduction to logic*, Academic Press, 1972.

[FBH97] Marcelo F. Frias, Gabriel A. Baum, and Armando M. Haeberer, *Fork algebras in algebra, logic and computer science*, Fundamenta Informaticae **32** (1997), 1–25.

[FBH98] ———, *Representability and program construction within fork algebras*, Logic Journal of the IGPL **6** (1998), no. 2, 227–257.

[FBHV95] Marcelo F. Frias, Gabriel A. Baum, Armando M. Haeberer, and Paulo A.S. Veloso, *Fork algebras are representable*, Bulletin of the Section of Logic **24** (1995), no. 2, 64–75.

[FBL01] Marcelo F. Frias, Gabriel A. Baum, and Carlos G. Lopez Pombo, *A comparisson of* $\mathbf{A_g}$ *with Alloy*, Proceedings of the 6th. Conference on Relational Methods in Computer Science (RelMiCS) - TARSKI (Oisterwijk, The Netherlands) (Harrie de Swart, ed.), October 2001, pp. 365–377.

[FBM02] Marcelo F. Frias, Gabriel A. Baum, and Tomas S. E. Maibaum, *Interpretability of first-order dynamic logic in a relational calculus*, Proceedings of the 6th. Conference on Relational Methods in Computer Science (RelMiCS) - TARSKI (Oisterwijk, The Netherlands) (Harrie de Swart, ed.), Lecture Notes in Computer Science, vol. 2561, Springer-Verlag, October 2002, pp. 66–80.

[FGLR05] Marcelo F. Frias, Juan P. Galeotti, Carlos G. Lopez Pombo, and Mario Roman, *Fork algebra as a formalism to reason across behavioral specifications (extended abstract)*, Proceedings of the 8th. Conference on Relational Methods in Computer Science (RelMiCS) - 3nd. International Workshop on Applications of Kleene Algebra (St. Catharines, Ontario, Canada) (Ivo Düntsch and Michael Winter, eds.), February 2005, pp. 61–68.

[FGSB05] Marcelo F. Frias, Rodolfo Gamarra, Gabriela Steren, and Lorena Bourg, *A strategy for efficient verification of relational specification, based in monotonicity analysis*, Proceedings of the 20th. IEEE/ACM International Conference on Automated Software Engineering (Long Beach, California, USA) (David F. Redmiles, Tom Ellman, and Andrea Zisman, eds.), Association for the Computer Machinery and IEEE Computer Society, ACM Press, November 2005, pp. 305–308.

[FHV95] Marcelo F. Frias, Armando M. Haeberer, and Paulo A.S. Veloso, *A finite axiomatization for fork algebras*, Bulletin of the Section of Logic **24** (1995), no. 4, 193–200.

[FHV97] ———, *A finite axiomatization for fork algebras*, Logic Journal of the IGPL **5** (1997), no. 3, 311–319.

[Fia96] José Luis Fiadeiro, *On the emergence of properties in component-based systems*, Proceedings of the 1996 Algebraic Methodology and Software Technology – AMAST 96 (Munich, Germany) (Martin Wirsing and Maurice Nivat, eds.), Lecture Notes in Computer Science, vol. 1101, Springer-Verlag, July 1996.

[Fia05] ———, *Categories for software engineering*, Springer-Verlag, 2005.

[FL03] Marcelo F. Frias and Carlos G. Lopez Pombo, *Time is on my side*, in Berghammer and Möller [**BM03**], pp. 105–111.

[FL06] ———, *Interpretability of first-order linear temporal logics in fork algebras*, Journal of Logic and Algebraic Programming **66** (2006), no. 2, 161–184.

[FLB02] Marcelo F. Frias, Carlos G. Lopez Pombo, and Gabriel A. Baum, *The specification language* $\mathbf{A_g}$, Available at `http://www.dc.uba.ar/people/profesores/mfrias/Files/Downloads/Ag.ps`, February 2002.

[FLB$^+$05] Marcelo F. Frias, Carlos G. Lopez Pombo, Gabriel A. Baum, Nazareno M. Aguirre, and Tomas S. E. Maibaum, *Reasoning about static and dynamic properties in Alloy: A purely relational approach*, ACM Transactions on Software Engineering and Methodology **14** (2005), no. 4, 478–526.

[FLM06] Marcelo F. Frias, Carlos G. Lopez Pombo, and Mariano Miguel Moscato, *Dynamite: Alloy Analyzer+PVS in the analysis and verification of Alloy specifications*, Proceedings of the 1st ACM SIGSOFT Alloy Workshop (Portland, Oregon, USA) (Daniel Jackson and Pamela Zave, eds.), Association for the Computer Machinery, November 2006.

[FLM07] ———, *Alloy Analyzer+PVS in the analysis and verification of Alloy specifications*, Proceedings of the 13th. International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007) (Braga, Portugal) (Orma Grumberg and Michael Huth, eds.), Lecture Notes in Computer Science, vol. 4424, Springer-Verlag, April 2007, pp. 587–601.

[FM92] José Luis Fiadeiro and Tomas S. E. Maibaum, *Temporal theories as modularisation units for concurrent system specification*, Formal Aspects of Computing **4** (1992), no. 3, 239–272.

[FO98] Marcelo F. Frias and Ewa Orlowska, *Equational reasoning in non-classical logics*, Journal of Applied Non-classical Logics **8** (1998), no. 1–2, 27–66.

[Fri02] Marcelo F. Frias, *Fork algebras in algebra, logic and computer science*, Advances in logic, vol. 2, World Scientific Publishing Co., Singapore, 2002.

[Fri03] ———, *Translating with sense*, in Berghammer and Möller [**BM03**], pp. 257–269.

[FS87] José Luis Fiadeiro and Amílcar Sernadas, *Structuring theories on consequence*, Selected papers from the 5th Workshop on Specification of Abstract Data Types (Gullane, Scotland) (Andrzej Tarlecki and Donald Sannella, eds.), Lecture Notes in Computer Science, Springer-Verlag, September 1987, pp. 44–72.

[GB84] Joseph A. Goguen and Rod M. Burstall, *Introducing institutions*, Proceedings of the Carnegie Mellon Workshop on Logic of Programs (Edmund M. Clarke and Dexter Kozen, eds.), Lecture Notes in Computer Science, vol. 184, Springer-Verlag, 1984, pp. 221–256.

[GB92] ———, *Institutions: abstract model theory for specification and programming*, Journal of the ACM **39** (1992), no. 1, 95–146.

[GMW79] Michael J.C. Gordon, Robin Milner, and C. Wadsworth (eds.), *Edinburgh LCF: A mechanized logic of computation*, Lecture Notes in Computer Science, vol. 78, Springer-Verlag, 1979.

[GS05] Rodolfo Gamarra and Gabriela Steren, *Implementación de una herramienta de model-checking basada en álgebra relacional*, Master's thesis, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, April 2005, Advisor: Marcelo F. Frias.

[Gyu97] Viktor Gyuris, *A short proof of representability of fork algebra*, Theoretical Computer Science **188** (1997), no. 1–2, 211–220.

[Hal63] Paul R. Halmos, *Lectures on boolean algebra*, D. Van Nostrand, Co., Inc., Princeton, 1963.

[HBS93a] Armando M. Haeberer, Gabriel A. Baum, and Gunther Schmidt, *Dealing with non-constructive specifications involving quantifiers*, Monografias en Ciências da Computação 4/93, Departamento de Informatica, Pontifícia Universidade Católica do Rio de Janeiro, May 1993.

[HBS93b] ———, *On the smooth calculation of relational recursive expressions out of first-order non-constructive specificationes involving quantifiers*, International Conference on Formal Methods in Programming and Their Applications (Academgorodok, Novosibirsk, Russia) (Dines Bjørner, Manfred Broy, and Igor V. Pottosin, eds.), Lecture Notes in Computer Science, vol. 735, Springer-Verlag, June 1993, pp. 281–298.

[HKT00] David Harel, Dexter Kozen, and J. Tiuryn, *Dynamic logic*, Foundations of Computing, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.

[HMT71] Leon A. Henkin, J. Donald Monk, and Alfred Tarski, *Cylindric algebras, part I*, vol. 64, North Holland, 1971.

[HMT85] ———, *Cylindric algebras, part II*, vol. 115, North Holland, 1985.

[HT99] Jesper G. Henriksen and P.S. Thiagarajan, *Dynamic linear time temporal logic*, Annals of Pure and Applied Logic **96** (1999), no. 1–3, 187–207.

[HV91] Armando M. Haeberer and Paulo A.S. Veloso, *Partial relations for program derivation: adequacy, inevitability and expressiveness*, Proceedings of the Working Conference on Constructing Programs from Specifications '91, IFIP TC2, Constructing Programs from Specifications, North Holland, 1991, pp. 310–352.

[Jac96] Daniel Jackson, *Nitpick: a checkable specification language*, Proceedings of the 1st. ACM SIGSOFT Workshop on Formal Methods in Software Practice (Mark Ardis, ed.), Association for the Computer Machinery, ACM Press, January 1996, pp. 60–69.

[Jac00] ———, *Automating first-order relational logic*, Proceedings of the 8th ACM SIGSOFT international symposium on Foundations of software engineering (San Diego, California, United States), Association for the Computer Machinery, ACM Press, 2000, pp. 130–139.

[Jac02] ———, *Alloy: a lightweight object modelling notation*, ACM Transactions on Software Engineering and Methodology **11** (2002), no. 2, 256–290.

[JJD98] Daniel Jackson, Somesh Jha, and Craig A. Damon, *Isomorph-free model enumeration: a new method for checking relational specifications*, ACM Transactions on Programming Languages and Systems **20** (1998), no. 2, 302–343.

[JT51] Bjarni Jónnson and Alfred Tarski, *Boolean algebra with operators, part I*, American Journal of Mathematics **73** (1951), 891–939.

[JT52] ———, *Boolean algebra with operators, part II*, American Journal of Mathematics **74** (1952), 127–162.

[KP81] Dexter Kozen and Rohit Parikh, *An elementary proof of the completeness of PDL*, Theoretical Computer Science **14** (1981), no. 1, 113–118.

[LF06] Carlos G. Lopez Pombo and Marcelo F. Frias, *Fork algebras as a sufficiently rich universal institution*, Proceedings of 11th International Conference on Algebraic Methodology and Software Technology, AMAST 2006 (Kuressaare, Estonia) (Michael Johnson and Varmo Vene, eds.), Lecture Notes in Computer Science, vol. 4019, Springer-Verlag, July, 5–8 2006, pp. 235–247.

[LOS02] Carlos G. Lopez Pombo, Sam Owre, and Natarajan Shankar, *A semantic embedding of the* $\mathbf{A_g}$ *dynamic logic in PVS*, Technical Report SRI-CSL-02-04, Computer Science Laboratory, SRI International, July 2002.

[Löw15] Leopold Löwenheim, *Uber Möglichkeiten im Relativkalkul*, Mathematische Annalen **76** (1915), 447–470.

[Lyn50] Roger C. Lyndon, *The representation of relation algebras, part I*, Annals of Mathematics (series 2) **51** (1950), no. 2, 707–729.

[Mad89] Roger D. Maddux, *Finitary algebraic logic*, Zeitschrift fur Mathematisch Logik und Grundlagen der Mathematik **35** (1989), 321–332.

[Mad91] ———, *The origin of relation algebras in the development of the calculus of relations*, Studia Logica **50** (1991), no. 3/4, 421–455.

[McK70] Ralph McKenzie, *Representation of ointegral relation algebras*, Michigan Mathematical Journal **17** (1970), 279–287.

[McL71] Saunder McLane, *Categories for working mathematicians*, Graduate Texts in Mathematics, Springer-Verlag, Berlin, Germany, 1971.

[Mes89] José Meseguer, *General logics*, Proceedings of the Logic Colloquium '87 (Granada, Spain) (Heinz-Dieter Ebbinghaus, José Fernandez-Prida, Manuel Garrido, Daniel Lascar, and Mario Rodríguez Artalejo, eds.), vol. 129, North Holland, 1989, pp. 275–329.

[Mes92a] ———, *Conditional rewriting logic as a unified model of concurrency*, Theoretical Computer Science **96** (1992), no. 1, 73–155, Also in [**Mes92b**].

[Mes92b] ———, *Conditional rewriting logic as a unified model of concurrency*, Proceedings of the Second Workshop on Concurrency and compositionality (WCC'92) (San Miniato, Italy) (Rocco DeNicola and Ugo Montanari, eds.), March 1992, Also in [**Mes92a**], pp. 73–155.

[MOM93] Narciso Martí-Oliet and José Meseguer, *Rewriting logic as a logical and semantic framework*, Technical Report SRI-CSL-93-05, Computer Science Laboratory, SRI International, August 1993, Also in [**MOM96**] and [**MOM01**].

[MOM96] ———, *Rewriting logic as a logical and semantic framework*, Proceedings of First International Workshop on Rewriting Logic and its Applications (WRLA'96) (José Meseguer, ed.), Electronic Notes in Theoretical Computer Science, vol. 4, Elsevier, 1996, Also in [**MOM93**] and [**MOM01**], pp. 190–225.

[MOM01] ———, *Rewriting logic as a logical and semantic framework*, Handbook of Philosophical Logic (Dov Gabbay and Franz Guenthner, eds.), vol. 9, Kluwer Academic Publishers, second ed., 2001, Also in [**MOM93**] and [**MOM96**].

[Mon64] J. Donald Monk, *On representable relation algebras*, Michigan Mathematical Journal **11** (1964), 207–210.

[MP95] Zohar Manna and Amir Pnueli, *Temporal verification of reactive systems*, Springer-Verlag, New York, 1995.

[MS95] Steven P. Miller and Mandayam Srivas, *Formal verification of the AAMP5 microprocessor: a case study in the industrial use of formal methods*, Proceedings of Workshop on Industrial-Strength Formal Specification Techniques (WIFT) '95 (Boca Raton, FL), IEEE Computer Society, April 1995, pp. 2–16.

[MSS92] Szabolcs Mikulás, Ildikó Sain, and Andras Simon, *Complexity of equational theory of relational algebras with projection elements*, Bulletin of the Section of Logic **21** (1992), no. 3, 103–111.

[OMG04a] Object Management Group, *Object constraint language specification*, Object Management Group, 2004, version 1.5.

[OMG04b] ———, *OMG SysML specification coversheet*, Object Management Group, 2004, version 1.0.

[ORS92] Sam Owre, John M. Rushby, and Natarajan Shankar, *PVS: A prototype verification system*, Proceedings of the 11th International Conference on Automated Deduction (CADE) (Saratoga, NY) (Deepak Kapur, ed.), Lecture Notes in Artificial Intelligence, vol. 607, Springer-Verlag, June 1992, pp. 148–752.

[ORSSC98] Sam Owre, John M. Rushby, Natarajan Shankar, and David Stringer-Calvert, *PVS: an experience report*, Proceedings of Applied Formal Methods – (FM-Trends) '98 (Boppard, Germany) (Dieter Hutter, Werner Stephan, Paolo Traverso, and Markus Ullman, eds.), Lecture Notes in Computer Science, vol. 1641, Springer-Verlag, October 1998, pp. 338–345.

[ORSv95] Sam Owre, John M. Rushby, Natarajan Shankar, and Friedrich von Henke, *Formal verification for fault-tolerant architectures: prolegomena to the design of PVS*, IEEE Transactions on Software Engineering **21** (1995), no. 2, 107–125.

[OS93] Sam Owre and Natarajan Shankar, *Abstract datatypes in PVS*, Technical Report SRI-CSL-93-9R, Computer Science Laboratory, SRI International, December 1993, Subtantially revised in June 1997.

[Par72] David Lorge Parnas, *On the criteria to be used in decomposing systems into modules*, Communications of the ACM **15** (1972), no. 12, 1053–1058, See also [**Par02**].

[Par02] ———, *On the criteria to be used in decomposing systems into modules*, Software pioneers: contributions to software engineering (Manfred Broy and Ernst Denert, eds.), Springer-Verlag, New York, 2002, See also [**Par72**]., pp. 411–427.

[Pei70] Charles Sanders Peirce, *Description of a notation for the logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic*, Memoirs of the American Academy of Science **9** (1870), 317–378.

[Pei83a] ———, *Note B: the logic of relatives*, in *Studies in logic by members of the John Hopkins University* [**Pei83b**], Reprinted in [**Pei83**]., pp. 187–203.

[Pei83b] Charles Sanders Peirce (ed.), *Studies in logic by members of the John Hopkins University*, Little, Brown and Co., Boston, 1883, Reprinted in [**Pei83**].

[Pei83] Charles Sanders Peirce (ed.), *Studies in logic by members of the John Hopkins University*, John Benjamins Publishing Co., Amsterdam, The Netherlands and Philadelphia, USA, 1983, Reprint of [**Pei83b**].

[RS01] Harald Rueß and Natarajan Shankar, *Deconstructing Shostak*, Proceedings of Symposium on Logic in Computer Science '01 (Boston, MA, USA) (Joseph Y. Halpern, ed.), IEEE Computer Society, June 2001, pp. 19–28.

[RSS95] Sreeranga Rajan, Natarajan Shankar, and Mandayam Srivas, *An integration of model-checking with automated proof checking*, Proceedings of the 8th. Computer Aided Verification (CAV) (Liege, Belgium) (Pierre Wolper, ed.), Lecture Notes in Computer Science, vol. 939, Springer-Verlag, June 1995, pp. 84–97.

[RW13] Bertrand Russell and Alfred North Whitehead, *Principia mathematica*, vol. 1–3, Cambridge University Press, 1910–1913.

[Sch95] F. W. K. Ernst Schöder, *Vorlesungen über die algebra der logik (exact logik)*, Algebra und Logik der Relative, vol. 3, Thoemmes Press, 1895.

[Seg77] Krister Segerberg, *A completeness theorem in the modal logic of programs*, Notices of the American Mathematical Society **24** (1977), no. 6, A–552, Also in [**Seg82**].

[Seg82] ———, *A completeness theorem in the modal logic of programs*, Proceedings of Seminar held at the Stefan Banach International Mathematical Center 1978 (Warsaw, Poland) (Tadeusz Traczyck, ed.), Universal Algebra and Applications, vol. 9, Banach Center Publications, 1982, Also in [**Seg77**]., pp. 31–46.

[Sha01] Natarajan Shankar, *Using decision procedures with a higher-order logic*, Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs) (Edinburgh, Scotland) (R.J. Boulton and P.B. Jackson, eds.), Lecture Notes in Computer Science, vol. 2152, Springer-Verlag, September 2001, pp. 5–26.

[Sha02] ———, *Static analysis for safe destructive updates in a functional language*, Proceedings of the 11th International Workshop on Logic-based Program Synthesis and Transformation (LOPSTR) (Paphos, Cyprus) (A. Pettorossi, ed.), Lecture Notes in Computer Science, vol. 2372, Springer-Verlag, November 2002, pp. 1–24.

[Sho84] Robert E. Shostak, *Deciding combinations of theories*, Journal of the ACM **31** (1984), no. 1, 1–12.

[Sik64] Roman Sikorsky, *Boolean algebras*, Springer-Verlag, Berlin, Germany, 1964.

[SR02] Natarajan Shankar and Harald Rueß, *Combining Shostak theories*, Proceedings of International Conference on Rewriting Techniques and Applications (RTA) '02 (Copenhagen, Denmark) (Sophie Tison, ed.), Lecture Notes in Computer Science, vol. 2378, Springer-Verlag, July 2002, pp. 1–18.

[SS99] Hassen Saïdi and Natarajan Shankar, *Abstract and model check while you prove*, Proceedings of the 12th. Computer Aided Verification (CAV) (Trento, Italy) (Nicolas Halbwachs and Doron Peled, eds.), Lecture Notes in Computer Science, vol. 1633, Springer-Verlag, July 1999, pp. 443–454.

[SSC97a] Amílcar Sernadas, Cristina Sernadas, and Carlos Caleiro, *Synchronization of logics*, Studia Logica **59** (1997), no. 1, 217–247.

[SSC97b] ———, *Synchronization of logics with mixed rules: Completeness preservation*, Proceedings of the 1997 Algebraic Methodology and Software Technology – AMAST 97 (Macquarie University, Sydney, Australia) (Michael Johnson, ed.), Lecture Notes in Computer Science, vol. 1349, Springer-Verlag, December 1997, pp. 465–478.

[Sto36] Marshall Harvey Stone, *The theory of representations for boolean algebras*, Transactions of the American Mathematical Society **40** (1936), 37–111.

[SW93] Detlef Sieling and Ingo Wegener, *Reduction of OBDDs in linear time*, Information Processing Letter **48** (1993), no. 3, 139–144.

[Tar41] Alfred Tarski, *On the calculus of relations*, Journal of Symbolic Logic **6** (1941), no. 3, 73–89.

[Tar55a] ———, *Contributions to the theory of models III*, Indagationes Mathematicae **17** (1955), 56–64, Also in [**Tar55b**].

[Tar55b] ———, *Contributions to the theory of models III*, Proceedings of Koninklijkle Nederlandsle Akademie van Wetenschappen, Series A, no. 58, 1955, Also in [**Tar55a**].

[Tar55c] ———, *Lattice-theoretic fixpoint theorem and its applications*, Pacific Journal of Mathematics **5** (1955), 285–309.

[Tar96] Andrzej Tarlecki, *Moving between logical systems*, Selected papers from the 11th Workshop on Specification of Abstract Data Types Joint with the 8th COMPASS Workshop on Recent Trends in Data Type Specification (Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, eds.), Lecture Notes in Computer Science, vol. 1130, Springer-Verlag, 1996, pp. 478–502.

[TBG91] Andrzej Tarlecki, Rod M. Burstall, and Joseph A. Goguen, *Some fundamental algebraic tools for the semantics of computation: Part 3: Indexed categories*, Theoretical Computer Science **91** (1991), no. 2, 239–264.

[TG87] Alfred Tarski and Steven Givant, *A formalization of set theory without variables*, American Mathematical Society Colloqium Publications, Providence, RI, USA, 1987.

[Tho52] Frederick Burtis Thompson, *Some contributions to abstract algebra and metamathematics*, Ph.D. thesis, University of California, Berkeley, 1952, Advisor: Alfred Tarski.

[TT52] Alfred Tarski and Frederick Burtis Thompson, *Some general properties of cylindric algebras. preliminary report*, Bulletin of the American Mathematical Society (Abstracts) **58** (1952), 65.

[VH91] Paulo A.S. Veloso and Armando M. Haeberer, *A finitary relational algebra for classical first-order logic*, Bulletin of the Section of Logic **20** (1991), no. 2, 52–62.

[VHB92] Paulo A.S. Veloso, Armando M. Haeberer, and Gabriel A. Baum, *On formal program construction within an extended calculus for binary relations*, Monografias en Ciências da Computação 19/92, Departamento de Informatica, Pontifícia Universidade Católica do Rio de Janeiro, May 1992.

[vJ01] Joachim van den Berg and Bart Jacobs, *The* LOOP *compiler for Java and JML*, Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (Genova, Italy) (T. Margaria and W. Yi, eds.), Lecture Notes in Computer Science, vol. 2031, Springer-Verlag, April 2001, pp. 299–312.

[Zav05] Pamela Zave, *A formal model of addressing for interoperating networks*, Proceedings of Formal Methods 2005: the 13th. International FME Symposium (Newcastle, UK) (John Fitzgerald, Ian J. Hayes, and Andrzej Tarlecki, eds.), Lecture Notes in Computer Science, vol. 3582, Springer-Verlag, July 2005, pp. 318–333.

APPENDIX A

Raimondo Lullos schematic logical proof of the existence of God

It is important to notice that we do not necessarily support Lullos schematic proof because we did not check its correctness; we only present it as a historical remark.

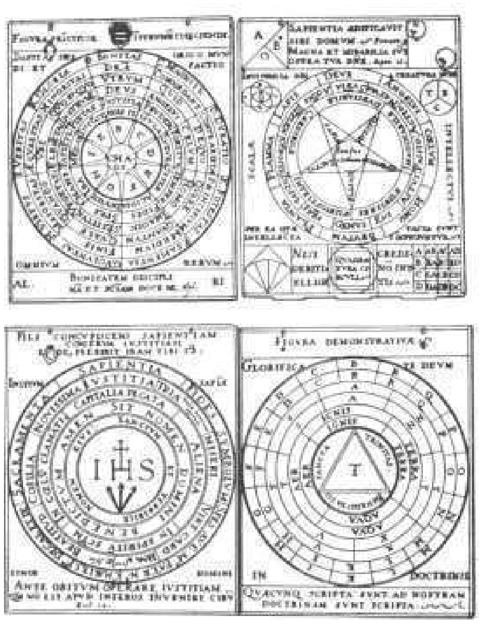


FIGURE A.1. Raimondo Lullos schematic logical proof of the existence of God.