Universitatea din București
Facultatea de Matematică și Informatică

Lucrare de disertație

An Institutional Foundation for the $\mathbb{K}$ Semantic Framework

Profesor îndrumător:
Conf. dr. Traian Florin Șerbănuță

Absolvent:
Claudia Elena Chiriță

București, 2014

Rezumat

$\mathbb{K}$ framework [18] este un framework semantic executabil bazat pe rescriere și utilizat în definirea limbajelor de programare, a sistemelor de calcul computațional, a sistemelor de tipuri și a uneltelor de analiză formală. A fost dezvoltat ca o alternativă la frameworkurile SOS (structural operational semantics) existente și a fost folosit de-a lungul timpului în definirea unor limbaje de programare reale precum C, Java, Verilog sau Scheme, în studierea metodelor de verificare în timpul execuției (runtime verification) și în dezvoltarea unor unelte software de analiză precum type checkere, type inferencere, model checkere, sau verificatoare bazate pe aserțiuni de tip Hoare. O expunere detaliată a frameworkului poate fi găsită în [19]. Programul software asociat frameworkului semantic $\mathbb{K}$ [6] permite scrierea unor definiții modulare și executabile pentru limbaje de programare complexe, dar facilitează în același timp și proiectarea unor limbaje de programare noi permițând utilizatorului să testeze programe și să exploreze comportamentul acestora în mod exhaustiv.

Bazându-ne pe studiile recente publicate în [17, 20] ce încearcă să stabilească fundamentele teoretice ale frameworkului semantic $\mathbb{K}$ și conexiunile cu alte frameworkuri semantice și sisteme formale precum semantica reductivă (reduction semantics), logica Hoare și logica separativă (separation logic), propunem în această lucrare o formalizare instituțională [9] a sistemelor logice ce stau la baza frameworkului $\mathbb{K}$: logica potrivirilor (matching logic) și logica accesibilității (reachability logic). Această formalizare va permite extinderea utilizării frameworkului prin exploatarea potențialului său ca un posibil limbaj de specificare. Mai mult, va permite stabilirea unor relații riguroase din punct de vedere matematic între $\mathbb{K}$ și alte limbaje similare, înlesnind integrarea pachetelor software și a tehnicilor de verificare corespunzătoare acestora.

Logica potrivirilor [17] este un sistem formal utilizat pentru descrierea și studierea proprietăților structurale ale obiectelor matematice și ale expresiilor specifice limbajelor de programare folosind tehnici de pattern matching. Propozițiile specifice logicii potrivirilor se numesc patternuri și sunt definite inductiv, similar termenilor din logica de ordinul întâi, folosind simbolurile de operație ale unei signaturi multisortate date, conectori booleeni și cuantificatori. Din punct de vedere semantic, modelele logicii potrivirilor sunt multialgebre. Patternurile sunt interpretate în multialgebre ca submulțimi ale mulțimilor subiacente acestora, definind astfel o relație de satisfacere ternară între patternuri, multialgebre și elemente ale multialgebrelor, numite stări.

Spre deosebire de logica de ordinul întâi, logica potrivirilor nu admite o formalizare fidelă ca instituție. Aceasta este o consecință atât a naturii ternare a relației de satisfacere, cât și a clasificării patternurilor în funcție de sorturi, în mod similar clasificării propozițiilor logicilor temporale cu ramificații în propoziții de stare (state), respectiv propoziții de cale (path). Pentru a depăși aceste limitări ale noțiunii de instituție ne bazăm pe conceptul de instituție stratificată introdus în [2], care extinde instituțiile prin considerarea unor abstractizări ale stărilor modelelor și prin definirea unei relații de satisfacere parametrizate nu numai de signaturi, ci și de stări ale modelelor. Extindem această noțiune asociind propozițiilor clase determinate de signaturi ce parametrizează atât stratificarea modelelor cât și relația de satisfacere. Prezentăm de asemenea descrierea logicilor potrivirilor și computation-tree (CTL) ca instituții stratificate cu clase și adaptăm construcția canonică a unei instituții simple pornind de la o instituție stratificată astfel încât să surprindă și rolul claselor propozițiilor.

Principalul avantaj al utilizării instituțiilor stratificate cu clase în formalizarea logicii potrivirilor este acela că ne permite extinderea construcției logicii accesibilității descrise în [20], astfel încât să admită descrierea patternurilor folosind și alte sisteme logice diferite de cea a potrivirilor. Logica accesibilității este un formalism pentru verificarea programelor prin care sistemele de tranziție corespunzătoare semanticii operaționale a limbajelor de programare pot fi descrise folosind reguli de accesibilitate. Aceste reguli au la bază patternuri și generalizează triplete Hoare pentru a specifica tranziții între configurațiile programelor (în mod asemănător regulilor de rescriere a termenilor). Logica accesibilității poate fi astfel considerată o alternativă independentă de limbaj a semanticilor axiomatice și a sistemelor de demonstrații (proof systems) particulare fiecărui limbaj în parte. Definim în această lucrare o instituție abstractă a logicii accesibilității peste o instituție stratificată cu clase oarecare, ce permite recuperarea noțiunii originale de accesibilitate prin instanțierea parametrului său cu logica potrivirilor.

Prin definirea atât a logicii potrivirilor cât și a logicii accesibilității ca instituții, putem integra aceste formalisme în grafuri ale logicilor limbajelor de specificare eterogene bazate pe instituții precum HetCasl [15]. Ca un rezultat direct al acestei formalizări, frameworkul $\mathbb{K}$ poate moșteni sistemele de module dezvoltate pentru specificații construite peste instituții arbitrare ce conțin operatori dedicați de agregare, redenumire, extindere, ascundere și parametrizare a modulelor. În plus, integrarea va permite combinarea logicii accesibilității și a programelor software suport oferite de $\mathbb{K}$ cu alte sisteme logice și uneltele lor software atașate. Încheiem această lucrare cu efortul preliminar integrării frameworkului $\mathbb{K}$ în Hets [16], al descrierii unor comorfisme ce codifică logicile potrivirilor și a accesibilității în instituția logicii de ordinul întâi.

Contents

1 Introduction

The $\mathbb{K}$ framework [18] is an executable semantic framework based on rewriting and used for defining programming languages, computational calculi, type systems and formal-analysis tools. It was developed as an alternative to the existing operational-semantics frameworks and over the years has been employed to define actual programming languages, to study runtime verification methods and to develop analysis tools such as type checkers, type inferencers, model checkers and verifiers based on Hoare-style assertions. A comprehensive overview of the framework can be found in [19]. Its associated tool [6] enables the development of modular and executable definitions of languages, and moreover, it allows the user to test programs and to explore their behaviour in an exhaustive manner, facilitating in this way the design of new languages.

Driven by recent developments on the theoretical foundations of the $\mathbb{K}$ semantic framework [17, 20] and on the established connections with other semantic frameworks and formal systems such as reduction semantics, Hoare logic and separation logic, we propose an institutional formalisation [9] of the logical systems on which the $\mathbb{K}$ framework is based: matching and reachability logic. This would allow us to extend the usage of $\mathbb{K}$ by focusing on its potential as a formal specification language, and furthermore, through its underlying logics, to establish rigorous mathematical relationships between $\mathbb{K}$ and other similar languages, enabling the integration of their verification tools and techniques.

Matching logic [17] is a formal system used to express properties about the structure of mathematical objects and language constructs, and to reason about them by means of pattern matching. Its sentences, called patterns, are built in an inductive manner, similar to the terms of first-order logic, using operation symbols provided by a many-sorted signature, as well as Boolean connectives and quantifiers. The semantics is defined in terms of multialgebras, which interpret patterns as subsets of their carriers. This leads to a ternary satisfaction relation between patterns, multialgebras and elements (or states) of multialgebras.

Unlike first-order logic, matching logic is difficult to formalise faithfully as an institution due to the ternary nature of its satisfaction relation and to the fact that patterns are classified by sorts, much in the way the sentences of branching temporal logics are classified into state or path sentences and evaluated accordingly. We overcome this limitations by relying on the concept of stratified institution developed in [2], which extends institutions with an abstract notion of model state and defines a parameterised satisfaction relation that takes into account the states of models. We further develop this concept by adding classes, which are determined by signatures, associated with sentences, and parameterise both the stratification of models and the satisfaction relation. We show that both matching and computation-tree logic can be described as stratified institutions with classes, and we adapt the canonical construction of an ordinary institution from a stratified one presented in [2] to take into consideration the role of classes.

The main advantage of using stratified institutions with classes to formalise matching logic is that we can extend the construction of reachability logic described in [20] from matching to other logical systems. Reachability logic is a formalism for program verification through which transition systems that correspond to the operational semantics of programming languages can be described using reachability rules; these rules rely on patterns and generalise Hoare triples in order to specify transitions between program configurations (similarly to term-rewrite rules). Therefore, reachability logic can be seen as a language-independent alternative to the axiomatic semantics and proof systems particular to each language. In our work, we define an abstract institution of reachability logic over an arbitrary stratified institution with classes such that by instantiating this parameter with matching logic we recover the original notion of reachability.

Having both matching and reachability logic defined as institutions allows us to integrate them into the logic graphs of institution-based heterogeneous specification languages such as HetCasl [15]. As an immediate result, the $\mathbb{K}$ framework can inherit the powerful module systems developed for specifications built over arbitrary institutions, with dedicated operators for aggregating, renaming, extending, hiding and parameterising modules. In addition, this will enable us to combine reachability logic and the tool support provided by $\mathbb{K}$ with other logical systems and tools. Towards that end, as a preliminary effort to integrate the $\mathbb{K}$ framework into Hets [16], we describe comorphism from matching and reachability logic to the institution of first-order logic.

2 Preliminaries

2.1 Institution Theory

The notion of institution [9] has evolved from the abstract model theory of Barwise [3] and the category theory of Mac Lane and Eilenberg [13] in order to address the growing number of logical systems used in formal specification and verification. It formalises the intuition about logics by abstracting their alphabets, syntax, semantics and the satisfaction relation between them. Due to their general nature, institutions proved to be successful in capturing a great variety of logical systems and in developing numerous results at an abstract level [7, 22], thus enabling these results to be applied to all or most of the logical systems described as institutions. In this way, institution theory gives a unique, relativistic and non-essentialist approach to logic, adhering to the universal-logic trend of Béziau [4].

Definition 1. An institution $\mathbf{I} = (\mathbb{S}\mathrm{ig}^{\mathbf{I}}, \mathrm{Sen}^{\mathbf{I}}, \mathrm{Mod}^{\mathbf{I}}, \models^{\mathbf{I}})$ consists of

- a category $\mathbb{S}\mathrm{ig}^{\mathbf{I}}$ whose objects are called *signatures*,
- a *sentence functor* $\mathrm{Sen}^{\mathbf{I}} \colon \mathbb{S}\mathrm{ig}^{\mathbf{I}} \to \mathbb{S}\mathrm{et}$ giving for every signature Σ the set $\mathrm{Sen}^{\mathbf{I}}(\Sigma)$ of Σ-*sentences* and for every signature morphism ϕ the *sentence translation map* $\mathrm{Sen}^{\mathbf{I}}(\phi)$,
- a *model functor* $\mathrm{Mod}^{\mathbf{I}} \colon (\mathbb{S}\mathrm{ig}^{\mathbf{I}})^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ defining for every signature Σ the category $\mathrm{Mod}^{\mathbf{I}}(\Sigma)$ of Σ-*models* and Σ-*model homomorphisms*, and for every signature morphism ϕ the *reduct functor* $\mathrm{Mod}^{\mathbf{I}}(\phi)$,
- a binary Σ-*satisfaction relation* $\models^{\mathbf{I}}_{\Sigma} \subseteq |\mathrm{Mod}^{\mathbf{I}}(\Sigma)| \times \mathrm{Sen}^{\mathbf{I}}(\Sigma)$, for every signature Σ,

such that the *satisfaction condition*

$$M' \models^{\mathbf{I}}_{\Sigma'} \mathrm{Sen}^{\mathbf{I}}(\phi)(\rho) \quad \text{iff} \quad \mathrm{Mod}^{\mathbf{I}}(\phi)(M') \models^{\mathbf{I}}_{\Sigma} \rho$$

holds for any signature morphism $\phi \colon \Sigma \to \Sigma'$, any Σ'-model M' and any Σ-sentence ρ.

Notation 1. We may omit the subscripts and superscripts in the notations of the entities of institutions when there is no risk of confusion: for example, $\models^{\mathbf{I}}_{\Sigma}$ may be simply denoted by $\models$ when the considered institution and signature are clear. We may also denote the sentence translation $\mathrm{Sen}^{\mathbf{I}}(\phi)$ by $\phi(\_)$ and the reduct functor $\mathrm{Mod}^{\mathbf{I}}(\phi)$ by $\_\!\upharpoonright_{\phi}$. When $M = M'\!\upharpoonright_{\phi}$ we say that M is a ϕ-*reduct* of M' and that M' is a ϕ-*expansion* of M.

The literature on specification languages [22] and abstract model theory [7] contains a multitude of examples of logical systems formalised as institutions both from computing science and from mathematical logic. One of the most representative logical systems is (many-sorted) first-order logic, which was first presented as an institution in [9].

Many-sorted first-order logic with equality (FOL)

Signatures. A (many-sorted) first-order signature is a tuple (S, F, P) consisting of

- a set S of *sorts*,
- a family $F = \{F_{w \to s} \mid w \in S^*, s \in S\}$ of sets of *operation* symbols indexed by arities and sorts, where $F_{w \to s}$ denotes the set of operations with arity w and sort s (when the arity is empty, $F_{\lambda \to s}$ denotes the set of *constants* of sort s), and
- a family $P = \{P_w \mid w \in S^*\}$ of sets of S-sorted *relation* symbols indexed by arities. When P is empty, we call the pair (S, F) an *algebraic signature*.

Signature morphisms. Signature morphisms $\phi\colon (S, F, P) \to (S', F', P')$ reflect the structure of signatures and consist of

- a function $\phi^{\mathrm{st}}\colon S \to S'$ between the sets of sorts,
- a family of functions $\phi^{\mathrm{op}} = \{\phi^{\mathrm{op}}_{w \to s}\colon F_{w \to s} \to F'_{\phi^{\mathrm{st}}(w) \to \phi^{\mathrm{st}}(s)} \mid w \in S^*, s \in S\}$ between the sets of operation symbols, and
- a family of functions $\phi^{\mathrm{rel}} = \{\phi^{\mathrm{rel}}_{w}\colon P_w \to P'_{\phi^{\mathrm{st}}(w)} \mid w \in S^*\}$ between the sets of relation symbols.

Models. For every signature (S, F, P), a model M interprets each sort symbol s as a set M_s, called the *carrier set of sort s*, each operation symbol $\sigma \in F_{w \to s}$ as a function $M_\sigma\colon M_w \to M_s$, where $M_w = M_{s_1} \times \cdots \times M_{s_n}$ for $w = s_1 \ldots s_n$, with $s_1, \ldots, s_n \in S$, and each relation symbol $\pi \in P_w$ as a subset $M_\pi \subseteq M_w$. We will assume throughout this paper that every signature has at least one constant for every sort, in order to avoid empty interpretations of sorts and thus simplify the presentation of some results.

A homomorphism of (S, F, P)-models $h\colon M \to N$ is an indexed family of functions $\{h_s\colon M_s \to N_s \mid s \in S\}$ such that

- h is an (S, F)-algebra homomorphism: $h_s(M_\sigma(m)) = N_\sigma(h_w(m))$, for every $\sigma \in F_{w \to s}$ and every $m \in M_w$, where $h_w\colon M_w \to N_w$ denotes the canonical component-wise extension of h to w-tuples, i.e., $h_w(m_1, \ldots, m_n) = h_{s_1}(m_1), \ldots, h_{s_n}(m_n)$ for $w = s_1 \cdots s_n$ and $m_i \in M_{s_i}$ for $i = \overline{1, n}$, and
- $h_w(m) \in N_\pi$ if $m \in M_\pi$, i.e. $h_w(M_\pi) \subseteq N_\pi$, for every relation symbol $\pi \in P_w$.

Model reducts. For every signature morphism $\phi\colon \Sigma \to \Sigma'$, the *reduct* $M'\upharpoonright_\phi$ of a Σ'-model M' is defined as the Σ-model given by $(M'\upharpoonright_\phi)_x = M'_{\phi(x)}$ for every sort, function, or relation symbol x from the domain signature of ϕ. The reduct $h'\upharpoonright_\phi$ of a model homomorphism is also defined as $(h'\upharpoonright_\phi)_s = h'_{\phi(s)}$, for every sort $s \in S$.

Sentences. The sentences are usual first-order sentences built from equational and relational atoms by applying in an iterative manner Boolean connectives and first-order quantifiers. Given a signature (S, F, P), and a sort s, the set $T_{F,s}$ of F-terms of sort s is the least set such that $\sigma(t) \in T_{F,s}$, for all $\sigma \in F_{w\to s}$, and all tuples $t \in T_{F,w}$, where $T_{F,w} = T_{F,s_1} \times \cdots \times T_{F,s_n}$, and $w = s_1 \cdots s_n$. The set of (S, F, P)-sentences is the least set containing *equational atoms* $t = t'$ (for $t, t' \in T_{F,s}$) and *relational atoms* $\pi(t_1, \ldots, t_n)$ (where $\pi \in P_w$ and $t_1, \ldots, t_n \in T_{F,w}$) that is closed under

- *Boolean connectives*: for any (S, F, P)-sentences ρ_1 and ρ_2, the negation $\neg\rho_i$, the conjunction $\rho_1 \wedge \rho_2$, the disjunction $\rho_1 \vee \rho_2$, the implication $\rho_1 \to \rho_2$, and the equivalence $\rho_1 \leftrightarrow \rho_2$ are also (S, F, P)-sentences,

- *existential and universal quantification* over sets of first-order variables, which are triples $\langle x, s, (S, F, P)\rangle$ sometimes denoted by $x : s$, where x is the name of the variable, and $s \in S$ is its sort: for any $(S, F \uplus X, P)$-sentence ρ, $\exists X.\rho$, and $\forall X.\rho$ are (S, F, P)-sentences, where $(S, F \uplus X, P)$ denotes the extension of (S, F, P) with the elements of X as new symbols of constants.

Sentence translations. The sentence translation $\mathrm{Sen}(\phi)\colon \mathrm{Sen}(S, F, P) \to \mathrm{Sen}(S', F', P')$ along a signature morphism $\phi\colon (S, F, P) \to (S', F', P')$ is defined inductively on the structure of the sentences and renames the sorts, function, and relation symbols of (S, F, P) with symbols of (S', F', P') according to ϕ. For terms, we define the extension of ϕ as $\phi^{\mathrm{tm}}(\sigma(t_1, \ldots, t_n)) = \phi^{\mathrm{op}}(\sigma)(\phi^{\mathrm{tm}}(t_1), \ldots, \phi^{\mathrm{tm}}(t_n))$. Then,

- $\mathrm{Sen}(\phi)(t = t') = (\phi^{\mathrm{tm}}(t) = \phi^{\mathrm{tm}}(t'))$ for equations,

- $\mathrm{Sen}(\phi)(\pi(t_1, \ldots, t_n)) = \phi^{\mathrm{rel}}(\pi)(\phi^{\mathrm{tm}}(t_1), \ldots, \phi^{\mathrm{tm}}(t_n))$ for relational atoms,

- $\mathrm{Sen}(\phi)(\rho_1 \wedge \rho_2) = \mathrm{Sen}(\phi)(\rho_1) \wedge \mathrm{Sen}(\phi)(\rho_2)$, and similarly for the rest of the Boolean connectives, and

- $\mathrm{Sen}(\phi)(\exists X.\rho) = \exists X^\phi.\mathrm{Sen}(\phi^X)(\rho)$ for every finite set of variables X, every $(S, F \uplus X, P)$-sentence ρ, where $X^\phi = \{x : \phi^{\mathrm{st}}(s) \mid x : s \in X\}$ and $\phi^X\colon (S, F \uplus X, P) \to (S', F' \uplus X^\phi, P')$ extends ϕ canonically. We define the translation of universally quantified sentences in a similar manner.

Satisfaction. The satisfaction between models and sentences is the usual Tarskian satisfaction defined inductively on the structure of sentences and based on the valuation of terms in models. Given a model M of a fixed arbitrary signature (S, F, P)

- for equational atoms: $M \models t = t'$ if $M_t = M_{t'}$ [1],

[1] where M_t and $M_{t'}$ are the interpretations of t and t' in M

- for relational atoms: $M \models \pi(t)$ if $M_t \in M_\pi$,
- $M \models \neg\rho$ if and only if $M \not\models \rho$,
- $M \models \rho_1 \wedge \rho_2$ if and only if $M \models \rho_1$ and $M \models \rho_2$,
- $M \models \rho_1 \vee \rho_2$ if and only if $M \models \rho_1$ or $M \models \rho_2$,
- $M \models \rho_1 \to \rho_2$ if and only if $M \models \rho_2$ whenever $M \models \rho_1$,
- $M \models \exists X.\rho$ if there exists an expansion M' of M along the signature inclusion $(S, F, P) \hookrightarrow (S, F \uplus X, P)$ such that $M' \models \rho$, and
- $M \models \forall X.\rho$ if and only if $M \models \neg\exists X.\neg\rho$.

Presentations

The presentations over an institution represent one of the simplest forms of specifications over that logic being formed merely of a signature and a (usually finite) set of its sentences. We will use presentations in our thesis to encode reachability logic into first-order logic in Chapter 4.3.

Definition 2. A *presentation* of an institution $\mathbf{I} = (\mathbb{S}\mathrm{ig}, \mathrm{Sen}, \mathrm{Mod}, \models)$ is a pair (Σ, E) consisting of a signature Σ and a set E of Σ-sentences.

The presentations of an institution form a category $\mathbb{P}\mathrm{res}$ whose arrows $\phi\colon (\Sigma, E) \to (\Sigma', E')$ are signature morphisms $\phi\colon \Sigma \to \Sigma'$ such that $E' \models \phi(E)$. By extending the sentence functor, the model functor and the satisfaction relation from the signatures of $\mathbf{I}$ to presentations we obtain an institution $\mathbf{I}^{\mathrm{pres}} = (\mathbb{P}\mathrm{res}, \mathrm{Sen}^{\mathrm{pres}}, \mathrm{Mod}^{\mathrm{pres}}, \models^{\mathrm{pres}})$ of $\mathbf{I}$ presentations.

Model amalgamation

Model amalgamation is one of the most important properties of an institution, with numerous applications in the context of institution-independent model theory [7] and module algebra [11]. In particular, as regards our work, model amalgamation will prove to be crucial in adding quantifiers over an arbitrary institution.

Essentially, model amalgamation allows us to combine models of different signatures whenever they are compatible with respect to a common sub-signature. Most logical systems of interest for mathematics and specification theory have model amalgamation, including the examples considered in this paper.

Definition 3. In any institution, a commuting square of signature morphisms

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Sigma_1 \arrow[d, "\theta_1"] \\
\Sigma_2 \arrow[r, "\theta_2"'] & \Sigma'
\end{tikzcd}$$

is a *weak amalgamation square* if and only if, for each Σ_1-model M_1 and Σ_2-model M_2 such that $\mathrm{Mod}(\varphi_1)(M_1) = \mathrm{Mod}(\varphi_2)(M_2)$, there exists a Σ'-model M', called an *amalgamation* of M_1 and M_2, such that $\mathrm{Mod}(\theta_1)(M') = M_1$ and $\mathrm{Mod}(\theta_2)(M') = M_2$. When M' is required to be unique, the square is called an *amalgamation square*.

We say that an institution has (weak) model amalgamation if and only if each pushout square of signature morphisms is a (weak) amalgamation square.Therefore, in order to have model amalgamation, the square of signature morphisms must not identify entities of Σ_1 and Σ_2 that do not come from Σ via the signature morphisms φ_1 and φ_2. Moreover, to guarantee the uniqueness of the amalgamation, Σ' must contain only entities that come from Σ_1 or Σ_2.

Moving between institutions

When describing and reasoning about properties of highly complex structures or systems, it is often desirable to use different formalisms for different tasks. Consequently, in order to use institutions as formalisations of logical systems in a heterogeneous setting, one needs to define formally a notion of map between institutions. Several concepts have been defined over the years, including semi-morphisms, morphisms, and comorphisms, some of which can be found in [22]. In our work, we focus only on comorphisms [14, 23], which reflect the intuition of embedding simpler institutions into more complex ones.

Definition 4. Given two institutions $\mathbf{I}$ and $\mathbf{I}'$, a *comorphism* $(\Phi, \alpha, \beta)\colon \mathbf{I} \to \mathbf{I}'$ consists of

- a *signature functor* $\Phi\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ig}'$,
- a natural transformation $\alpha\colon \mathrm{Sen} \Rightarrow \Phi \,;\mathrm{Sen}'$, and
- a natural transformation $\beta\colon \Phi^{\mathrm{op}} \,;\mathrm{Mod}' \Rightarrow \mathrm{Mod}$

such that the following *satisfaction condition* holds for any $\mathbf{I}$-signature Σ, $\Phi(\Sigma)$-model M', and Σ-sentence ρ:

$$M' \models^{\mathbf{I}'}_{\Phi(\Sigma)} \alpha_\Sigma(\rho) \quad \text{iff} \quad \beta_\Sigma(M') \models^{\mathbf{I}}_{\Sigma} \rho.$$

Example. There exists a comorphism that allows us to embed into **FOL** the first-order equational logic (**FOEQL**), i.e. the fragment of **FOL** obtained by discarding the relation symbols and their interpretations. We define $(\Phi, \alpha, \beta)\colon \mathbf{FOEQL} \to \mathbf{FOL}$ as follows:

- $\Phi\colon \mathbb{S}\mathrm{ig}^{\mathbf{FOEQL}} \to \mathbb{S}\mathrm{ig}^{\mathbf{FOL}}$ is the embedding of algebraic signatures into the category of first-order signatures,
- for every signature $\Sigma \in |\mathbb{S}\mathrm{ig}^{\mathbf{FOEQL}}|$, $\alpha_\Sigma\colon \mathrm{Sen}^{\mathbf{FOEQL}}(\Sigma) \to \mathrm{Sen}^{\mathbf{FOL}}(\Phi(\Sigma))$ is an inclusion of sets,
- for every signature $\Sigma \in |\mathbb{S}\mathrm{ig}^{\mathbf{FOEQL}}|$, $\beta_\Sigma\colon \mathrm{Mod}^{\mathbf{FOL}}(\Phi(\Sigma)) \to \mathrm{Mod}^{\mathbf{FOEQL}}(\Sigma)$ is the identity functor.

2.2 $\mathbb{K}$ Semantic Framework

The $\mathbb{K}$ framework [19] is an executable semantic framework based on rewriting and used for defining programming languages, computational calculi, type systems and formal analysis tools that was developed as an alternative to the existing operational-semantics (SOS) frameworks and has been employed to define actual programming languages such as C, Python, Java, and Verilog.

Although based on rewriting, $\mathbb{K}$ could be regarded as a notation for rewriting logic, along the majority of semantic frameworks – big-step (natural) semantics, small-step SOS, modular SOS –, only if its concurrent semantics were to be ignored. Its concurrency properties make it rather hard to translate faithfully (step for step) to rewriting logic. In defining semantics for programming languages, $\mathbb{K}$ handles cell-like structures named *configurations* and relies on computational structures – *computations* – to model transitions between these configurations by applying local rewriting *rules*. Computations are sequences of terms over the abstract syntax of the language – seen as computational tasks – that are usually used for managing the evaluation strategies and the sequential part of the formalised language. The configurations are multisets of nested cells that allow us to manage concurrency and modularity issues. The rules of $\mathbb{K}$ have the advantage of being more concise and more modular than the regular rewrite rules, by enabling us to specify only the fragments of configurations that we need to change.

The $\mathbb{K}$ tool allows not only the development of modular, executable definitions of languages, but it also facilitates language design as it supports testing and exploring behaviour in an exhaustive manner. The tool has been also used for defining type checkers and type inferencers, for studying runtime verification techniques, and for developing a model-checking tool based on predicate abstraction and a program verification tool that uses Hoare-like assertions written in matching logic.

To briefly describe the $\mathbb{K}$ semantic framework we consider the following definition of the IMP language [1], an elementary example of an imperative programming language.

Listing 2.1: The IMP programming language ($\mathbb{K}$)

```
module IMP−SYNTAX
  syntax AExp ::= Int | Id
                | AExp "/" AExp          [left, strict]
                ⟩ AExp "+" AExp          [left, strict]
                | "(" AExp ")"           [bracket]
  syntax BExp ::= Bool
                | AExp "⟨=" AExp         [seqstrict]
                | "!" BExp               [strict]
                ⟩ BExp "&&" BExp         [left, strict(1)]
                | "(" BExp ")"           [bracket]
  syntax Block ::= "{" "}"
                | "{" Stmt "}"
  syntax Stmt ::= Block
                | Id "=" AExp ";"        [strict(2)]
```

```
                    | "if" "(" BExp ")"
                      Block "else" Block          [strict (1)]
                    | "while" "(" BExp ")" Block
                    ⟩ Stmt Stmt                   [left]
  syntax Pgm ::= "int" Ids ";" Stmt
  syntax Ids ::= List{Id,","}
endmodule

module IMP
  imports IMP−SYNTAX
  syntax KResult ::= Int | Bool
  configuration ⟨t⟩
      ⟨k⟩ $PGM:Pgm ⟨/k⟩
      ⟨state⟩ .Map ⟨/state⟩
                    ⟨/t⟩
// AExp
  rule ⟨k⟩ X:Id  =>  I ⋯⟨/k⟩ ⟨state⟩⋯ X |−> I ⋯⟨/state⟩
  rule I1:Int / I2:Int  =>  I1 /Int I2  when I2 =/=Int 0
  rule I1:Int + I2:Int  =>  I1 +Int I2
// BExp
  rule I1:Int ⟨= I2:Int  =>  I1 ⟨=Int I2
  rule ! T:Bool  =>  notBoolT
  rule true && B  => B
  rule false && _ =>  false
// Block
  rule {}  =>  .
  rule {S}  =>  S
// Stmt
  rule ⟨k⟩ X = I:Int;  =>  . ⋯⟨/k⟩
  ⟨state⟩⋯ X |−> (_  =>  I) ⋯⟨/state⟩
  rule S1 S2  =>  S1 ↝ S2
  rule if (true)  S else _  =>  S
  rule if ( false ) _ else S  =>  S
  rule while (B) S  =>  if (B) {S while (B) S} else {}
// Pgm
  rule ⟨k⟩ int (X:Id,Xs:Ids  =>  Xs);_ ⟨/k⟩
  ⟨state⟩ Rho:Map (.  =>  X |−>0) ⟨/state⟩
  when notBool(X in keys(Rho))
  rule int .Ids; S  =>  S
endmodule
```

The 𝕂 definitions are written in machine-readable ASCII which the tool subsequently translates for execution into Maude rewrite theories. Language features are grouped in 𝕂 using importable modules: language definitions must contain at least one module, but it is considered a good practice (and helpful for the parser) to separate the formal syntax from the semantic declaration. This is why we consider two distinct modules, IMP-SYNTAX and IMP, in order to define the IMP language.

Syntactic structures are defined in the 𝕂 tool using a variant of the Backus-Naur Form notation: we introduce nonterminals for arithmetic expressions (AExp), Boolean expressions (BExp), statements (Stmt), blocks of statements (Block), identifiers (Ids) and programs (Pgm). It should be noted that 𝕂 provides a number of built-in syntactic categories (*Int*, *Id*, *Bool*, etc.) and corresponding semantic operations that we take advantage of throughout the formalization of our simple imperative language.

In order to specify *language semantics* in 𝕂, one needs to provide *evaluation strategies* (by annotating the syntax declarations with strictness-constraining attributes), to define the nested structure of the configuration holding the state of the executed program, and to give rules that describe reconfigurations – the transitions between configurations. In our example, we annotate the declaration of some operations with the **strict** and **seqstrict** attributes in order to specify that the arguments of the operation must be evaluated before giving semantics to the construct itself. The terms that represent values, or results, are distinguished by declaring them of sort *KResult*. The sequencing of evaluation is implemented by computations that extend the syntax of the language with a list structure through the separator ~> ("followed by"). The initial computation contains only one task – the program to be executed –, that is translated into a sequence of tasks by applying rules corresponding to strictness annotations. The resulting list of computational tasks is then processed in order. Computations define a new sort, called K, that is also a suprasort of *KResult*.

We represent the state of a running program with a configuration consisting of nested, labelled cells containing various data structures such as sets, lists, and maps. We define the configurations using an XML-like notation, where the labels of the cells are tag names and the contents are enclosed by tags. In our example, we define the the structure of IMP configurations and we specify the initial configuration by introducing a cell labelled `k` that contains the program we are executing (denoted by the variable $PGM of sort K that is being initialized at the beginning of the runtime with the original IMP program), and a cell `state` holding a mapping between identifier names and values. This configuration structure will serve as a backbone for the local semantic rules discussed in what follows.

The semantic rules describe the evolution of the configuration, or how a subterm of the configuration is rewritten to another term: any term that matches the left-hand side of a rule is replaced with the right-hand side thereof. The reconfigurations consist, in our case, in the advances of the computation and the changes of the state. The local aspect of the rewriting in 𝕂 allows us to omit unchanging fragments of terms from the semantic rules. Our example reflects local rewriting through the multiple occurrences in a rule of the rewrite symbol =>. Moreover, the rules may include only the relevant cells and not the entire configuration, the rest of the context being inferred automatically through a technique called *configuration abstraction*. Boolean predicates can be added to the semantic rules, defining in this way side conditions.

In the following sections we introduce two logics in order to formalize the 𝕂 framework. *Matching logic* will be used to define the syntactic constructs – the syntax of the specified programming languages – and the patterns matched in the semantic rules, and to partially capture the semantics of the programming languages by defining the states of the running programs. Subsequently, we build *reachability logic* upon matching logic to capture the semantic rules. Its sentences, defined over the signatures of matching logic, correspond to the rules in the $\mathbb{K}$ modules, while the models represent implementations of programming languages. The language definitions written in $\mathbb{K}$ will thus be seen, leaving aside some parsing instructions without logical interpretation, as formal specifications over reachability logic.

3 Matching Logic

The generality of institutions allowed them to accommodate a great variety of logical systems. As a downside however, and as it would be expected for such abstract notions, certain logics cannot be captured in full detail by institutions; that is, by considering them only as institutions we lose precious information. An example is computation-tree logic, for which we lose the distinction between state and path sentences (which, in fact, do not belong to the sentences of computation-tree logic formalised as an institution).

Matching logic falls in the same category, but this time, we lose the sorts of patterns and the states of models. To palliate this, we extend institutions with notions of classes (for sorts) and stratification of models (for states). The end result – the concept of stratified institution with classes – is obtained as a combination of the institutions with classes described in Definition 5 with the stratified institutions introduced in [2].

3.1 Stratified Institutions with Classes

Definition 5. An *institution with classes* is a tuple $(\mathbb{S}\mathrm{ig}, \mathrm{Cls}, \mathrm{Sen}, \kappa, \mathrm{Mod}, \models)$, where

- $(\mathbb{S}\mathrm{ig}, \mathrm{Sen}, \mathrm{Mod}, \models)$ is an institution,
- $\mathrm{Cls}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{et}$ is a functor giving for each signature a set whose elements are called *classes* of that signature, and
- $\kappa\colon \mathrm{Sen} \Rightarrow \mathrm{Cls}$ is a natural transformation associating a class to each sentence.

We will also use the notation $\mathrm{Sen}(\Sigma)_c$ for $\kappa^{-1}(c)$, $c \in \mathrm{Cls}(\Sigma)$ to denote the set of Σ-sentences of class c.

Example. An immediate example of an institution with classes is the atomic fragment of equational first-order logic. In this case, Cls is the forgetful functor that maps every algebraic signature (S, F) to its underlying set of sorts S, and $\kappa_{(S,F)}$ is the function that assigns to each equational atom $t = t'$ the common sort of t and t'.

Definition 6. A *stratified institution with classes* $\underline{\mathrm{I}} = (\mathbb{S}\mathrm{ig}, \mathrm{Cls}, \mathrm{Sen}, \kappa, \mathrm{Mod}, [\![\_]\!], \models)$ consists of:

- a category $\mathbb{S}\mathrm{ig}$ of *signatures* and *signature morphisms*,
- a *class functor* $\mathrm{Cls}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{et}$, giving for every signature a set of *classes*,
- a *sentence functor* $\mathrm{Sen}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{et}$, defining for every signature Σ a set of *sentences*,

- a natural transformation $\kappa\colon \mathrm{Sen} \Rightarrow \mathrm{Cls}$, associating a class to each sentence,
- a *model functor* $\mathrm{Mod}\colon \mathbb{S}\mathrm{ig}^{op} \to \mathbb{C}\mathrm{at}$, defining a category of *models* for every signature,
- a *stratification* $\llbracket\_\rrbracket$ giving
  - for every signature Σ, a family of functors $\llbracket\_\rrbracket_{\Sigma,c}\colon \mathrm{Mod}(\Sigma) \to \mathbb{S}\mathrm{et}$, indexed by classes $c \in \mathrm{Cls}(\Sigma)$, and
  - for every signature morphism $\phi\colon \Sigma \to \Sigma'$, a functorial family (see Remark 1 below) of natural transformations $\llbracket\_\rrbracket_{\phi,c}\colon \llbracket\_\rrbracket_{\Sigma',\mathrm{Cls}(\phi)(c)} \Rightarrow \mathrm{Mod}(\phi)\mathbin{;}\llbracket\_\rrbracket_{\Sigma,c}$, indexed by classes $c \in \mathrm{Cls}(\Sigma)$, such that $\llbracket M'\rrbracket_{\phi,c}$ is surjective for every $M' \in |\mathrm{Mod}(\Sigma')|$, and
- a satisfaction relation between models and sentences, parameterised by model states and classes: $M \models^{m}_{\Sigma,c} \rho$, where Σ is a signature, $c \in \mathrm{Cls}(\Sigma)$, $M \in |\mathrm{Mod}(\Sigma)|$, $m \in \llbracket M\rrbracket_{\Sigma,c}$, and $\rho \in \mathrm{Sen}(\Sigma)_c$

such that the following properties are equivalent:

i. $\mathrm{Mod}(\phi)(M') \models^{\llbracket M'\rrbracket_{\phi,c}(m')}_{\Sigma,c} \rho$

ii. $M' \models^{m'}_{\Sigma',\mathrm{Cls}(\phi)(c)} \mathrm{Sen}(\phi)(\rho)$,

for every signature morphism $\phi\colon \Sigma \to \Sigma'$, every class $c \in \mathrm{Cls}(\Sigma)$, every model $M' \in |\mathrm{Mod}(\Sigma')|$, every state $m' \in \llbracket M'\rrbracket_{\Sigma',\mathrm{Cls}(\phi)(c)}$, and every sentence $\rho \in \mathrm{Sen}(\Sigma)_c$.

Remark 1. The functoriality of the stratifications $\llbracket\_\rrbracket_{\phi,c}\colon \llbracket\_\rrbracket_{\Sigma',\mathrm{Cls}(\phi)(c)} \Rightarrow \mathrm{Mod}(\phi);\llbracket\_\rrbracket_{\Sigma,c}$ means that for every signature morphisms $\phi\colon \Sigma \to \Sigma'$, $\phi'\colon \Sigma' \to \Sigma''$, every Σ''-model M'', and every class $c \in \mathrm{Cls}(\Sigma)$, $\llbracket M''\rrbracket_{\phi;\phi',c} = \llbracket M''\rrbracket_{\phi',\phi(c)}\mathbin{;}\llbracket M''\!\upharpoonright_{\phi'}\rrbracket_{\phi,c}$.

$$\begin{tikzcd}
\llbracket M''\rrbracket_{\Sigma'',\phi'(\phi(c))} \arrow[r, "\llbracket M''\rrbracket_{\phi',\phi(c)}"] \arrow[rr, bend right, "\llbracket M''\rrbracket_{\phi;\phi',c}"'] & \llbracket M''\!\upharpoonright_{\phi'}\rrbracket_{\Sigma',\phi(c)} \arrow[r, "\llbracket M''\upharpoonright_{\phi'}\rrbracket_{\phi,c}"] & \llbracket (M''\!\upharpoonright_{\phi'})\!\upharpoonright_{\phi}\rrbracket_{\Sigma,c}
\end{tikzcd}$$

Proposition 1. Every stratified institution with classes $(\mathbb{S}\mathrm{ig}, \mathrm{Cls}, \mathrm{Sen}, \kappa, \mathrm{Mod}, \llbracket\_\rrbracket, \models)$ determines an institution whose category of signatures is $\mathbb{S}\mathrm{ig}$, sentence functor is Sen, model functor is Mod, and satisfaction relation $\models_\Sigma \subseteq |\mathrm{Mod}(\Sigma)| \times \mathrm{Sen}(\Sigma)$ is defined, for every signature $\Sigma \in |\mathbb{S}\mathrm{ig}|$, as follows:

$$M \models_\Sigma \rho \quad \text{iff} \quad M \models^{m}_{\Sigma,c} \rho \text{ for every } m \in \llbracket M\rrbracket_{\Sigma,c}, \text{ where } c = \kappa_\Sigma(\rho)\ .$$

Proof. We have to prove that the satisfaction condition holds for every signature morphism $\phi\colon \Sigma \to \Sigma'$, every Σ'-model M', and every Σ-sentences ρ, that is

$$M' \models_{\Sigma'} \mathrm{Sen}(\phi)(\rho) \quad \text{iff} \quad \mathrm{Mod}(\phi)(M') \models_\Sigma \rho.$$

We denote by c the class $\kappa_\Sigma(\rho)$ of the sentence ρ, and by c' the class $\kappa_{\Sigma'}(\mathrm{Sen}(\phi)(\rho))$ of the sentence $\mathrm{Sen}(\phi)(\rho)$. We recall that by the naturality of κ, $c' = \mathrm{Cls}(\phi)(c)$.

By the definition of the satisfaction relation of the derived institution,

$$M' \models_{\Sigma'} \mathrm{Sen}(\phi)(\rho) \quad \text{iff} \quad M' \models^{m'}_{\Sigma',c'} \mathrm{Sen}(\phi)(\rho) \text{ for every } m' \in [\![M']\!]_{\Sigma',c'}.$$

Since the satisfaction condition holds for ϕ as a signature morphism in a stratified institution with classes, it follows that

$$M' \models^{m'}_{\Sigma',c'} \mathrm{Sen}(\phi)(\rho) \quad \text{iff} \quad \mathrm{Mod}(\phi)(M') \models^{[\![M']\!]_{\phi,c}(m')}_{\Sigma,c} \rho \text{ for every } m' \in [\![M']\!]_{\Sigma',c'}.$$

Hence, by the surjectivity of $[\![M']\!]_{\phi,c}$, we obtain

$$\mathrm{Mod}(\phi)(M') \models^{[\![M']\!]_{\phi,c}(m')}_{\Sigma,c} \rho \quad \text{iff} \quad \mathrm{Mod}(\phi)(M') \models^{m}_{\Sigma,c} \rho \text{ for every } m \in [\![\mathrm{Mod}(\phi)(M')]\!]_{\Sigma,c}.$$

To conclude the proof notice that the right-hand side of the above equivalence means that $\mathrm{Mod}(\phi)(M') \models_\Sigma \rho$. □

Notation 2. Let $\flat\mathbf{I}$ denote the institution obtained by applying Proposition 1 to a stratified institution with classes $\underline{\mathrm{I}}$.

Computation-tree logic ($\underline{\mathrm{CTL}}$)

In this section, we formalise computation-tree logic as a first example of stratified institution with classes. Similarly to other temporal logics, the usual presentation of $\underline{\mathrm{CTL}}$ is based on propositional logic. $\underline{\mathrm{CTL}}$ inherits the signatures of propositional logic, which means that the category of its *signatures* is $\mathbb{S}\mathrm{et}$.

$\underline{\mathrm{CTL}}$ formulae can express properties of a state or a path (infinite sequence of states) of a transition system (defined below), being *classified* into state and path formulae: $\mathrm{Cls}(\Sigma) = \{\mathit{state}, \mathit{path}\}$, for every $\Sigma \in |\mathbb{S}\mathrm{ig}| = |\mathbb{S}\mathrm{et}|$.

We define the functor Sen, and the natural transformation κ simultaneously, describing the *sentences* of a signature and their classes:

- the atomic propositions $a \in \Sigma$ are sentences of class *state*,
- init is a proposition of class *state*,
- $\varphi_1 \wedge \varphi_2$ is a sentence of class *state*, for every φ_1, φ_2 sentences of class *state*,
- $\neg\varphi$ is a sentence of class *state*, for every sentence φ of class *state*,
- $\exists\pi, \forall\pi$ are sentences of class *state*, for every sentence π of class *path*,
- $\bigcirc\varphi$ is a sentence of class *path*, for every sentence φ of class *state*,
- $\varphi_1 \mathsf{U} \varphi_2$ is a sentence of class *path*, for every φ_1, φ_2 sentences of class *state*.

The *models* of a $\underline{\text{CTL}}$ signature Σ are transition systems $\text{TS} = (S, \rightarrow, I, \Sigma, L)$, where S is a set of states, $\rightarrow \subseteq S \times S$ is a transition relation, $I \subseteq S$ is a set of initial states, and $L\colon S \rightarrow 2^{\Sigma}$ is a labelling function. We define a transition system morphism $h\colon (S, \rightarrow, I, \Sigma, L) \rightarrow (S', \rightarrow', I', \Sigma, L')$ as a function $h\colon S \rightarrow S'$ such that $h(I) \subseteq I'$, $h(\rightarrow) \subseteq \rightarrow'$, and $L(s) = L'(h(s))$, for every $s \in S$.

The *stratification* of models is defined as follows:

- $[\![\text{TS}]\!]_{\Sigma,state}$ is the set S of states of TS,
- $[\![\text{TS}]\!]_{\Sigma,path}$ is the set of paths of TS, that is sequences $s_0, s_1, \ldots \in S^{\omega}$, such that $s_i \rightarrow s_{i+1}$ for $i \in \omega$.

For every signature morphism $\phi\colon \Sigma \rightarrow \Sigma'$, the components of the natural transformations $[\![\_]\!]_{\phi,c}$ are identity functions:

$$[\![\text{TS}']\!]_{\phi,state}(s') = s', \quad [\![\text{TS}']\!]_{\phi,path}(p') = p',$$

for every $s' \in [\![\text{TS}']\!]_{\Sigma',state}$, $p' \in [\![\text{TS}']\!]_{\Sigma',path}$, and $\text{TS}' \in |\text{Mod}(\Sigma')|$.

The *satisfaction relation* between models and sentences is given by:

- $\text{TS} \models^{s}_{\Sigma,state} \rho$ iff
  - $\rho \in L(s)$, for $\rho \in \Sigma$
  - $s \in I$, for $\rho = \text{init}$
  - $\text{TS} \not\models^{s}_{\Sigma,state} \varphi$, for $\rho = \neg\varphi$
  - $\text{TS} \models^{s}_{\Sigma,state} \varphi_1$ and $\text{TS} \models^{s}_{\Sigma,state} \varphi_2$, for $\rho = \varphi_1 \wedge \varphi_2$
  - there is $p = s_0, s_1, \ldots \in [\![\text{TS}]\!]_{\Sigma,path}$ with $s_0 = s$ such that $\text{TS} \models^{p}_{\Sigma,path} \pi$, for $\rho = \exists\pi$,
- $\text{TS} \models^{p}_{\Sigma,path} \rho$ iff
  - $\text{TS} \models^{s_1}_{\Sigma,state} \varphi$, for $\rho = \bigcirc\varphi$
  - there exists an index j such that $\text{TS} \models^{s_j}_{\Sigma,state} \varphi_2$, and for all $i < j$, $\text{TS} \models^{s_i}_{\Sigma,state} \varphi_1$, for $\rho = \varphi_1 \mathsf{U} \varphi_2$,

for every signature Σ, every state $s \in S$, every path $p = s_0, s_1, \ldots$, every sentence ρ and every model $\text{TS} \in |\text{Mod}(\Sigma)|$.

3.2 Matching Logic

The original notion of matching logic developed in [17] can be described as a stratified institution with classes $\underline{\text{ML}}$ as follows.

Signatures. The signatures are algebraic signatures, that is pairs (S, F), where S is a set of sort names, and F is a family of sets $F_{w \to s}$ of operation symbols of arity $w \in S^*$ and sort $s \in S$. For signatures $\Sigma = (S, F)$, $\Sigma' = (S', F')$, a signature morphism $\phi \colon \Sigma \to \Sigma'$ is a pair $(\phi^{\mathrm{st}}, \phi^{\mathrm{op}})$, where $\phi^{\mathrm{st}} \colon S \to S'$ is a function, and ϕ^{op} is a family of functions that respect the arities and the sorts of operation symbols in Σ, that is $\phi^{\mathrm{op}} = \{\phi^{\mathrm{op}}_{w \to s} \colon F_{w \to s} \to F'_{\phi^{\mathrm{st}}(w) \to \phi^{\mathrm{st}}(s)} \mid w \in S^*, s \in S\}$.

Example. Let us consider the specification of the IMP programming language exemplified in Listing 2.1. The signature of the IMP-SYNTAX module is obtained by adding to the built-in syntactic categories and their corresponding semantic operations new sorts and operation symbols introduced by the **syntax** keyword. For example, in the fragment of the syntax module below, the AExp sort is introduced as a suprasort of *Int* and *Id* sorts[1]. Addition and division are defined as binary operations with arguments and results of sort AExp, while bracketing is defined as a unary operation of the same sort. We note that only the **bracket** attribute has an effect on the signature of the specification as it determines the removal of its corresponding symbol of operation from the signature. The **left** and **strict** attributes are only used in parsing programs and in refining the evaluation strategy (by sequencing computational tasks), and thus, they do not play a role in defining the signature.

Listing 3.1: The IMP programming language – AExp syntax

```
syntax AExp ::= Int | Id
              | AExp "/" AExp        [left, strict]
              ⟩ AExp "+" AExp        [left, strict]
              | "(" AExp ")"         [bracket]
```

The signature of the fragment in Listing 3.1 is defined as

$$\mathrm{AExp}^{\mathbb{S}\mathrm{ig}} = (S \cup S_{\text{BUILT-IN}}, F \cup F_{\text{BUILT-IN}}),$$

where $S_{\text{BUILT-IN}}$ and $F_{\text{BUILT-IN}}$ are the built-in sorts and operations, $S = \{\mathrm{AExp}\}$ and $F_{AExp\,AExp \to AExp} = \{\_+\_,\ \_/\_\}$.

Similarly, the signature of the IMP module is obtained from the signature of the imported moduleIMP-SYNTAX, extending its signature through the addition of the sorts T, K, State and KResult and the operations $\langle k \rangle\_\langle/k\rangle \in F_{\mathrm{Pgm} \to \mathrm{K}}$, $\langle \mathrm{state} \rangle\_\langle/\mathrm{state}\rangle \in F_{\mathrm{Map} \to \mathrm{State}}$ and $\langle \mathrm{t} \rangle\_\langle/\mathrm{t}\rangle \in F_{\mathrm{K\,State} \to \mathrm{T}}$ introduced by the keyword **configuration**.

Classes of a signature. Every algebraic signature (S, F) determines (through the functor Cls) the set of classes S. This means that for matching logic, classes are simply sorts, just as in the case of the atomic equational first-order logic. Similarly, every morphism $\phi \colon (S, F) \to (S', F')$ determines a translation of classes $\phi^{\mathrm{st}} \colon S \to S'$.

[1] For simplicity, the formalism we used in this paper does not take into account the subsorting relation. We could further include subsorts following ideas developed for order-sorted equational logic [10].

Sentences. The sentences (or *patterns*) in ML of given sorts are defined as follows: for every signature Σ, $\mathrm{Sen}(\Sigma)$ is the least set that contains basic patterns (terms over Σ) and that is closed under the Boolean connectives $\neg, \wedge$, and the existential quantifier $\exists$.

- For each sort $s \in S$, the basic patterns of sort s are terms in $(T_\Sigma)_s$, where $(T_\Sigma)_s$ is the least set such that $\sigma(t_1, \ldots, t_n) \colon s \in (T_\Sigma)_s$ for all operation symbols $\sigma \in F_{s_1 \ldots s_n \to s}$ and for all terms $t_i \in (T_\Sigma)_{s_i}$.
- For every pattern π of sort s, $\neg\pi$ is a pattern of sort s.
- For every two patterns π_1, π_2 of sort s, $\pi_1 \wedge \pi_2$ is a pattern of the common sort s.
- For every variable x of sort s (defined formally as a tuple $\langle x, s, \Sigma \rangle$, where x is the name of the variable, and s is its sort), and every pattern $\pi \in \mathrm{Sen}(S, F \uplus \{x \colon s\})$, $\exists x \colon s.\pi$ is a pattern in $\mathrm{Sen}(\Sigma)$.

The sentence translation along a signature morphism $\phi \colon \Sigma \to \Sigma'$ is defined inductively, extending the translation of Σ-terms to Σ'-terms generated by ϕ – the family of functions $\phi_s^{\mathrm{tm}} \colon (T_\Sigma)_s \to (T_{\Sigma'})_{\phi^{\mathrm{st}}(s)}$ that map $\sigma(t_1, \ldots, t_n) \colon s$ to $\phi^{\mathrm{op}}(\sigma)(\phi^{\mathrm{tm}}(t_1), \ldots, \phi^{\mathrm{tm}}(t_n)) \colon \phi^{\mathrm{st}}(s)$, for every $\sigma \in F_{s_1 \ldots s_n \to s}$, and $t_i \in (T_\Sigma)_{s_i}$:

- $\mathrm{Sen}(\phi)(\pi) = \phi_s^{\mathrm{tm}}(\pi)$, for every basic pattern $\pi \in (T_\Sigma)_s$,
- $\mathrm{Sen}(\phi)(\neg\pi) = \neg\mathrm{Sen}(\phi)(\pi)$, for every pattern π,
- $\mathrm{Sen}(\phi)(\pi_1 \wedge \pi_2) = \mathrm{Sen}(\phi)(\pi_1) \wedge \mathrm{Sen}(\phi)(\pi_2)$, for every two patterns π_1, π_2,
- $\mathrm{Sen}(\phi)(\exists x \colon s.\pi) = \exists x \colon \phi^{\mathrm{st}}(s).\mathrm{Sen}(\phi^x)(\pi)$, where ϕ^x is the canonical extension of ϕ that maps $x \colon s$ to $x \colon \phi^{\mathrm{st}}(s)$, just as in the case of first-order logic.

Example. We can give as examples of sentences of an ML-signature, the patterns matched in the $\mathbb{K}$ rules corresponding to the IMP programming language specification presented in Listing 2.1: I1 : *Int* + I2 : *Int*, I1 $+_{\mathrm{Int}}$ I2, ! T : *Bool*, *true* && B etc.

Classes of sentences. The class of a pattern is given by its sort through the natural transformation $\kappa \colon \mathrm{Sen} \Rightarrow \mathrm{Cls}$ that is defined inductively on the structure of sentences:

- $\kappa_{(S,F)}(\pi) = s$, for every basic pattern $\pi \in (T_\Sigma)_s$,
- $\kappa_{(S,F)}(\neg\pi) = \kappa_{(S,F)}(\pi)$, for every pattern π,
- $\kappa_{(S,F)}(\pi_1 \wedge \pi_2) = \kappa_{(S,F)}(\pi_1)$, for every two patterns π_1, π_2,
- $\kappa_{(S,F)}(\exists x \colon s.\pi) = \kappa_{(S,F \uplus \{x \colon s\})}(\pi)$, for every pattern π.

Models. The models of ML are *multialgebras* [12]. Multialgebras are generalisations of algebras having nondeterministic operations that return sets of possible values; that is, multialgebras interpret operation symbols from the carrier set of their arity to the powerset of the carrier set of their sort. For a signature $\Sigma = (S, F)$, a *multialgebra homomorphism* $h\colon M \to N$ is a family of functions indexed by the signature's sorts $\{h_s\colon M_s \to N_s \mid s \in S\}$, such that $h_s(M_\sigma(m_1, \ldots, m_n)) \subseteq N_\sigma(h_{s_1}(m_1), \ldots, h_{s_n}(m_n))$, for every $\sigma \in F_{s_1 \ldots s_n \to s}$ and every $m_i \in M_{s_i}$.

Stratification. The stratification of models is given, for every signature Σ and class s of Σ, by $[\![M]\!]_{\Sigma,s} = M_s$, and for every signature morphism $\phi\colon \Sigma \to \Sigma'$, class s of Σ and model M' of Σ', by $[\![M']\!]_{\phi,s}(m') = m'$, where $m' \in M'_{\phi^{\mathrm{st}(s)}}$.

Satisfaction relation. The satisfaction relation between a model and a sentence is based on the interpretation of patterns in models. For any multialgebra M, we define M_π, the interpretation of a pattern π in M, inductively as follows:

- for every basic pattern $\pi \in F_{\lambda \to s}$, M_π is the interpretation of constant π in M,
- $M_\pi = \bigcup\{M_\sigma(m_1, \ldots, m_n) \mid m_i \in M_{t_i}\}$, for every basic pattern $\pi = \sigma(t_1, \ldots, t_n)$,
- $M_\pi = M_s \setminus M_{\pi_1}$, for every pattern $\pi = \neg\pi_1$, where π_1 is a pattern of sort s,
- $M_\pi = M_{\pi_1} \cap M_{\pi_2}$, for every pattern $\pi = \pi_1 \wedge \pi_2$,
- $M_\pi = \bigcup\{(M, X)_{\pi_1} \mid X \subseteq M_t\}$, for every pattern $\pi = \exists x\!:\!t.\pi_1$, where π_1 is a pattern of sort s, and (M, X) is the expansion of M along the inclusion $(S, F) \subseteq (S, F \uplus \{x\!:\!t\})$ given by $(M, X)_x = X$.

We now have all the necessary concepts for defining the *satisfaction relation*:

$$M \models^m_{\Sigma,s} \pi \quad \text{iff} \quad m \in M_\pi.$$

Proposition 2 (Satisfaction condition)**.** The following properties are equivalent:

i. $\mathrm{Mod}(\phi)(M') \models^{[\![M']\!]_{\phi,s}(m')}_{\Sigma,s} \pi$

ii. $M' \models^{m'}_{\Sigma',\mathrm{Cls}(\phi)(s)} \mathrm{Sen}(\phi)(\pi)$,

for every signature morphism $\phi\colon \Sigma \to \Sigma'$, every sort $s \in \mathrm{Cls}(\Sigma)$, every multialgebra $M' \in \mathrm{Mod}(\Sigma')$, every state $m' \in [\![M']\!]_{\Sigma',\mathrm{Cls}(\phi)(s)}$, and every pattern π of sort s.

Proof. We rewrite the two properties as:

i. $[\![M']\!]_{\phi,s}(m') = m' \in (M'\!\upharpoonright_\phi)_\pi$

ii. $m' \in M'_{\mathrm{Sen}(\phi)(\pi)}$.

We therefore have to prove the equality between the interpretation of a pattern in the reduct of the model M' along the signature morphism ϕ and the interpretation of the translation of the pattern π along ϕ in M'. We show this by induction on the structure of the patterns:

- for every basic pattern $\pi \in F_{\lambda \to s}$

  $$(M'\restriction_{\phi})_{\pi} = M'_{\phi^{\mathrm{op}}_{\lambda\to s}(\pi)},$$

- for every basic pattern $\pi = \sigma(t_1, \ldots, t_n)$

  $$\begin{aligned}(M'\restriction_{\phi})_{\sigma(t_1,\ldots,t_n)} &= \bigcup\{(M'\restriction_{\phi})_{\sigma}(m_1,\ldots,m_n) \mid m_i \in M'\restriction_{\phi_{t_i}}\} \\ &= \bigcup\{M'_{\phi^{\mathrm{op}}(\sigma)}(m_1,\ldots,m_n) \mid m_i \in M'_{\phi^{\mathrm{tm}}(t_i)}\} \\ &= M'_{\phi^{\mathrm{op}}(\sigma(\phi^{\mathrm{tm}}(t_1),\ldots,\phi^{\mathrm{tm}}(t_n)))} \\ &= M'_{\phi^{\mathrm{tm}}(\sigma(t_1,\ldots,t_n))},\end{aligned}$$

- for every pattern $\pi = \neg\pi_1$ of sort s

  $$\begin{aligned}(M'\restriction_{\phi})_{\neg\pi_1} &= (M'\restriction_{\phi})_{s} \setminus (M'\restriction_{\phi})_{\pi_1} = M'_{\phi^{\mathrm{st}}(s)} \setminus M'_{\mathrm{Sen}(\phi)(\pi_1)} \\ &= M'_{\neg\mathrm{Sen}(\phi)(\pi_1)} = M'_{\mathrm{Sen}(\phi)(\neg\pi_1)},\end{aligned}$$

- for every pattern $\pi = \pi_1 \wedge \pi_2$

  $$\begin{aligned}(M'\restriction_{\phi})_{\pi_1\wedge\pi_2} &= (M'\restriction_{\phi})_{\pi_1} \cap (M'\restriction_{\phi})_{\pi_2} \\ &= M'_{\mathrm{Sen}(\phi)(\pi_1)} \cap M'_{\mathrm{Sen}(\phi)(\pi_2)} = M'_{\mathrm{Sen}(\phi)(\pi_1\wedge\pi_2)},\end{aligned}$$

- for every pattern $\pi = \exists x : t.\pi_1$

  $$(M'\restriction_{\phi})_{\exists x\,:\,t.\pi_1} = \bigcup\{(M'\restriction_{\phi}, X)_{\pi_1} \mid X \subseteq (M'\restriction_{\phi})_{s}\} = \bigcup\{(M'\restriction_{\phi}, X)_{\pi_1} \mid X \subseteq M'_{\phi^{\mathrm{st}}(t)}\}$$
  $$M'_{\mathrm{Sen}(\phi)(\exists x\,:\,t.\pi_1)} = M'_{\exists x\,:\,\phi^{\mathrm{st}}(t).\mathrm{Sen}(\phi^{x})(\pi_1)} = \bigcup\{(M', X)_{\mathrm{Sen}(\phi^{x})(\pi_1)} \mid X \subseteq M'_{\phi^{st}(t)}\}$$

  The conclusion follows from the induction hypothesis by noticing that $(M'\restriction_{\phi}, X) = (M', X)\restriction_{\phi^{x}}$. □

In order to formalise the $\mathbb{K}$ framework, we should interpret the variables in a deterministic manner. For example, in the specification of the IMP programming language, the variables in the patterns matched by the semantic rules have a deterministic interpretation: the variables I1, I2 and T of the patterns I1 : *Int* + I2 : *Int*, ! T : *Bool* are interpreted as sole elements of sort *Int* or *Bool* respectively, as opposed to the interpretation of variables in <u>ML</u> as sets of elements. In the section that follows, we present a different formalisation of matching logic that takes into consideration the deterministic aspect of the interpretation of variables.

$\underline{\mathrm{ML}}^+$

We refine the above definition of matching logic $\underline{\mathrm{ML}} = (\mathbb{S}\mathrm{ig}, \mathrm{Cls}, \mathrm{Sen}, \kappa, \mathrm{Mod}, [\![\_]\!], \models)$, by interpreting the variables in a deterministic way, as presented in [17].

$\underline{\mathrm{ML}}^+$ is defined as a stratified institution with classes, whose *category of signatures* is denoted by $\mathbb{S}\mathrm{ig}^+$. Its objects are tuples (S, F, D), where (S, F) and (S, D) are algebraic signatures of $\underline{\mathrm{ML}}$, such that $F_{w\to s} \cap D_{w\to s} = \emptyset$ for every $w \in S^*$, and $s \in S$. For signatures $\Sigma = (S, F, D)$ and $\Sigma' = (S', F', D')$, a signature morphism $\phi\colon \Sigma \to \Sigma'$ is a tuple $(\phi^{\mathrm{st}}, \phi^{\mathrm{op}}, \phi^{\mathrm{det}})$, where the pairs $(\phi^{\mathrm{st}}, \phi^{\mathrm{op}})$ and $(\phi^{\mathrm{st}}, \phi^{\mathrm{det}})$ are signature morphisms in $\mathbb{S}\mathrm{ig}$.

We define the functor $\mathrm{U}\colon \mathbb{S}\mathrm{ig}^+ \to \mathbb{S}\mathrm{ig}$ by $\mathrm{U}(S, F, D) = (S, F \cup D)$ for signatures, and by $\mathrm{U}(\phi) = (\phi^{\mathrm{st}}, \phi^{\mathrm{op}} \cup \phi^{\mathrm{det}})$ for signature morphisms. The *classes* and the *sentences of a signature* are given by the functor compositions $\mathrm{Cls}^+ = \mathrm{U}\,;\mathrm{Cls}$, and $\mathrm{Sen}^+ = \mathrm{U}\,;\mathrm{Sen}^2$ respectively. The *classes of sentences* are determined by the composition $\mathrm{U}\cdot\kappa\colon \mathrm{Sen}^+ \to \mathrm{Cls}^+$ of the functor U with the natural transformation κ, that is, $(U\cdot\kappa)_\Sigma = \kappa_{\mathrm{U}(\Sigma)}$, for every signature Σ.

The *models* of $\underline{\mathrm{ML}}^+$ are determined by the functor $\mathrm{Mod}^+\colon (\mathbb{S}\mathrm{ig}^+)^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$, that assigns to each signature $\Sigma = (S, F, D)$ the full subcategory of $\mathrm{Mod}(\mathrm{U}(\Sigma))$ consisting of the models M in which every operation symbol in D is interpreted in a deterministic way. For every signature morphism $\phi\colon \Sigma \to \Sigma'$ and every model $M' \in |\mathrm{Mod}^+(\Sigma')|$ we define $\mathrm{Mod}^+(\phi)(M')$ as $\mathrm{Mod}(\mathrm{U}(\phi))(M')$. Notice that Mod^+ is well-defined, as $|\mathrm{Mod}(\mathrm{U}(\phi))(M')_\sigma(m_1, \ldots, m_n)| = |M'_{\phi^{\mathrm{det}}(\sigma)}(m_1, \ldots, m_n)| = 1$, for every operation symbol σ of D.

The *stratification of models* is defined just as in the case of $\underline{\mathrm{ML}}$:

- $[\![\_]\!]^+_{\Sigma,c}\colon \mathrm{Mod}^+(\Sigma) \to \mathbb{S}\mathrm{et}$ maps every model $M \in |\mathrm{Mod}^+(\Sigma)|$ to $[\![M]\!]_{\mathrm{U}(\Sigma),c}$
- $[\![M]\!]^+_{\phi,c}\colon [\![M]\!]^+_{\Sigma',\mathrm{Cls}^+(\phi)(c)} \to \mathrm{Mod}(\phi)\,; [\![M]\!]^+_{\Sigma,c}$ maps every state m to $[\![M]\!]_{\mathrm{U}(\phi),c}(m)$ for every signature morphism $\phi\colon \Sigma \to \Sigma'$ and every class c of Σ.

Finally, the *satisfaction relation* between models and sentences is defined analogously to the satisfaction relation of $\underline{\mathrm{ML}}$. As a result, it holds for example, that for any basic pattern π, any signature $\Sigma \in \mathbb{S}\mathrm{ig}^+$, any class $c \in \mathrm{Cls}^+(\Sigma)$, and any model $M \in |\mathrm{Mod}^+(\Sigma)|$,

$$M(\models^{\underline{\mathrm{ML}}^+})^m_{\Sigma,c}\pi \quad \text{iff} \quad M(\models^{\underline{\mathrm{ML}}})^m_{\mathrm{U}(\Sigma),c}\pi.$$

We note, however, that the satisfaction relation of $\underline{\mathrm{ML}}^+$ is not a restriction of the satisfaction relation of $\underline{\mathrm{ML}}$. For example, if π were an existentially quantified pattern $\exists x : t.\pi_1$, then only the converse implication of the above equivalence would be ensured to hold. This follows because in $\underline{\mathrm{ML}}$ every expansion of M may interpret in a non-deterministic manner the variable $x : t$; in order words, there is no guarantee that there exists an expansion of M in $\underline{\mathrm{ML}}$ that satisfies π and is also a model of $\underline{\mathrm{ML}}^+$. A similar proof to the one of Proposition 2 can be given for the satisfaction condition of $\underline{\mathrm{ML}}^+$.

[2] Technically, the quantification in $\underline{\mathrm{ML}}^+$ is done only over variables that are interpreted in a deterministic manner. This means that every extension with variables over signature $\mathrm{U}(\Sigma)$ (in $\underline{\mathrm{ML}}$) corresponds to a deterministic extension of Σ in $\underline{\mathrm{ML}}^+$.

3.3 Relationship to First-Order Logic

There exists a comorphism of institutions between $\flat\mathbf{ML}^+$, the institution obtained from $\underline{\mathrm{ML}}^+$ following Proposition 1, and **FOL**, the institution of first-order logic. We define $(\Phi, \alpha, \beta)\colon \flat\mathbf{ML}^+ \to \mathbf{FOL}$ as follows:

For signatures: The underlying signature functor $\Phi\colon \mathbb{S}\mathrm{ig}^{\flat\mathbf{ML}^+} \to \mathbb{S}\mathrm{ig}^{\mathbf{FOL}}$ maps

- every $\flat\mathbf{ML}^+$ signature $\Sigma = (S, F, D)$ to the **FOL** signature $\Sigma' = (S', F', P')$, with the same sorts as Σ, with predicates $\sigma \in P'_{ws}$ for every function symbol $\sigma \in F_{w \to s}$, and function symbols $\sigma \in F'_{w \to s}$ for each function symbol $\sigma \in D_{w \to s}$

$$S' = S \qquad F'_{w \to s} = D_{w \to s} \qquad P'_{w'} = \begin{cases} F_{w \to s} & \text{for } w' = ws \\ \emptyset & \text{for } w' = \lambda \end{cases}$$

- every $\flat\mathbf{ML}^+$-signature morphism $\phi\colon \Sigma_1 \to \Sigma_2$ to the **FOL**-signature morphism $\phi' = (\phi'^{\mathrm{st}}, \phi'^{\mathrm{op}}, \phi'^{\mathrm{rel}})$, where $\phi'^{\mathrm{st}} = \phi^{\mathrm{st}}$, $\phi'^{\mathrm{op}} = \phi^{\mathrm{det}}$, and $\phi'^{\mathrm{rel}}_{ws} = \phi^{\mathrm{op}}_{w \to s}$, for $ws \neq \lambda$.

For models: For every signature $\Sigma = (S, F, D)$, $\beta_\Sigma\colon \mathrm{Mod}^{\mathbf{FOL}}(\Phi(\Sigma)) \to \mathrm{Mod}^{\flat\mathbf{ML}^+}(\Sigma)$ is the functor that maps

- every first-order structure M' for $\Phi(\Sigma)$ to the multialgebra M whose carrier sets M_s are defined as M'_s, for every sort $s \in S$, whose interpretations $M_\sigma\colon M_{s_1} \times \ldots \times M_{s_n} \to 2^{M_s}$ of function symbols $\sigma \in F_{s_1 \ldots s_n \to s}$ are defined as $M_\sigma(m_1, \ldots, m_n) = \{m \in M_s \mid (m_1, \ldots, m_n, m) \in M'_\sigma\}$, and whose interpretations M_σ of function symbols $\sigma \in D_{w \to s}$ are given by the composition of M'_σ with the singleton-forming map $\{\_\}\colon M_s \to 2^{M_s}$, and
- every morphism of first-order structures $h'\colon M' \to N'$ in $\mathrm{Mod}^{\mathbf{FOL}}(\Phi(\Sigma))$ to a multialgebra morphism $h\colon \beta_\Sigma(M') \to \beta_\Sigma(N')$ given by $h_s = h'_s$, for every $s \in S$. We note that the fact that h' commutes with the interpretation of operation symbols suits the deterministic nature of the morphism h for the interpretation of operations in D, while its compatibility with the interpretation of predicate symbols guarantees the satisfaction of the morphism condition for multialgebras.

Proposition 3. *$\beta\colon \Phi^{\mathrm{op}} ; \mathrm{Mod}^{\mathbf{FOL}} \Rightarrow \mathrm{Mod}^{\flat\mathbf{ML}^+}$ is a natural transformation.*

Proof. We have to show that the following diagram commutes for every pair of $\flat\mathbf{ML}^+$ signatures $\Sigma_1 = (S_1, F_1, D_1)$, $\Sigma_2 = (S_2, F_2, D_2)$, and every morphism $\phi\colon \Sigma_1 \to \Sigma_2$.

$$\begin{tikzcd}
\mathrm{Mod}^{\mathbf{FOL}}(\Phi(\Sigma_1)) \arrow[r, "\beta_{\Sigma_1}"] & \mathrm{Mod}^{\flat\mathbf{ML}^+}(\Sigma_1) \\
\mathrm{Mod}^{\mathbf{FOL}}(\Phi(\Sigma_2)) \arrow[u, "\mathrm{Mod}^{\mathbf{FOL}}(\Phi(\phi))"] \arrow[r, "\beta_{\Sigma_2}"'] & \mathrm{Mod}^{\flat\mathbf{ML}^+}(\Sigma_2) \arrow[u, "\mathrm{Mod}^{\flat\mathbf{ML}^+}(\phi)"']
\end{tikzcd}$$

We only give the details for the case of models; the morphisms of models can be treated similarly. Therefore, we want to show that for every model $M' \in |\mathrm{Mod}^{\mathbf{FOL}}(\Phi(\Sigma_2))|$, $\beta_{\Sigma_1}(M'\restriction_{\Phi(\phi)}) = \beta_{\Sigma_2}(M')\restriction_\phi$. We denote by S_i, F_i, D_i the components of Σ_i, for $i \in \{1, 2\}$.

- For every $s \in S_1$, we can easily see that $\beta_{\Sigma_1}(M'\restriction_{\Phi(\phi)})_s = (\beta_{\Sigma_2}(M')\restriction_\phi)_s$, because $\beta_{\Sigma_1}(M'\restriction_{\Phi(\phi)})_s = (M'\restriction_{\Phi(\phi)})_s = M'_{\phi^{\mathrm{st}}(s)}$ and $(\beta_{\Sigma_2}(M')\restriction_\phi)_s = \beta_{\Sigma_2}(M')_{\phi^{\mathrm{st}}(s)} = M'_{\phi^{\mathrm{st}}(s)}$.
- To show that $\beta_{\Sigma_1}(M'\restriction_{\Phi(\phi)})_\sigma = (\beta_{\Sigma_2}(M')\restriction_\phi)_\sigma$ for every $\sigma \in F_{1 s_1 \dots s_n \to s}$, we first observe that $(M'\restriction_{\Phi(\phi)})_\sigma = M'_{\phi^{\mathrm{op}}(\sigma)}$, and thus

  $$\begin{aligned}\beta_{\Sigma_1}(M'\restriction_{\Phi(\phi)})_\sigma(m_1, \dots, m_n) &= \{m \in \beta_{\Sigma_1}(M'\restriction_{\Phi(\phi)})_s \mid (m_1, \dots, m_n, m) \in (M'\restriction_{\Phi(\phi)})_\sigma\} \\ &= \{m \in M'_{\phi^{\mathrm{st}}(s)} \mid (m_1, \dots, m_n, m) \in M'_{\phi^{\mathrm{op}}(\sigma)}\}.\end{aligned}$$

  Moreover, by the definition of model reducts, we obtain

  $$\begin{aligned}(\beta_{\Sigma_2}(M')\restriction_\phi)_\sigma(m_1, \dots, m_n) &= \beta_{\Sigma_2}(M')_{\phi^{\mathrm{op}}(\sigma)}(m_1, \dots, m_n) \\ &= \{m \in \beta_{\Sigma_2}(M')_{\phi^{\mathrm{st}}(s)} \mid (m_1, \dots, m_n, m) \in M'_{\phi^{\mathrm{op}}(\sigma)}\} \\ &= \{m \in M'_{\phi^{\mathrm{st}}(s)} \mid (m_1, \dots, m_n, m) \in M'_{\phi^{\mathrm{op}}(\sigma)}\}\end{aligned}$$

- It can be easily observed that $\beta_{\Sigma_1}(M'\restriction_{\Phi(\phi)})_\sigma = (\beta_{\Sigma_2}(M')\restriction_\phi)_\sigma$ for every function symbol $\sigma \in D_{1 s_1 \dots s_n \to s}$, as $(M'\restriction_{\Phi(\phi)})_\sigma = M'_{\phi^{\mathrm{det}}(\sigma)}$.

□

For sentences: For every signature $\Sigma = (S, F, D)$, $\alpha_\Sigma \colon \mathrm{Sen}^{\flat\mathbf{ML}^+}(\Sigma) \to \mathrm{Sen}^{\mathbf{FOL}}(\Phi(\Sigma))$ is the function that maps every Σ-pattern π to

$$\alpha_\Sigma(\pi) = \forall m : s.\mathrm{FOL}_\Sigma^{m:s}(\pi),$$

where $s = \kappa_\Sigma(\pi)$, m is a first-order variable of sort s for the signature $\Phi(\Sigma)$, and $\mathrm{FOL}_\Sigma^{m:s} \colon \kappa_\Sigma^{-1}(s) \to \mathrm{Sen}^{\mathbf{FOL}}(\Phi(\Sigma) \uplus \{m : s\})$ is the sorted translation of sentences defined as follows:

> We begin with a notation: for every operation symbol $\sigma \in (F \cup D)_{s_1, \dots, s_n \to s}$, and every variables $m_i : s_i$ and $m : s$, we denote by $\sigma^{=}(m_1, \dots, m_n, m)$ either the relational atom $\sigma(m_1, \dots, m_n, m)$ if $\sigma \in F$, or the equational atom $\sigma(m_1, \dots, m_n) = m$ if $\sigma \in D$.

- for every basic pattern $\pi \in (F \cup D)_{\lambda \to s}$,

  $$\mathrm{FOL}_\Sigma^{m:s}(\pi) = \pi^{=}(m),$$

- for every basic pattern $\pi = \sigma(t_1, \dots, t_n)$, with $\sigma \in (F \cup D)_{s_1, \dots, s_n \to s}$,

  $$\begin{aligned}\mathrm{FOL}_\Sigma^{m:s}(\pi) = \exists m_1 : s_1 \dots \exists m_n : s_n.\mathrm{FOL}_\Sigma^{m_1:s_1}(t_1) \wedge \dots \wedge \mathrm{FOL}_\Sigma^{m_n:s_n}(t_n) \\ \wedge\, \sigma^{=}(m_1, \dots, m_n, m),\end{aligned}$$

- for every pattern $\pi = \neg\pi_1$,

$$\mathrm{FOL}_\Sigma^{m:s}(\neg\pi_1) = \neg\mathrm{FOL}_\Sigma^{m:s}(\pi_1),$$

- for every pattern $\pi = \pi_1 \wedge \pi_2$,

$$\mathrm{FOL}_\Sigma^{m:s}(\pi_1 \wedge \pi_2) = \mathrm{FOL}_\Sigma^{m:s}(\pi_1) \wedge \mathrm{FOL}_\Sigma^{m:s}(\pi_2),$$

- for every pattern $\pi = \exists x:t.\pi_1$, where $\pi_1 \in \mathrm{Sen}^{\flat\mathbf{ML}^+}(\Sigma \uplus \{x:t\})$ we have

$$\mathrm{FOL}_\Sigma^{m:s}(\exists x:t.\pi_1) = \exists x:t.\xi_\Sigma(\mathrm{FOL}_{\Sigma\uplus\{x:t\}}^{m:s}(\pi_1)),$$

  where ξ_Σ is a **FOL**-signature morphism from $\Phi(\Sigma\uplus\{x:t\})\uplus\{m:s\}$ to $\Phi(\Sigma)\uplus\{m:s\}\uplus\{x:t\}$ defined as the extension of $1_{\Phi(\Sigma)}$ that maps the matching-logic variable $x:t$ for the signature Σ to the first-order variable $x:t$ for the signature $\Phi(\Sigma) \uplus \{m:s\}$, and the first-order variable $m:s$ for the signature $\Phi(\Sigma \uplus \{x:t\})$ to the first-order variable $m:s$ but for the signature $\Phi(\Sigma)$ (see Remark 2 below).

Remark 2. We recall from the definitions of the institutions of matching and first-order logic that from a technical point of view, variables are triples, consisting of name, sort, and signature over which they are defined. Consequently, the signature morphism ξ_Σ maps $\langle x, t, \Sigma\rangle$ to $\langle x, t, \Phi(\Sigma) \uplus \langle m, s, \Phi(\Sigma)\rangle\rangle$, and $\langle m, s, \Phi(\Sigma \uplus x)\rangle$ to $\langle m, s, \Phi(\Sigma)\rangle$. Nonetheless, whenever possible, we will choose the compact notations $x:t$ and $x:s$, and specify the logics and the signatures to which the variables correspond.

The naturality of α results from an analogous property for the family of maps $\mathrm{FOL}_\Sigma^{m:s}$.

Proposition 4. For every two $\flat\mathbf{ML}^+$ signatures Σ_1, Σ_2, every signature morphism $\phi\colon \Sigma_1 \to \Sigma_2$, and every variable $m:s$ for Σ_1, the following diagram commutes.

$$\begin{tikzcd}
\kappa_{\Sigma_1}^{-1}(s) \arrow[r, "\mathrm{FOL}_{\Sigma_1}^{m:s}"] \arrow[d, "\mathrm{Sen}^{\flat\mathbf{ML}^+}(\phi)(\_)"'] & \mathrm{Sen}^{\mathbf{FOL}}(\Phi(\Sigma_1) \uplus \{m:s\}) \arrow[d, "\mathrm{Sen}^{\mathbf{FOL}}(\Phi(\phi)^m)"] \\
\kappa_{\Sigma_2}^{-1}(\phi^{\mathrm{st}}(s)) \arrow[r, "\mathrm{FOL}_{\Sigma_2}^{m:\phi^{\mathrm{st}}(s)}"'] & \mathrm{Sen}^{\mathbf{FOL}}(\Phi(\Sigma_2) \uplus \{m:\phi^{\mathrm{st}}(s)\})
\end{tikzcd}$$

Proof. We prove the statement by induction on the structure of patterns, that is for every pattern $\pi \in \kappa_{\Sigma_1}^{-1}(s)$,

$$\mathrm{FOL}_{\Sigma_2}^{m:\phi^{\mathrm{st}}(s)}(\phi(\pi)) = \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m:s}(\pi)).$$

To this purpose, let us first recall that $\Phi(\phi)^m$ is the canonical extension of the first-order signature morphism $\Phi(\phi)$ that maps the $\Phi(\Sigma_1)$ variable $m:s$ to the $\Phi(\Sigma_2)$ variable $m:\phi^{\mathrm{st}}(s)$.

- For basic patterns $\pi \in (F \cup D)_{\lambda \to s}$

$$\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi)) = \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi^{\mathrm{op}}(\pi)) = \phi^{\mathrm{op}}(\pi)^{=}(m)$$

  and

$$\Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi)) = \Phi(\phi)^m(\pi^{=}(m)) = \phi^{\mathrm{op}}(\pi)^{=}(m).$$

- For basic patterns $\pi = \sigma(t_1, \ldots, t_n) \in (F \cup D)_{s_1 \ldots s_n \to s}$

$$\begin{aligned}\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi)) &= \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi^{\mathrm{op}}(\sigma)(\phi^{\mathrm{tm}}(t_1) \ldots \phi^{\mathrm{tm}}(t_n)))\\ &= \exists m_1 : \phi^{\mathrm{st}}(s_1) \ldots \exists m_n : \phi^{\mathrm{st}}(s_n).\phi^{\mathrm{op}}(\sigma)^{=}(m_1, \ldots, m_n, m)\\ &\quad \wedge \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi^{\mathrm{tm}}(t_1)) \wedge \ldots \wedge \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi^{\mathrm{tm}}(t_n))\end{aligned}$$

  and

$$\begin{aligned}\Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi)) &= \Phi(\phi)^m(\exists m_1 : s_1 \ldots \exists m_n : s_n.\sigma^{=}(m_1, \ldots, m_n, m)\\ &\quad \wedge \mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(t_1) \wedge \ldots \wedge \mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(t_n))\\ &= \exists m_1 : \phi^{\mathrm{st}}(s_1) \ldots \exists m_n : \phi^{\mathrm{st}}(s_n).\phi^{\mathrm{op}}(\sigma)^{=}(m_1, \ldots, m_n, m)\\ &\quad \wedge \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(t_1)) \wedge \ldots \wedge \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(t_n)).\end{aligned}$$

  The conclusion follows since from the induction hypothesis $\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(t_i)) = \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(t_i))$, for $i = \overline{1, n},$.

- For every pattern $\pi = \neg\pi_1$

$$\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\neg\pi_1)) = \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\neg\phi(\pi_1)) = \neg\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi_1))$$

  and

$$\Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\neg\pi_1)) = \Phi(\phi)^m(\neg\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_1)) = \neg\Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_1)).$$

  The conclusion follows since from the induction hypothesis $\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi_1)) = \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_1))$.

- For every pattern $\pi = \pi_1 \wedge \pi_2$

$$\begin{aligned}\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi_1 \wedge \pi_2)) &= \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi_1) \wedge \phi(\pi_2))\\ &= \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi_1)) \wedge \mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi_2))\end{aligned}$$

  and

$$\begin{aligned}\Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_1 \wedge \pi_2)) &= \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_1) \wedge \mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_2))\\ &= \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_1)) \wedge \Phi(\phi)^m(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi_2)).\end{aligned}$$

The conclusion follows since from the induction hypothesis $\mathrm{FOL}^{m\,:\,\phi^{\mathrm{st}}(s)}_{\Sigma_2}(\phi(\pi_i)) = \Phi(\phi)^m(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1}(\pi_i))$, for $i \in \{1, 2\}$.

- For every pattern $\pi = \exists x : t.\pi_1$, where $\pi_1 \in \mathrm{Sen}^{\flat\mathbf{ML}}(\Sigma_1 \uplus \{x : t\})$

$$\begin{aligned}\mathrm{FOL}^{m\,:\,\phi^{\mathrm{st}}(s)}_{\Sigma_2}(\phi(\exists x : t.\pi_1)) &= \mathrm{FOL}^{m\,:\,\phi^{\mathrm{st}}(s)}_{\Sigma_2}(\exists x : \phi^{\mathrm{st}}(t).\phi^x(\pi_1)) \\ &= \exists x : \phi^{\mathrm{st}}(t).\xi_{\Sigma_2}(\mathrm{FOL}^{m\,:\,\phi^{\mathrm{st}}(s)}_{\Sigma_2 \uplus \{x\,:\,\phi^{\mathrm{st}}(t)\}}(\phi^x(\pi_1)))\end{aligned}$$

and

$$\begin{aligned}\Phi(\phi)^m(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1}(\exists x : t.\pi_1)) &= \Phi(\phi)^m(\exists x : t.\xi_{\Sigma_1}(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1 \uplus \{x\,:\,t\}}(\pi_1))) \\ &= \exists x : \phi^{\mathrm{st}}(t).(\Phi(\phi)^m)^x(\xi_{\Sigma_1}(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1 \uplus \{x\,:\,t\}}(\pi_1)))\end{aligned}$$

From the induction hypothesis we have

$$\mathrm{FOL}^{m\,:\,\phi^{\mathrm{st}}(s)}_{\Sigma_2 \uplus \{x\,:\,\phi^{\mathrm{st}}(t)\}}(\phi^x(\pi_1)) = (\Phi(\phi)^x)^m(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1 \uplus \{x\,:\,t\}}(\pi_1)),$$

which, combined with the commutativity of the diagram below, yields

$$\begin{aligned}\mathrm{FOL}^{m\,:\,\phi^{\mathrm{st}}(s)}_{\Sigma_2}(\phi(\exists x : t.\pi_1)) &= \exists x : \phi^{\mathrm{st}}(t).\xi_{\Sigma_2}((\Phi(\phi)^x)^m(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1 \uplus \{x\,:\,t\}}(\pi_1))) \\ &= \exists x : \phi^{\mathrm{st}}(t).(\Phi(\phi)^m)^x(\xi_{\Sigma_1}(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1 \uplus \{x\,:\,t\}}(\pi_1))) \\ &= \Phi(\phi)^m(\mathrm{FOL}^{m\,:\,s}_{\Sigma_1}(\exists x : t.\pi_1)).\end{aligned}$$

$$\begin{tikzcd}
\Phi(\Sigma_1 \uplus \{x : t\}) \uplus \{m : s\} \arrow[r, "\xi_{\Sigma_1}"] \arrow[d, "(\Phi(\phi^x))^m"'] & \Phi(\Sigma_1) \uplus \{m : s\} \uplus \{x : t\} \arrow[d, "((\Phi(\phi))^m)^x"] \\
\Phi(\Sigma_2 \uplus \{x : \phi^{\mathrm{st}}(t)\}) \uplus \{m : \phi^{\mathrm{st}}(s)\} \arrow[r, "\xi_{\Sigma_2}"'] & \Phi(\Sigma_2) \uplus \{m : \phi^{\mathrm{st}}(s)\} \uplus \{x : \phi^{\mathrm{st}}(t)\}
\end{tikzcd}$$

□

It is now straightforward to prove the naturality of α.

Proposition 5. $\alpha\colon \mathrm{Sen}^{\flat\mathbf{ML}^+} \Rightarrow \Phi\,;\mathrm{Sen}^{\mathbf{FOL}}$ is a natural transformation.

Proof. We have to show that the following diagram commutes for every two $\flat\mathbf{ML}^+$ signatures $\Sigma_1,\ \Sigma_2$, and every signature morphism $\phi\colon \Sigma_1 \to \Sigma_2$.

$$\begin{tikzcd}
\mathrm{Sen}^{\flat\mathbf{ML}^+}(\Sigma_1) \arrow[r, "\alpha_{\Sigma_1}"] \arrow[d, "\mathrm{Sen}^{\flat\mathbf{ML}^+}(\phi)"'] & \mathrm{Sen}^{\mathbf{FOL}}(\Phi(\Sigma_1)) \arrow[d, "\mathrm{Sen}^{\mathbf{FOL}}(\Phi(\phi))"] \\
\mathrm{Sen}^{\flat\mathbf{ML}^+}(\Sigma_2) \arrow[r, "\alpha_{\Sigma_2}"'] & \mathrm{Sen}^{\mathbf{FOL}}(\Phi(\Sigma_2))
\end{tikzcd}$$

For every pattern $\pi \in \mathrm{Sen}^{\flat\mathbf{ML}^+}(\Sigma_1)$,

$$\alpha_{\Sigma_2}(\phi(\pi)) = \forall m\colon \phi^{\mathrm{st}}(s).\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi)),$$

where $s = \kappa_{\Sigma_1}(\pi)$. On the other hand,

$$\Phi(\phi)(\alpha_{\Sigma_1}(\pi)) = \Phi(\phi)(\forall m\colon s.\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi)) = \forall m\colon \phi^{\mathrm{st}}(s).\Phi(\phi)(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi)).$$

We recall that the equality

$$\mathrm{FOL}_{\Sigma_2}^{m\,:\,\phi^{\mathrm{st}}(s)}(\phi(\pi)) = \Phi(\phi)(\mathrm{FOL}_{\Sigma_1}^{m\,:\,s}(\pi))$$

is guaranteed by Proposition 4. □

Satisfaction condition. We conclude our presentation by showing that the definitions of the components of the comorphism given above guarantee that the satisfaction condition holds.

Proposition 6. For every $\flat\mathbf{ML}^+$ signature Σ, every first-order structure M for $\Phi(\Sigma)$, and every Σ-pattern π of sort s,

$$M \models^{\mathbf{FOL}}_{\Phi(\Sigma)} \alpha_\Sigma(\pi) \quad \text{iff} \quad \beta_\Sigma(M) \models^{\flat\mathbf{ML}^+}_{\Sigma} \pi.$$

Proof. We rewrite the left-hand side of the equivalence as

$$\begin{aligned} M \models^{\mathbf{FOL}} \alpha_\Sigma(\pi) \quad &\text{iff} \quad M \models^{\mathbf{FOL}} \forall m\colon s.\mathrm{FOL}_{\Sigma}^{m\,:\,s}(\pi) \\ &\text{iff} \quad M_{\mathrm{FOL}_{\Sigma}^{m\,:\,s}(\pi)} = M_s \end{aligned}$$

and the right-hand side as

$$\beta_\Sigma(M) \models^{\flat\mathbf{ML}^+} \pi \quad \text{iff} \quad \beta_\Sigma(M)_\pi = \beta_\Sigma(M)_s = M_s.$$

In order to show that the satisfaction condition holds it thus suffices to prove the equality $M_{\mathrm{FOL}_{\Sigma}^{m\,:\,s}(\pi)} = \beta_\Sigma(M)_\pi$, which is discussed below, in Proposition 7. □

Proposition 7. For every $\flat\mathbf{ML}^+$ signature Σ, every first-order structure M for $\Phi(\Sigma)$, and every Σ-pattern π of sort s,

$$M_{\mathrm{FOL}_{\Sigma}^{m\,:\,s}(\pi)} = \beta_\Sigma(M)_\pi.$$

Proof. We prove the property by induction on the structure of π:

- For basic patterns $\pi \in F_{\lambda \to s}$

$$M_{\mathrm{FOL}^{m\,:\,s}_{\Sigma}(\pi)} = \{m \in M_s \mid (M, m) \models \pi(m)\} = M_\pi$$

  and

$$\beta_\Sigma(M)_\pi = M_\pi.$$

- For basic patterns $\pi = \sigma(t_1, \ldots, t_n) \in F_{s_1 \ldots s_n \to s}$

$$\begin{aligned}
&M_{\mathrm{FOL}^{m\,:\,s}_{\Sigma}(\pi)}\\
&\quad = \{m \in M_s \mid (M, m) \models \exists x_1 : s_1 \ldots \exists x_n : s_n . \sigma(x_1, \ldots, x_n, m)\\
&\qquad\qquad \wedge \mathrm{FOL}^{x_1\,:\,s_1}_{\Sigma}(t_1) \wedge \ldots \wedge \mathrm{FOL}^{x_n\,:\,s_n}_{\Sigma}(t_n)\}\\
&\quad = \{m \in M_s \mid \text{there exist } m_i \in M_{s_i},\ \text{for } i = \overline{1, n},\ \text{such that}\\
&\qquad\qquad (M, m, m_1, \ldots, m_n) \models \sigma(x_1, \ldots, x_n, m)\\
&\qquad\qquad \wedge \mathrm{FOL}^{x_1\,:\,s_1}_{\Sigma}(t_1) \wedge \ldots \wedge \mathrm{FOL}^{x_n\,:\,s_n}_{\Sigma}(t_n)\}\\
&\quad = \{m \in M_s \mid \text{there exist } m_i \in M_{s_i},\ \text{for } i = \overline{1, n},\ \text{such that}\\
&\qquad\qquad (M, m, m_i) \models \mathrm{FOL}^{x_i\,:\,s_i}_{\Sigma}(t_i),\ \text{and}\\
&\qquad\qquad (M, m, m_1, \ldots, m_n) \models \sigma(x_1, \ldots, x_n, m)\}\\
&\quad = \{m \in M_s \mid \text{there exist } m_i \in M_{s_i},\ \text{for } i = \overline{1, n},\ \text{such that } m_i \in M_{\mathrm{FOL}^{x_i\,:\,s_i}_{\Sigma}(t_i)},\\
&\qquad\qquad \text{for } i = \overline{1, n},\ \text{and } (m_1, \ldots, m_n, m) \in M_\sigma\}\\
&\quad = \{m \in M_s \mid \text{there exist } m_i \in M_{\mathrm{FOL}^{x_i\,:\,s_i}_{\Sigma}(t_i)}, \text{for } i = \overline{1, n},\\
&\qquad\qquad \text{and } (m_1, \ldots, m_n, m) \in M_\sigma\}
\end{aligned}$$

  and

$$\begin{aligned}
&\beta_\Sigma(M)_\pi\\
&\quad = \bigcup \{\beta_\Sigma(M)_\sigma(m_1, \ldots, m_n) \mid m_i \in \beta_\Sigma(M)_{t_i},\ \text{for } i = \overline{1, n}\}\\
&\quad = \{m \in \beta_\Sigma(M)_s \mid \text{there exist } m_i \in \beta_\Sigma(M)_{t_i},\ \text{for } i = \overline{1, n},\\
&\qquad\qquad \text{s.t. } (m_1, \ldots, m_n, m) \in M_\sigma\}\\
&\quad = \{m \in M_s \mid \text{there exist } m_i \in \beta_\Sigma(M)_{t_i},\ \text{for } i = \overline{1, n},\\
&\qquad\qquad \text{s.t. } (m_1, \ldots, m_n, m) \in M_\sigma\}.
\end{aligned}$$

  The result follows from the induction hypothesis, as

$$M_{\mathrm{FOL}^{x_i\,:\,s_i}_{\Sigma}(t_i)} = \beta_\Sigma(M)_{t_i}.$$

- For every pattern $\pi \in D_{\lambda \to s}$ or $\pi = \sigma(t_1, \ldots, t_n) \in D_{s_1 \ldots s_n \to s}$ we proceed similarly to the above two items.

- For every pattern $\pi = \neg\pi_1$

$$\begin{aligned} M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi)} &= \{m \in M_s \mid (M,m) \models \neg\mathrm{FOL}^{m:s}_{\Sigma}(\pi_1)\} \\ &= \{m \in M_s \mid (M,m) \not\models \mathrm{FOL}^{m:s}_{\Sigma}(\pi_1)\} \\ &= \{m \in M_s \mid m \notin M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi_1)}\} \end{aligned}$$

  and

$$\beta_{\Sigma}(M)_{\pi} = \beta_{\Sigma}(M)_s \setminus \beta_{\Sigma}(M)_{\pi_1} = M_s \setminus \beta_{\Sigma}(M)_{\pi_1}.$$

  The result follows from the induction hypothesis, as

$$M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi_1)} = \beta_{\Sigma}(M)_{\pi_1}.$$

- For every pattern $\pi = \pi_1 \wedge \pi_2$

$$\begin{aligned} M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi)} &= \{m \in M_s \mid (M,m) \models \mathrm{FOL}^{m:s}_{\Sigma}(\pi_1 \wedge \pi_2)\} \\ &= \{m \in M_s \mid (M,m) \models \mathrm{FOL}^{m:s}_{\Sigma}(\pi_1) \wedge \mathrm{FOL}^{m:s}_{\Sigma}(\pi_2)\} \\ &= \{m \in M_s \mid (M,m) \models \mathrm{FOL}^{m:s}_{\Sigma}(\pi_1) \text{ and} \\ &\qquad\qquad\quad (M,m) \models \mathrm{FOL}^{m:s}_{\Sigma}(\pi_2)\} \\ &= \{m \in M_s \mid m \in M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi_1)} \text{ and } m \in M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi_2)}\} \end{aligned}$$

  and

$$\beta_{\Sigma}(M)_{\pi} = \beta_{\Sigma_1}(M)_{\pi_1} \cap \ \beta_{\Sigma_1}(M)_{\pi_2}.$$

  The result follows from the induction hypothesis, as

$$M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi_i)} = \beta_{\Sigma}(M)_{\pi_i}, \text{ for } i \in \{1,2\}.$$

- For every pattern $\pi = \exists x\!:\!t.\pi_1$, where $\pi_1 \in \mathrm{Sen}^{\flat\mathbf{ML}}(\Sigma_1 \uplus \{x\!:\!t\})$

$$\begin{aligned} M_{\mathrm{FOL}^{m:s}_{\Sigma}(\pi)} &= \{m \in M_s \mid (M,m) \models \exists x\!:\!t.\xi_{\Sigma}(\mathrm{FOL}^{m:s}_{\Sigma\uplus\{x:t\}}(\pi_1))\} \\ &= \{m \in M_s \mid \text{exists } n \in M_t \text{ s.t. } (M,m,n) \models \xi_{\Sigma}(\mathrm{FOL}^{m:s}_{\Sigma\uplus\{x:t\}}(\pi_1))\} \\ &= \{m \in M_s \mid \text{exists } n \in M_t \text{ s.t. } (M,m,n)\!\upharpoonright_{\xi_{\Sigma}} \models \mathrm{FOL}^{m:s}_{\Sigma\uplus\{x:t\}}(\pi_1)\} \\ &= \{m \in M_s \mid \text{exists } n \in M_t \text{ s.t. } m \in (M,n)_{\mathrm{FOL}^{m:s}_{\Sigma\uplus\{x:t\}}(\pi_1)}\}, \\ &\qquad \text{where } (M,n) = (M,m,n)\!\upharpoonright_{\xi_{\Sigma}}\!\upharpoonright_{\Phi(\Sigma)\uplus\{x:t\}} \end{aligned}$$

and

$$\begin{aligned}\beta_\Sigma(M)_\pi &= \bigcup\{(\beta_\Sigma(M), n)_{\pi_1} \mid n \in \beta_\Sigma(M)_t\} \\ &= \{m \in (\beta_\Sigma(M), n)_s \mid \text{exists } n \in \beta_\Sigma(M)_t \text{ s.t. } m \in (\beta_\Sigma(M), n)_{\pi_1}\} \\ &= \{m \in M_s \mid \text{exists } n \in M_t, \text{ s.t. } m \in \beta_{\Sigma \uplus \{x\,:\,t\}}(M, n)_{\pi_1}\}.\end{aligned}$$

The result follows from the induction hypothesis, as

$$(M, n)_{\mathrm{FOL}^{m\,:\,s}_{\Sigma \uplus \{x\,:\,t\}}(\pi_1)} = \beta_{\Sigma \uplus \{x\,:\,t\}}(M, n)_{\pi_1}.$$

□

4 Reachability Logic

In order to capture reachability logic [20] as an institution, we first define an abstract, parameterised institution over an arbitrary stratified institution with classes, which necessarily has to enjoy properties such as the existence of a quantification space, model amalgamation, and preservation of pushouts by the class functor. We then obtain the concrete version of reachability logic that underlies the $\mathbb{K}$ framework by instantiating the parameter of the abstract version with $\underline{\mathrm{ML}}^+$, the stratified institution with classes of matching logic, which we show to satisfy the desired properties.

4.1 Abstract Reachability Logic

We formalise reachability logic in two steps: we begin by describing a sub-institution of reachability logic whose sentences are all atomic (reachability atoms), and we subsequently extend it by adding logical connectives and quantifiers through a general universal-quantification construction.

To define atomic abstract reachability logic we first describe it as a pre-institution [21] whose construction is based upon a stratified institution with classes. This amounts to defining the same elements as those comprised by an institution – a category of signatures, sentence and model functors, as well as a satisfaction relation between sentences and models – but without imposing the requirement of the satisfaction condition.

Throughout this section we assume an arbitrary, but fixed stratified institution with classes $\underline{\mathrm{M}} = (\mathbb{S}\mathrm{ig}^{\underline{\mathrm{M}}}, \mathrm{Cls}^{\underline{\mathrm{M}}}, \mathrm{Sen}^{\underline{\mathrm{M}}}, \mathrm{Mod}^{\underline{\mathrm{M}}}, [\![\_]\!]^{\underline{\mathrm{M}}}, \models^{\underline{\mathrm{M}}})$. This serves as a parameter for all the constructions considered below.

Signatures. The category of signatures of atomic abstract reachability logic, denoted by $\mathbb{S}\mathrm{ig}^{\mathbf{ARL}(\underline{\mathrm{M}})}$, is the same as the category of signatures of $\underline{\mathrm{M}}$.

Sentences. For every signature Σ, $\mathrm{Sen}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\Sigma)$ is the set of pairs of sentences of the stratified institution with classes, denoted by $\pi_1 \Rightarrow \pi_2$, where $\pi_1, \pi_2 \in \mathrm{Sen}^{\underline{\mathrm{M}}}(\Sigma)$. The translation of such a sentence $\pi_1 \Rightarrow \pi_2$ along a signature morphism $\phi\colon \Sigma \to \Sigma'$ is defined as the pair of its translated components according to $\mathrm{Sen}^{\underline{\mathrm{M}}}(\phi)$:

$$\mathrm{Sen}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\phi)(\pi_1 \Rightarrow \pi_2) = \mathrm{Sen}^{\underline{\mathrm{M}}}(\phi)(\pi_1) \Rightarrow \mathrm{Sen}^{\underline{\mathrm{M}}}(\phi)(\pi_2).$$

Example. If we instantiate the parameter $\underline{\mathrm{M}}$ with the stratified institution with classes $\underline{\mathrm{ML}}^+$, the sentences of $\mathbf{ARL}(\underline{\mathrm{ML}}^+)$ will only capture atomic $\mathbb{K}$ semantic rules, i.e. without quantification and side conditions. This means we could only express atomic rules in the specification of the simple imperative programming language IMP, like **rule** ! true => $\mathrm{not}_{\mathrm{Bool}}$ true. The quantifiers and boolean conditions will be introduced later in this section, through a general construction.

Models. The reachability models of a signature Σ, given by the functor $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}$, are pairs $(M, \rightsquigarrow)$, of Σ-models M of the underlying stratified institution with classes, and families of preorders $\rightsquigarrow_c \subseteq \llbracket M \rrbracket_{\Sigma,c} \times \llbracket M \rrbracket_{\Sigma,c}$ indexed by the classes of the signature. The model homomorphisms $h\colon (M_1, \rightsquigarrow_1) \to (M_2, \rightsquigarrow_2)$ are defined as the morphisms between the $\underline{\mathrm{M}}$-models M_1 and M_2 that preserve the preorders: for every $c \in \mathrm{Cls}(\Sigma)$, the function $\llbracket h \rrbracket_{\Sigma,c}$ from $(\llbracket M_1 \rrbracket_{\Sigma,c}, \rightsquigarrow_1)$ to $(\llbracket M_2 \rrbracket_{\Sigma,c}, \rightsquigarrow_2)$ is monotone. This allows $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\Sigma)$ to inherit the identities and the composition of model homomorphisms of $\mathrm{Mod}^{\underline{\mathrm{M}}}(\Sigma)$.

For every two signatures Σ, Σ', and every signature morphism $\phi\colon \Sigma \to \Sigma'$, the model reduct $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\phi)\colon \mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\Sigma) \to \mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\Sigma')$ is defined as

- for every Σ' model $(M', \rightsquigarrow')$, $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\phi)(M', \rightsquigarrow') = (\mathrm{Mod}^{\underline{\mathrm{M}}}(\phi)(M'), \rightsquigarrow)$, where $\rightsquigarrow_c \subseteq \llbracket M' \upharpoonright_\phi \rrbracket_{\Sigma,c} \times \llbracket M' \upharpoonright_\phi \rrbracket_{\Sigma,c}$ is the reflexive and transitive closure of $\llbracket M' \rrbracket_{\phi,c}(\rightsquigarrow'_{\phi(c)})$, which will be further denoted by $\to_c$,
- for every two Σ' models M'_1, M'_2, and every model homomorphism $h'\colon (M'_1, \rightsquigarrow'_1) \to (M'_2, \rightsquigarrow'_2)$, $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\phi)(h')$ is simply $\mathrm{Mod}^{\underline{\mathrm{M}}}(\phi)(h')$.

Proposition 8. For every signature morphism $\phi\colon \Sigma \to \Sigma'$, the map $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\phi)$ is well-defined.

Proof. We have to show that for every two Σ'-models $(M'_1, \rightsquigarrow'_1), (M'_2, \rightsquigarrow'_2)$, every model homomorphism $h'\colon (M'_1, \rightsquigarrow'_1) \to (M'_2, \rightsquigarrow'_2)$, and every class $c \in \mathrm{Cls}(\Sigma)$, the function $\llbracket h' \upharpoonright_\phi \rrbracket_{\Sigma,c}\colon \llbracket M'_1 \upharpoonright_\phi \rrbracket_{\Sigma,c} \to \llbracket M'_2 \upharpoonright_\phi \rrbracket_{\Sigma,c}$ is monotone. Therefore, for every $m, n \in \llbracket M'_1 \upharpoonright_\phi \rrbracket_{\Sigma,c}$, such that $m \rightsquigarrow_{1,c} n$, we need to show that $\llbracket h' \upharpoonright_\phi \rrbracket_{\Sigma,c}(m) \rightsquigarrow_{2,c} \llbracket h' \upharpoonright_\phi \rrbracket_{\Sigma,c}(n)$, where $\rightsquigarrow_{1,c}$ and $\rightsquigarrow_{2,c}$ are the preorders resulting from the reduction of $(M'_1, \rightsquigarrow'_1)$ and $(M'_2, \rightsquigarrow'_2)$ along ϕ.

From $m \rightsquigarrow_{1,c} n$ we deduce that there exist $x_0, \ldots, x_k \in \llbracket M'_1 \upharpoonright_\phi \rrbracket_{\Sigma,c}$ such that $m = x_0 \to_{1,c} x_1 \to_{1,c} \ldots \to_{1,c} x_k = n$. From the definition of $\to_{1,c}$ results that there exist $u_0, \ldots, u_{k-1}, v_1, \ldots, v_k \in \llbracket M'_1 \rrbracket_{\Sigma',\phi(c)}$ such that $\llbracket M'_1 \rrbracket_{\phi,c}(u_i) = \llbracket M'_1 \rrbracket_{\phi,c}(v_i) = x_i$ and $u_i \rightsquigarrow'_{1,\phi(c)} v_{i+1}$, for $i = \overline{1,k}$. As $\llbracket h' \rrbracket_{\Sigma',\phi(c)}\colon (\llbracket M'_1 \rrbracket_{\Sigma',\phi(c)}, \rightsquigarrow'_{1,\phi(c)}) \to (\llbracket M'_2 \rrbracket_{\Sigma',\phi(c)}, \rightsquigarrow'_{2\phi(c)})$ is monotone, we deduce that $\llbracket h' \rrbracket_{\Sigma',\phi(c)}(u_i) \rightsquigarrow'_{2,\phi(c)} \llbracket h' \rrbracket_{\Sigma',\phi(c)}(v_{i+1})$, for $i = \overline{1,k}$. From the definition of $\to_{2,c}$ it follows that $\llbracket M'_2 \rrbracket_{\phi,c}(\llbracket h' \rrbracket_{\Sigma',\phi(c)}(u_i)) \to_{2,c} \llbracket M'_2 \rrbracket_{\phi,c}(\llbracket h' \rrbracket_{\Sigma',\phi(c)}(v_{i+1}))$, and thus, from the commutativity of the diagram below, $\llbracket h'_1 \upharpoonright_\phi \rrbracket_{\Sigma,c}(\llbracket M'_1 \rrbracket_{\phi,c}(u_i)) \to_{2,c} \llbracket h'_1 \upharpoonright_\phi \rrbracket_{\Sigma,c}(\llbracket M'_1 \rrbracket_{\phi,c}(v_{i+1}))$, for $i = \overline{1,k}$. We now have $\llbracket h'_1 \upharpoonright_\phi \rrbracket_{\Sigma,c}(x_i) \to_{2,c} \llbracket h'_1 \upharpoonright_\phi \rrbracket_{\Sigma,c}(x_{i+1})$, for $i = \overline{1,k}$ and hence $\llbracket h' \upharpoonright_\phi \rrbracket_{\Sigma,c}(m) = \llbracket h' \upharpoonright_\phi \rrbracket_{\Sigma,c}(x_0) \rightsquigarrow_{2,c} \llbracket h' \upharpoonright_\phi \rrbracket_{\Sigma,c}(x_k) = \llbracket h' \upharpoonright_\phi \rrbracket_{\Sigma,c}(n)$.

$$\begin{tikzcd}
\llbracket M_1' \rrbracket_{\Sigma',\phi(c)} \arrow[r, "\llbracket M_1' \rrbracket_{\phi,c}"] \arrow[d, "\llbracket h' \rrbracket_{\Sigma',\phi(c)}"'] & \llbracket M_1' {\upharpoonright_{\phi}} \rrbracket_{\Sigma,c} \arrow[d, "\llbracket h' {\upharpoonright_{\phi}} \rrbracket_{\Sigma,c}"] \\
\llbracket M_2' \rrbracket_{\Sigma',\phi(c)} \arrow[r, "\llbracket M_2' \rrbracket_{\phi,c}"'] & \llbracket M_2' {\upharpoonright_{\phi}} \rrbracket_{\Sigma,c}
\end{tikzcd}$$

□

Proposition 9. $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}$ is a functor.

Proof. It can be easily observed that $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}$ inherits from $\mathrm{Mod}^{\underline{\mathrm{M}}}$ most of the properties needed to ensure that it is a functor. The only situation that requires special attention is the preservation of the composition of signature morphisms, and even in this case, it suffices to consider the reduction of models. We need to make sure that for every three signatures Σ, Σ', and Σ'', every signature morphisms $\phi\colon \Sigma \to \Sigma'$, $\phi'\colon \Sigma' \to \Sigma''$, and every Σ''-model $(M'', \leadsto'')$, the models $((M'', \leadsto''){\upharpoonright_{\phi'}}){\upharpoonright_{\phi}}$ and $(M'', \leadsto''){\upharpoonright_{\phi;\phi'}}$ are equal. Since the equality $(M''{\upharpoonright_{\phi'}}){\upharpoonright_{\phi}} = M''{\upharpoonright_{\phi;\phi'}}$ of their underlying $\underline{\mathrm{M}}$ models is assured by the functoriality of $\mathrm{Mod}^{\underline{\mathrm{M}}}$, we only have to focus on the equality of their corresponding preorders. This means that we need to compare, for every class $c \in \mathrm{Cls}(\Sigma)$,

$$\llbracket M''{\upharpoonright_{\phi'}} \rrbracket_{\phi,c}(\llbracket M'' \rrbracket_{\phi',\phi(c)}(\leadsto''_{\phi'(\phi(c))})^*)^* \quad \text{and} \quad \llbracket M'' \rrbracket_{\phi;\phi',c}(\leadsto''_{\phi'(\phi(c))})^*.$$

From the functoriality of the stratification (see Remark 1) we know that

$$\llbracket M'' \rrbracket_{\phi;\phi',c}(\leadsto''_{\phi'(\phi(c))}) = \llbracket M''{\upharpoonright_{\phi'}} \rrbracket_{\phi,c}(\llbracket M'' \rrbracket_{\phi',\phi(c)}(\leadsto''_{\phi'(\phi(c))})),$$

therefore we need to compare

$$\llbracket M''{\upharpoonright_{\phi'}} \rrbracket_{\phi,c}(\underbrace{\llbracket M'' \rrbracket_{\phi',\phi(c)}(\leadsto''_{\phi'(\phi(c))})}_{R}{}^*)^* \quad \text{and} \quad \llbracket M''{\upharpoonright_{\phi'}} \rrbracket_{\phi,c}(\underbrace{\llbracket M'' \rrbracket_{\phi',\phi(c)}(\leadsto''_{\phi'(\phi(c))})}_{R})^*.$$

The conclusion now follows from a well-known property of reflexive and transitive closures: for any function $f\colon A \to B$ (in this case $\llbracket M''{\upharpoonright_{\phi'}} \rrbracket_{\phi,c}$ from $\llbracket M''{\upharpoonright_{\phi'}} \rrbracket_{\Sigma',\phi(c)}$ to $\llbracket (M''{\upharpoonright_{\phi'}}){\upharpoonright_{\phi}} \rrbracket_{\Sigma,c}$), and any relation $R \subseteq A \times A$, $f(R^*) \subseteq f(R)^*$. This entails that $f(R^*)^* \subseteq f(R)^*$. The inclusion $f(R)^* \subseteq f(R^*)^*$ holds for any closure operators. □

Satisfaction relation. The satisfaction relation between any model $(M, \leadsto)$ and any sentence $\pi_1 \Rightarrow \pi_2$ is defined as follows: $(M, \leadsto) \models_{\Sigma}^{\mathbf{ARL}(\underline{\mathrm{M}})} \pi_1 \Rightarrow \pi_2$ iff for every $m \in \llbracket M \rrbracket_{\Sigma,c}$ such that $M(\models^{\underline{\mathrm{M}}})^{m}_{\Sigma,c}\pi_1$, there exists $n \in \llbracket M \rrbracket_{\Sigma,c}$ such that $M(\models^{\underline{\mathrm{M}}})^{n}_{\Sigma,c}\pi_2$, and $m \leadsto_c n$.

Corollary 1. The tuple $\mathbf{ARL}(\underline{\mathrm{M}}) = (\mathbb{S}\mathrm{ig}^{\mathbf{ARL}(\underline{\mathrm{M}})}, \mathrm{Sen}^{\mathbf{ARL}(\underline{\mathrm{M}})}, \mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}, \models^{\mathbf{ARL}(\underline{\mathrm{M}})})$ is a pre-institution.

We now focus on proving the satisfaction condition, showing that the direct implication of the equivalence holds unconditionally.

Proposition 10. For every signature morphism $\phi\colon \Sigma \to \Sigma'$, every class $c \in \mathrm{Cls}(\Sigma)$, every model $(M', \rightsquigarrow') \in |\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\Sigma')|$, and every sentence $\pi_1 \Rightarrow \pi_2$,

$$(M', \rightsquigarrow') \models^{\mathbf{ARL}(\underline{\mathrm{M}})}_{\Sigma')} \phi(\pi_1 \Rightarrow \pi_2) \text{ implies } (M', \rightsquigarrow')\restriction_\phi \models^{\mathbf{ARL}(\underline{\mathrm{M}})}_{\Sigma} \pi_1 \Rightarrow \pi_2.$$

Proof. Let c be the class of π_1 and π_2, and $m \in [\![M'\restriction_\phi]\!]_{\Sigma,c}$ such that $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{m}_{\Sigma,c}\pi_1$. We have to show that there exists $n \in [\![M'\restriction_\phi]\!]_{\Sigma,c}$, such that $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{n}_{\Sigma,c}\pi_2$ and $m \rightsquigarrow_c n$. From the surjectivity of $[\![M']\!]_{\phi,c}$, there exists $m' \in [\![M']\!]_{\Sigma',\phi(c)}$ such that $[\![M']\!]_{\phi,c}(m') = m$, and thus $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{[\![M']\!]_{\phi,c}(m')}_{\Sigma,c}\pi_1$. From the satisfaction condition in $\underline{\mathrm{M}}$ it follows that $M'(\models^{\underline{\mathrm{M}}})^{m'}_{\Sigma',\phi(c)}\phi(\pi_1)$. According to the hypothesis, $(M', \rightsquigarrow') \models^{\mathbf{ARL}(\underline{\mathrm{M}})}_{\Sigma'} \phi(\pi_1) \Rightarrow \phi(\pi_2)$, and hence, from the definition of the satisfaction relation of $\mathbf{ARL}(\underline{\mathrm{M}})$, there exists $n' \in [\![M']\!]_{\Sigma',\phi(c)}$ such that $M'(\models^{\underline{\mathrm{M}}})^{n'}_{\Sigma',\phi(c)}\phi(\pi_2)$ and $m' \rightsquigarrow'_{\phi(c)} n'$. By applying once more the satisfaction condition in $\underline{\mathrm{M}}$, we have $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{[\![M']\!]_{\phi(c)}(n')}_{\Sigma,c}\pi_2$. By choosing $n = [\![M']\!]_{\phi,c}(n')$, from the fact that $m' \rightsquigarrow'_{\phi(c)} n'$, we obtain $m \to_c n$, and thus $m \rightsquigarrow_c n$. □

The converse of Proposition 10 holds if the stratification of the underlying institution of $\mathbf{ARL}(\underline{\mathrm{M}})$ satisfies a property similar to that of lifting relations from [7, Chapter 9].

Proposition 11. If in $\mathbf{ARL}(\underline{\mathrm{M}})$, for every signature morphism $\phi\colon \Sigma \to \Sigma'$, every class $c \in \mathrm{Cls}(\Sigma)$, every Σ'-model $(M', \rightsquigarrow')$, and every states $m' \in [\![M']\!]_{\Sigma',\phi(c)}$ and $n \in [\![M'\restriction_\phi]\!]_{\Sigma,c}$ such that $[\![M']\!]_{\phi,c}(m') \rightsquigarrow_c n$, there exists $n' \in [\![M']\!]_{\Sigma',\phi(c)}$ such that $m' \rightsquigarrow'_{\phi(c)} n'$ and $[\![M']\!]_{\phi,c}(n') = n$, then

$$(M', \rightsquigarrow')\restriction_\phi \models^{\mathbf{ARL}(\underline{\mathrm{M}})}_{\Sigma} \pi_1 \Rightarrow \pi_2 \text{ implies } (M', \rightsquigarrow') \models^{\mathbf{ARL}(\underline{\mathrm{M}})}_{\Sigma'} \phi(\pi_1 \Rightarrow \pi_2),$$

for every sentence $\pi_1 \Rightarrow \pi_2$.

Proof. Let c be the class of π_1 and π_2, and $m' \in [\![M']\!]_{\Sigma',\phi(c)}$ such that $M'(\models^{\underline{\mathrm{M}}})^{m'}_{\Sigma',\phi(c)}\phi(\pi_1)$. We have to prove that there exists $n' \in [\![M']\!]_{\Sigma',\phi(c)}$, such that $M'(\models^{\underline{\mathrm{M}}})^{n'}_{\Sigma',\phi(c)}\phi(\pi_2)$ and $m' \rightsquigarrow'_{\phi(c)} n'$. From the satisfaction condition in $\underline{\mathrm{M}}$, we have $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{[\![M']\!]_{\phi,c}(m')}_{\Sigma,c}\pi_1$. As $(M', \rightsquigarrow') \models^{\mathbf{ARL}(\underline{\mathrm{M}})}_{\Sigma} \pi_1 \Rightarrow \pi_2$ and $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{m}_{\Sigma,c}\pi_1$ for $m = [\![M']\!]_{\phi,c}(m')$, we deduce that there exists $n \in [\![M'\restriction_\phi]\!]_{\Sigma,c}$ such that $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{n}_{\Sigma,c}\pi_2$ and $m \rightsquigarrow n$. From the surjectivity of $[\![M']\!]_{\Sigma',\phi(c)}$ it follows that there exists $n' \in [\![M']\!]_{\Sigma',\phi(c)}$ such that $[\![M']\!]_{\phi,c}(n') = n$, and hence $M'\restriction_\phi(\models^{\underline{\mathrm{M}}})^{[\![M']\!]_{\phi,c}(n')}_{\Sigma,c}\pi_2$. By applying the satisfaction condition in $\underline{\mathrm{M}}$ we obtain that $M'(\models^{\underline{\mathrm{M}}})^{n'}_{\Sigma',\phi(c)}\phi(\pi_2)$. □

Corollary 2. If the stratified institution with classes $\underline{\mathrm{M}}$ satisfies the hypothesis of Proposition 11, then $\mathbf{ARL}(\underline{\mathrm{M}})$ is an institution.

Remark 3. In most concrete examples of stratified institutions with classes the natural transformations $[\![M']\!]_{\phi,c}$ of the stratification are bijective, or even identities (see for example the definitions of $\underline{\mathrm{ML}}^{+}$ and $\underline{\mathrm{CTL}}$). Therefore, the hypothesis of Proposition 11 is usually satisfied, entailing that $\mathbf{ARL}(\underline{\mathrm{M}})$ is an institution. However, we note that the aforementioned hypothesis does not imply the injectivity (and bijectivity) of the maps $[\![M']\!]_{\phi,c}$. Even though by requiring these maps to be bijective we are imposing more restrictive conditions than what would be necessary to insure that $\mathbf{ARL}(\underline{\mathrm{M}})$ is an institution, we still choose to do so from now on because the property is satisfied in all of our examples and it brings significant technical advantages in the developments that follow.

We have hitherto defined only an atomic fragment of the desired institution of abstract reachability logic. To describe the construction of the institution with universally quantified Horn-clause sentences over the atomic sentences of $\mathbf{ARL}(\underline{\mathrm{M}})$, we use the notion of quantification space originating from [8].

Definition 7. For any category $\mathbb{S}\mathrm{ig}$ a class of arrows $\mathcal{D} \subseteq \mathbb{S}\mathrm{ig}$ is called a *quantification space* if, for any $\chi \colon \Sigma \to \Sigma' \in \mathcal{D}$ and $\varphi \colon \Sigma \to \Sigma_1$ there exists a designated pushout

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma_1 \arrow[d, "\chi(\varphi)"] \\
\Sigma' \arrow[r, "{\varphi[\chi]}"'] & \Sigma'_1
\end{tikzcd}$$

with $\chi(\varphi) \in \mathcal{D}$ and such that the horizontal composition of these designated pushouts is also a designated pushout, i.e. for the pushouts in the diagram below

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma_1 \arrow[r, "\theta"] \arrow[d, "\chi(\varphi)"] & \Sigma_2 \arrow[d, "\chi(\varphi)(\theta)"] \\
\Sigma' \arrow[r, "{\varphi[\chi]}"'] & \Sigma'_1 \arrow[r, "{\theta[\chi(\varphi)]}"'] & \Sigma'_1
\end{tikzcd}$$

$\varphi[\chi]\,;\theta[\chi(\varphi)] = (\varphi\,;\theta)[\chi]$ and $\chi(\varphi)(\theta) = \chi(\varphi\,;\theta)$, and such that $\chi(1_\Sigma) = \chi$ and $1_\Sigma[\chi] = 1_{\Sigma'}$. A quantification space $\mathcal{D}$ for $\mathbb{S}\mathrm{ig}$ is *adequate* for a functor $\mathrm{Mod} \colon \mathbb{S}\mathrm{ig}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ when the aforementioned designated pushouts are weak amalgamation squares for Mod. A quantification space $\mathcal{D}$ for $\mathbb{S}\mathrm{ig}$ is called *adequate* for an institution if it is adequate for its model functor.

Proposition 12. For any institution $\mathbf{I}$ with an adequate quantification space $\mathcal{D}$, the following data defines an institution, called the *institution of universally $\mathcal{D}$-quantified Horn clauses over* $\mathbf{I}$, and denoted $\mathbf{HCL}(\mathbf{I})$:

- $\mathbb{S}\mathrm{ig}^{\mathbf{HCL}(\mathbf{I})} = \mathbb{S}\mathrm{ig}^{\mathbf{I}}$,
- $\mathrm{Mod}^{\mathbf{HCL}(\mathbf{I})} = \mathrm{Mod}^{\mathbf{I}}$,

- $\mathrm{Sen}^{\mathbf{HCL(I)}}(\Sigma) = \{\forall\chi.\rho'_1 \wedge \ldots \wedge \rho'_n \to \rho' \mid (\chi\colon \Sigma \to \Sigma') \in \mathcal{D} \text{ and } \rho'_i, \rho' \in \mathrm{Sen}^{\mathbf{I}}(\Sigma')\}$, for every signature Σ,
- $\mathrm{Sen}^{\mathbf{HCL(I)}}(\varphi)(\forall\chi.\rho'_1 \wedge \ldots \wedge \rho'_n \to \rho') = \forall\chi(\varphi).\mathrm{Sen}^{\mathbf{I}}(\varphi[\chi])(\rho'_1) \wedge \ldots \wedge \mathrm{Sen}^{\mathbf{I}}(\varphi[\chi])(\rho'_n) \to \mathrm{Sen}^{\mathbf{I}}(\varphi[\chi])(\rho')$, for every signature morphism $\varphi\colon \Sigma \to \Sigma_1$,
- $M \models^{\mathbf{HCL(I)}}_{\Sigma} \forall\chi.\rho'_1 \wedge \ldots \wedge \rho'_n \to \rho'$ iff for all χ-expansions M' of M, $M' \models^{\mathbf{I}}_{\Sigma'} \rho'$ whenever $M' \models^{\mathbf{I}}_{\Sigma'} \rho'_i$, for $i = \overline{1,n}$.

In order to build an institution of universally quantified sentences over $\mathbf{ARL}(\underline{\mathrm{M}})$ as described in Proposition 12, we need to ensure that $\mathbf{ARL}(\underline{\mathrm{M}})$ satisfies its hypothesis. This cannot be guaranteed in general, because $\underline{\mathrm{M}}$ is abstract. Nevertheless, we can obtain an appropriate set of hypotheses for the underlying stratified institution $\underline{\mathrm{M}}$ that allow us to apply Proposition 12:

- the existence of a quantification space for $\mathbf{ARL}(\underline{\mathrm{M}})$ is guaranteed by the existence of a quantification space for $\underline{\mathrm{M}}$, as the categories $\mathbb{S}\mathrm{ig}^{\mathbf{ARL}(\underline{\mathrm{M}})}$ and $\mathbb{S}\mathrm{ig}^{\underline{\mathrm{M}}}$ are equal,
- that fact that $\mathbf{ARL}(\underline{\mathrm{M}})$ has weak model amalgamation follows from the weak model amalgamation property of $\underline{\mathrm{M}}$ (see Definition 8 below) and the preservation of pushouts by the class functor of $\underline{\mathrm{M}}$ (see Proposition 13 below).

Definition 8. A stratified institution with classes $\underline{\mathrm{M}}$ has *(weak) model amalgamation* whenever its corresponding institution $\flat\mathbf{M}$ has this property.

Proposition 13. For any stratified institution with classes $\underline{\mathrm{M}}$ having (weak) model amalgamation, if its Cls functor preserves pushouts, $\mathbf{ARL}(\underline{\mathrm{M}})$ has (weak) model amalgamation.

Proof. Consider the following pushout square in $\mathbb{S}\mathrm{ig}^{\mathbf{ARL}(\underline{\mathrm{M}})}$, i.e. in $\mathbb{S}\mathrm{ig}^{\underline{\mathrm{M}}}$.

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Sigma_1 \arrow[d, "\theta_1"] \\
\Sigma_2 \arrow[r, "\theta_2"'] & \Sigma'
\end{tikzcd}$$

We have to show that for each Σ_1-model $(M_1, \rightsquigarrow_1)$ and Σ_2-model $(M_2, \rightsquigarrow_2)$ such that $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\varphi_1)(M_1, \rightsquigarrow_1) = \mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\varphi_2)(M_2, \rightsquigarrow_2)$, there exists a Σ'-model $(M', \rightsquigarrow')$ so that $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\theta_1)(M', \rightsquigarrow') = (M_1, \rightsquigarrow_1)$ and $\mathrm{Mod}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\theta_2)(M', \rightsquigarrow') = (M_2, \rightsquigarrow_2)$.

From the definition of the model reduct in $\mathbf{ARL}(\underline{\mathrm{M}})$ and from the hypothesis that $\underline{\mathrm{M}}$ has (weak) model amalgamation it follows that there exists a Σ'-model M' in $\underline{\mathrm{M}}$ such that $\mathrm{Mod}^{\underline{\mathrm{M}}}(\theta_1)(M') = M_1$ and $\mathrm{Mod}^{\underline{\mathrm{M}}}(\theta_2)(M') = M_2$. Therefore, we only need to focus on the preorders that need to be defined over the stratifications of $\underline{\mathrm{M}}$ in order to obtain an $\mathbf{ARL}(\underline{\mathrm{M}})$ model. This means that we need to define a family of preorders $\rightsquigarrow'$ on the stratifications of M' based on $\rightsquigarrow_1$ and $\rightsquigarrow_2$.

Consider a class $c' \in \mathrm{Cls}(\Sigma')$. Since Cls preserves pushouts we deduce that there exists a class $c_i \in \mathrm{Cls}(\Sigma_i)$, for some $i \in \{1, 2\}$, such that $\theta_i(c_i) = c'$. We recall that, by hypothesis, the function $[\![M']\!]_{\theta_i,c_i}$ is bijective. This allows us to define $\rightsquigarrow'_{c'} = [\![M']\!]^{-1}_{\theta_i,c_i}(\rightsquigarrow_{ic_i})$. Note that this relation is well-defined. First of all, it is a preorder. And if $c' = \theta_1(c_1) = \theta_2(c_2)$, there exists $c \in \mathrm{Cls}(\Sigma)$ such that $c_1 = \varphi_1(c)$, $c_2 = \varphi_2(c)$; since $(M_1, \rightsquigarrow_1)$ and $(M_2, \rightsquigarrow_2)$ have the same Σ reduct it follows that $\rightsquigarrow_{1c_1} = [\![M_1]\!]^{-1}_{\varphi_1,c}(\rightsquigarrow_c)$, $\rightsquigarrow_{2c_2} = [\![M_2]\!]^{-1}_{\varphi_2,c}(\rightsquigarrow_c)$. Therefore, $[\![M']\!]^{-1}_{\theta_1,c_1}(\rightsquigarrow_{1c_1}) = [\![M']\!]^{-1}_{\theta_2,c_2}(\rightsquigarrow_{2c_2})$. We have thus obtained a Σ'-model $(M', \rightsquigarrow')$ that is obviously a Σ'-expansion of $(M_1, \rightsquigarrow_1)$ and $(M_2, \rightsquigarrow_2)$. Moreover, $(M', \rightsquigarrow')$ is unique with this property if $\underline{\mathrm{M}}'$ is the unique Σ'-expansion of M_1 and M_2 in $\underline{\mathrm{M}}$. □

Corollary 3. For any stratified institution with classes $\underline{\mathrm{M}}$ having an adequate quantification space and a pushout preserving class functor, $\mathbf{HCL}(\mathbf{ARL}(\underline{\mathrm{M}}))$ is an institution.

4.2 Defining Reachability over Matching Logic

In order to capture reachability logic in its original, concrete form, we must instantiate the parameter of the institution $\mathbf{HCL}(\mathbf{ARL}(\underline{\mathrm{M}}))$ defined above, with the stratified institution $\underline{\mathrm{ML}}^+$. To this end, we first prove that by adding variables as deterministic constants to the signatures of $\underline{\mathrm{ML}}^+$ we obtain a quantification space. Furthermore, to show that the quantification space is adequate, we use the property of model amalgamation of the comorphism (Φ, α, β) between $\flat\mathbf{ML}^+$ and $\mathbf{FOL}$ defined in Section 3.3.

Proposition 14. $\underline{\mathrm{ML}}^+$ has pushouts of signatures. Moreover, its class functor preserves pushouts.

Proof (sketch). We recall that the signatures of $\underline{\mathrm{ML}}^+$ are tuples (S, F, D), where (S, F), and (S, D) are algebraic signatures such that $F_{w\to s} \cap D_{w\to s} = \emptyset$. Let $\varphi_1 \colon (S, F, D) \to (S_1, F_1, D_1)$, and $\varphi_2 \colon (S, F, D) \to (S_2, F_2, D_2)$ be two signature morphisms in $\mathbb{S}\mathrm{ig}^{\mathbf{ML}^+}$. They determine two pairs of morphisms of algebraic signatures, $\varphi_i^F \colon (S, F) \to (S_i, F_i)$, $\varphi_i^D \colon (S, D) \to (S_i, D_i)$, for $i \in \{1, 2\}$, that agree on sorts – the signatures have the same sorts, and the morphisms are defined in the same way on sorts: $(\varphi_i^F)^{\mathrm{st}} = (\varphi_i^D)^{\mathrm{st}}$. According to the construction of pushouts of algebraic signatures, we can choose the pushouts (θ_1^F, θ_2^F), and (θ_1^D, θ_2^D) of $(\varphi_1^F, \varphi_2^F)$, and $(\varphi_1^D, \varphi_2^D)$ respectively,

$$\begin{tikzcd}
(S,F) \arrow[r, "\varphi_1^F"] \arrow[d, "\varphi_2^F"'] & (S_1,F_1) \arrow[d, "\theta_1^F"] \\
(S_2,F_2) \arrow[r, "\theta_2^F"'] & (S',F')
\end{tikzcd}
\qquad
\begin{tikzcd}
(S,D) \arrow[r, "\varphi_1^D"] \arrow[d, "\varphi_2^D"'] & (S_1,D_1) \arrow[d, "\theta_1^D"] \\
(S_2,D_2) \arrow[r, "\theta_2^D"'] & (S',D')
\end{tikzcd}$$

such that $(\theta_i^F)^{\mathrm{st}} = (\theta_i^D)^{\mathrm{st}}$; this follows from the fact that pushouts can be first built on sorts. Moreover, the families F', D' of sets of operation symbols can be chosen such that $F'_{w'\to s'} \cap D'_{w'\to s'} = \emptyset$, for each $w' \in S'^*, s' \in S'$. In this way, the category of $\underline{\mathrm{ML}}^+$ signatures inherits the pushouts of algebraic signatures. In addition, Cls preserves pushouts because the forgetful functor from the category of algebraic signatures to $\mathbb{S}\mathrm{et}$ that maps every signature to its set of sorts preserves pushouts. □

Example. Let us consider the $\mathbb{K}$ definition of the IMP programming language. By splitting the syntax module into three modules, AExp, BExp and IMP-SYNTAX importing the two expressions modules, we have an immediate and natural example of a pushout of signatures: as both the AExp and BExp modules import the BUILT-IN module containing the built-in sorts and corresponding operations of $\mathbb{K}$, we need to construct the pushout of their signatures in order to obtain the signature of the module IMP-SYNTAX.

$$\begin{tikzcd}
\text{BUILT-IN}^{\mathbb{S}\mathrm{ig}} \arrow[r, "\subseteq"] \arrow[d, "\subseteq"'] & \text{AExp}^{\mathbb{S}\mathrm{ig}} \arrow[d, "\subseteq"] \\
\text{BExp}^{\mathbb{S}\mathrm{ig}} \arrow[r, "\subseteq"'] & \text{IMP-SYNTAX}^{\mathbb{S}\mathrm{ig}}
\end{tikzcd}$$

Proposition 15. In $\underline{\mathrm{ML}}^+$, the family of extensions with deterministic constants forms a quantification space.

Proof. Since we know that the signature extensions with a finite set of variables form a quantification space in the case of the institution of total algebras [8], we can easily conclude that the $\underline{\mathrm{ML}}^+$ signature extensions $\chi\colon (S, F, D) \hookrightarrow (S, F, D \uplus \{x : s\})$ constitute a quantification space for $\mathbb{S}\mathrm{ig}^{\underline{\mathrm{ML}}^+}$: for any signature morphism $\varphi\colon (S, F, D) \to (S_1, F_1, D_1)$,

- the inclusions $\chi(\varphi)\colon (S_1, F_1, D_1) \hookrightarrow (S_1, F_1, D_1 \uplus \{x : \varphi^{\mathrm{st}}(s)\})$ and
- the canonical extensions $\varphi[\chi]$ of φ that map the variable $x : s$ to $x : \varphi^{\mathrm{st}}(s)$

define a pushout square satisfying the properties of Definition 7. □

The following definition originates from [5].

Definition 9. An institution comorphism $(\Phi, \alpha, \beta)\colon \mathbf{I} \to \mathbf{I}'$ *has weak model amalgamation* if for every $\mathbf{I}$-signature morphism $\varphi\colon \Sigma \to \Sigma'$, every Σ'-model M', and every $\Phi(\Sigma)$-model N such that $\beta_\Sigma(N) = M'\restriction_\varphi$, there exists a $\Phi(\Sigma')$-model N' such that $\beta_{\Sigma'}(N') = M'$ and $N'\restriction_{\Phi(\varphi)} = N$. We say that $(\Phi, \alpha, \beta)\colon \mathbf{I} \to \mathbf{I}'$ *has model amalgamation* when N' is required to be unique.

Proposition 16. $\underline{\mathrm{ML}}^+$ has model amalgamation.

Proof. Let us first note that the comorphism (Φ, α, β) between the institution $\flat\mathbf{ML}^+$ and **FOL** defined in the previous section has model amalgamation. This property holds trivially since the model reduction functors β_Σ are isomorphisms of categories, for every signature Σ. As the institution of **FOL** also has model amalgamation, we can use a general result of institution theory to deduce that $\flat\mathbf{ML}^+$, and implicitly $\underline{\mathrm{ML}}^+$, has model amalgamation. □

Corollary 4. $\mathbf{HCL}(\mathbf{ARL}(\mathbf{ML}^+))$ is an institution.

Example. We can now rewrite the $\mathbb{K}$ definition of our imperative programming language IMP into a specification over $\mathbf{HCL}(\mathbf{ARL}(\mathbf{ML}^+))$ using a HetCasl-like syntax.

Listing 4.1: The IMP programming language (HetCasl)

```
spec IMP_SYNTAX =
     BUILT_IN and LIST[sort Id] then
     free type AExp ::= sort Int | sort Id
            | ___/___(AExp, AExp)
            | ___+___(AExp, AExp)
            | (___)(AExp)
     free type BExp ::= sort Bool
            | ___ ≤ ___(AExp, AExp)
            | !___(BExp)
            | ___&&___(BExp, BExp)
            | (___)(BExp)
     free types Block ::= {} | {___}(Stmt);
                Stmt ::= sort Block
            | ___=___;(Id, AExp)
            | if (___)___else___(BExp, Block, Block)
            | while(___)___(BExp, Block)
            | ______(Stmt, Stmt)
     free types Pgm ::= int___;___(Ids, Stmt);
                Ids   ::= sort List[Id]
end

spec IMP =
     IMP_SYNTAX then
     free type  KResult ::= sort Int | sort Bool
     free types K        ::= ⟨k⟩___⟨/k⟩(Pgm);
                State    ::= ⟨state⟩___⟨/state⟩(Map);
                T        ::= ⟨t⟩___ ___⟨/t⟩(K, State)

% AExp
     ∀ X:Id; I:Int
     • ⟨t⟩ ⟨k⟩ X ⋯⟨/k⟩ ⟨state⟩⋯ X ↦ I ⋯⟨/state⟩ ⟨/t⟩
       ⇒
       ⟨t⟩ ⟨k⟩ I ⋯⟨/k⟩ ⟨state⟩⋯ X ↦ I ⋯⟨/state⟩ ⟨/t⟩
     ∀ I1, I2:Int
     • I2  ≠Int  0 → (I1 / I2) ⇒ (I1 /Int I2)
     ∀ I1, I2:Int
     • I1 + I2 ⇒ I1 +Int I2
% BExp
     ∀ I1, I2:Int
     • I1  ≤  I2 ⇒ I1  ≤ Int I2
     ∀ T:Bool
     • ! T ⇒ notBoolT
     ∀ B:Bool
     • true && B ⇒B
     ∀ B:Bool
     •  false && B ⇒ false
```

```
% Block
    • {} ⇒ .
    ∀ S:Stmt
    • {S} ⇒ S
% Stmt
    ∀ X:Id; I, J:Int
    • ⟨t⟩ ⟨k⟩ X = I; ···⟨/k⟩ ⟨state⟩··· X ↦ J ···⟨/state⟩ ⟨/t⟩
        ⇒
      ⟨t⟩ ⟨k⟩ . ···⟨/k⟩ ⟨state⟩··· X ↦ I ···⟨/state⟩ ⟨/t⟩
    ∀ S1, S2:Stmt
    • S1 S2 ⇒ S1 ↝ S2
    ∀ S1, S2:Block
    • if (true) S1 else S2 ⇒ S1
    ∀ S1, S2:Block
    • if (false) S1 else S2 ⇒ S2
    ∀ S:Block; B:BExp
    • while (B) S ⇒ if (B) {S while (B) S} else {}
% Pgm
    ∀ X:Id; Xs:Ids; Rho:Map
    • notBool(X in keys(Rho)) →
      ⟨t⟩ ⟨k⟩ int (X, Xs); ···⟨/k⟩ ⟨state⟩ Rho .Map ⟨/state⟩ ⟨/t⟩
      ⇒
      ⟨t⟩ ⟨k⟩ int (Xs); ···⟨/k⟩ ⟨state⟩ Rho X ↦ 0 ⟨/state⟩ ⟨/t⟩
    ∀ S:Stmt
    • int .Ids; S ⇒ S
end
```

4.3 Relationship to First-Order Logic

For any institution **ARL**($\underline{\mathrm{M}}$) defined over a stratified institution with classes $\underline{M}$, there exists a comorphism of institutions between **ARL**($\underline{\mathrm{M}}$) and $\mathbf{FOL}^{\mathrm{pres}}$, the institution of presentations over first-order logic whenever there exists a comorphism of institutions (Φ, α, β) between $\flat\mathbf{M}$ and **FOL** such that:

- the classes of a signature in $\mathbb{S}\mathrm{ig}^{\flat\mathbf{M}}$ are given by the sorts of its translation to **FOL**: $\mathrm{Cls} = \Phi \,;\mathrm{St}$, where St is the forgetful functor $\mathrm{St}\colon \mathbb{S}\mathrm{ig}^{\mathbf{FOL}} \to \mathbb{S}\mathrm{et}$,
- for every signature Σ, $\alpha_\Sigma(\pi) = \forall m : s.\mathrm{FOL}^{m\,:\,s}_{\Sigma}(\pi)$, for every sentence π of class s,[1]
- for every model $N \in |\mathrm{Mod}^{\mathbf{FOL}}(\Phi(\Sigma))|$, and every $s \in \mathrm{Cls}(\Sigma)$, $N_s = [\![\beta_\Sigma(N)]\!]_{\Sigma,s}$.

We define $(\Phi^R, \alpha^R, \beta^R)\colon \mathbf{ARL}(\underline{\mathrm{M}}) \to \mathbf{FOL}^{\mathrm{pres}}$ as follows:

[1] Note that, in this case, $\mathrm{FOL}^{m\,:\,s}_{\Sigma}(\pi)$ is just a notation, and it should not be confused with the first-order sentence described in the previous section, for which we would need to instantiate $\underline{M}$ with $\underline{\mathrm{ML}}^{+}$.

For signatures: The signature functor $\Phi^R\colon \mathbb{S}\mathrm{ig}^{\mathbf{ARL}(\underline{\mathrm{M}})} \to \mathbb{S}\mathrm{ig}^{\mathbf{FOL}^{\mathrm{pres}}}$ maps every signature Σ of $\mathbf{ARL}(\underline{\mathrm{M}})$ to

$$\Phi^R(\Sigma) = (\mathrm{Reach}(\Sigma), E),$$

where

- $\mathrm{Reach}(\Sigma)$ denotes the first-order signature obtained by adding to $\Phi(\Sigma) = (S', F', P')$ a predicate *reach* of arity $s\,s$ for every sort $s \in S'$, and
- E is a set of axioms that define the predicates *reach* as preorders:

$$E = \{\forall x : s.\mathit{reach}(x, x), \forall x, y, z : s.\mathit{reach}(x, y) \wedge \mathit{reach}(y, z) \to \mathit{reach}(x, z) \mid s \in S'\}.$$

For sentences: For every signature Σ of $\mathbf{ARL}(\underline{\mathrm{M}})$, the sentence translation function $\alpha^R_\Sigma\colon \mathrm{Sen}^{\mathbf{ARL}(\underline{\mathrm{M}})}(\Sigma) \to \mathrm{Sen}^{\mathbf{FOL}}(\mathrm{Reach}(\Sigma))$ maps every Σ-sentence $\pi_1 \Rightarrow \pi_2$ to

$$\alpha^R_\Sigma(\pi_1 \Rightarrow \pi_2) = \forall m : s.\mathrm{FOL}^{m : s}_\Sigma(\pi_1) \to \exists n : s.\mathrm{FOL}^{n : s}_\Sigma(\pi_2) \wedge \mathit{reach}(m, n).$$

For models: For every signature Σ, $\beta^R_\Sigma\colon \mathrm{Mod}^{\mathbf{FOL}}(\mathrm{Reach}(\Sigma), E) \to \mathrm{Mod}^{\mathbf{ARL}(M)}(\Sigma)$ is the functor that maps every first-order structure $N \in |\mathrm{Mod}^{\mathbf{FOL}}(\mathrm{Reach}(\Sigma), E)|$ to the model $(M, \rightsquigarrow) \in |\mathrm{Mod}^{\mathbf{ARL}(M)}(\Sigma)|$, given by $M = \beta_\Sigma(N) \in |\mathrm{Mod}^{\underline{\mathrm{M}}}(\Sigma)|$ and $\rightsquigarrow_s = \{(m, n) \mid (N, m, n) \models \mathit{reach}(x, y)\}$, for every sort s. Note that the preorder $\rightsquigarrow_s$ is well-defined as $\mathrm{Cls} = \Phi\,;\mathrm{St}$ and $N_s = [\![M]\!]_{\Sigma,s}$.

Proposition 17. $(\Phi^R, \alpha^R, \beta^R)$ is a comorphism of institutions.

To encode the Horn-clause reachability logic of Corollary 4 (defined over $\underline{\mathrm{ML}}^+$) into first-order logic, it suffices to notice that the comorphism $(\Phi, \alpha, \beta)\colon \flat\mathbf{ML}^+ \to \mathbf{FOL}$ considered in Section 3.3 satisfies all of the above requirements, and thus can be extended to a comorphism $(\Phi^R, \alpha^R, \beta^R)\colon \mathbf{ARL}(\underline{\mathrm{M}}) \to \mathbf{FOL}^{\mathrm{pres}}$. This comorphism can be further extended to an encoding of $\mathbf{HCL}(\mathbf{ARL}(\underline{\mathrm{ML}}^+))$ into $\mathbf{FOL}^{\mathrm{pres}}$ through the use of a general results about Horn-clause institutions (described in Proposition 18 below).

Proposition 18. Let $\mathbf{I}$ and $\mathbf{I}'$ be institutions equipped with quantification spaces. Every comorphism of institutions $(\Phi, \alpha, \beta)\colon \mathbf{I} \to \mathbf{I}'$ that has weak model amalgamation, and for which Φ preserves the quantification space of $\mathbf{I}$, can be extended to a comorphism of institutions between $\mathbf{HCL}(\mathbf{I})$ and $\mathbf{HCL}(\mathbf{I}')$.

5 Conclusions

In this thesis, we proposed an institutional formalisation of the logical systems that underlie the $\mathbb{K}$ semantic framework. These logical systems account for the structural properties of program configurations (through matching logic), and changes of these configurations (through reachability logic).

Since reachability relies on matching, our first results focus on the formalisation of matching logic. To this purpose, we extended the notion of stratified institution with classes, thus allowing us to integrate, at the abstract level, the classification of matching-logic patterns according to their sorts, and the states associated with the models and satisfaction relation of matching logic. Building on these results, we developed an abstract form of reachability logic over an arbitrary stratified institution with classes subject to a few elementary properties, namely model amalgamation, the existence of a quantification space and the preservation of pushouts of signatures with respect to classes. We proved that the stratified institution of matching logic smoothly satisfies these properties and we used it to derive the institution of reachability logic. Finally, for both matching and reachability, we defined comorphisms to the institution of first-order logic.

Our work sets the foundation for integrating the $\mathbb{K}$ semantic framework into heterogeneous institution-based toolsets like HETS, allowing us to exploit the combined potential of the $\mathbb{K}$ tool and of other software tools such as the MiniSat solver, the SPASS automated prover or the Isabelle interactive proof assistant. Another line of research concerns the development of $\mathbb{K}$ from a purely formal-specification perspective, including for example, studies on modularisation and initial semantics. Within this context, verification can be performed based on the proof systems that have already by defined for $\mathbb{K}$.

Bibliography

[1] The IMP language. http://www.kframework.org/imgs/releases/k/tutorial/1_k/2_imp/lesson_5/imp.pdf.

[2] Marc Aiguier and Răzvan Diaconescu. Stratified institutions and elementary homomorphisms. *Information Processing Letters*, 103(1):5–13, 2007.

[3] Kenneth Jon Barwise. Axioms for abstract model theory. *Annals of Mathematical Logic*, 7(2–3):221 – 265, 1974.

[4] Jean Yves Béziau. *Universal Logic: An Anthology: From Paul Hertz to Dov Gabbay.* Studies in Universal Logic. Birkhauser Verlag GmbH, 2012.

[5] Tomasz Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286(2):197–245, 2002.

[6] Traian Florin Şerbănuţă, Andrei Arusoaie, David Lazar, Chucky Ellison, Dorel Lucanu, and Grigore Roşu. The K primer (version 3.3). *Electronic Notes in Theoretical Computer Science*, 304(0):57–80, 2014.

[7] Răzvan Diaconescu. *Institution-independent Model Theory.* Studies in Universal Logic. Springer London, Limited, 2008.

[8] Răzvan Diaconescu. Quasi-boolean encodings and conditionals in algebraic specification. *Journal of Logic and Algebraic Programming*, 79(2):174–188, 2010.

[9] Joseph A. Goguen and Rod M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.

[10] Joseph A. Goguen and Razvan Diaconescu. An oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4(3):363–392, 1994.

[11] Joseph A. Goguen, Răzvan Diaconescu, and Petros Stefaneas. Logical support for modularisation. *Proceedings of Logical Environments*, pages 83–130, 1993.

[12] Yngve Lamo. *The Institution of Multialgebras–a general framework for algebraic software development.* PhD thesis, University of Bergen, 2003.

[13] Saunders Mac Lane. *Categories for the Working Mathematician.* Graduate Texts in Mathematics. Springer, 1998.

[14] José Meseguer. General logics. In M. Garrido D. Lascar H.-D. Ebbinghaus, J. Fernandez-Prida and M. Rodriquez Artalejo, editors, *Logic Colloquium'87 Proceedings of the Colloquium held in Granada*, volume 129 of *Studies in Logic and the Foundations of Mathematics*, pages 275 – 329. Elsevier, 1989.

[15] Till Mossakowski. HetCasl–heterogeneous specification. *Language summary*, 2004.

[16] Till Mossakowski, Christian Maeder, and Klaus Lüttich. The heterogeneous tool set, Hets. In Orna Grumberg and Michael Huth, editors, *TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer, 2007.

[17] Grigore Roşu. Matching logic: A logic for structural reasoning. Technical Report http://hdl.handle.net/2142/47004, University of Illinois, Jan 2014.

[18] Grigore Roşu and Traian Florin Şerbănuţă. An overview of the k semantic framework. *Journal of Logic and Algebraic Programming*, 79(6):397–434, 2010.

[19] Grigore Roşu and Traian Florin Şerbănuţă. K overview and SIMPLE case study. *Electronic Notes in Theoretical Computer Science*, 304(0):3–56, 2014.

[20] Grigore Roşu, Andrei Ştefănescu, Ştefan Ciobâcă, and Brandon M. Moore. One-path reachability logic. In *Proceedings of the 28th Symposium on Logic in Computer Science (LICS'13)*, pages 358–367. IEEE, June 2013.

[21] Antonino Salibra and Giuseppe Scollo. Interpolation and compactness in categories of pre-institutions. *Mathematical Structures in Computer Science*, 6(3):261–286, 1996.

[22] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2012.

[23] Andrzej Tarlecki. Moving between logical systems. In Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, editors, *COMPASS/ADT*, volume 1130 of *Lecture Notes in Computer Science*, pages 478–502. Springer, 1995.