From Conventional to Institution-Independent Logic Programming

Ionuţ Ţuţu*†[1,2] and José Luiz Fiadeiro†[1]

[1]*Department of Computer Science, Royal Holloway University of London*
[2]*Institute of Mathematics of the Romanian Academy, Research group of the project ID-3-0439*

August 2, 2014

We propose a logic-independent approach to logic programming through which the paradigm as we know it for Horn-clause logic can be explored for other formalisms. Our investigation is based on abstractions of notions such as logic program, clause, query, solution, and computed answer, which we develop over Goguen and Burstall's theory of institutions. These give rise to a series of concepts that formalize the interplay between the denotational and the operational semantics of logic programming. We examine properties concerning the satisfaction of quantified sentences, discuss a variant of Herbrand's theorem that is not limited in scope to any particular logical system or construction of logic programs, and describe a general resolution-based procedure for computing solutions to queries. We prove that this procedure is sound; moreover, under additional hypotheses that reflect faithfully properties of actual logic-programming languages, we show that it is also complete.

Keywords: Institution theory, Substitution systems, Logic programming, Herbrand's theorem, Resolution.

1 Introduction

The essential idea of exploring the computational aspects of logical inference as a foundation for the development of new programming paradigms was first investigated by Kowalski [30, 29] and Colmerauer [7], based on the pioneering work in proof theory of Herbrand [28] and on the introduction of resolution by Robinson [41] as an inference rule well suited for automation. Logic programming thus originated from the observation that a considerable fragment of first order logic has a natural computational interpretation that makes it adequate not only as a declarative language but also as a programming language. To be more precise, logic programming (*a*) defines computable functions by way of standard constructs from model theory, and (*b*) executes these definitions as programs through goal-directed deductions that are performed according to a fixed strategy. This means that logic programming provides, at the same time, both a denotational and an operational perspective on programs.

Conventional logic programming is semantically based upon the Horn-clause fragment of (single-sorted) first-order logic without equality, and implements deduction as backwards reasoning defined in terms of resolution steps [32]. However, the essence of the paradigm is to a great extent independent of any logical system of choice.

*Correspondence to: Department of Computer Science, Royal Holloway University of London.
†E-mail addresses: ittutu@gmail.com (I. Ţuţu), jose.fiadeiro@rhul.ac.uk (J. L. Fiadeiro).

This is reflected by the multitude of its variants that have been developed over time, such as order-sorted [25] and category-based equational logic programming [8], constraint logic programming [9], as well as higher-order [34, 35], and behavioural (i.e. object-oriented) logic programming [24]. Even more, recent developments in the semantics of service-oriented computing [16] have suggested a deep connection with logic programming; this connection was made explicit in [49] with the introduction of a service-oriented variant of the paradigm.

Through our inquiry we aim to provide a single unifying framework for the model-theoretic foundations of logic programming, with a high level of abstraction, that incorporates ideas from concrete instances of the phenomena, but without being committed to any particular logical system. To this end, our work is rooted in the institution theory of Goguen and Burstall [23]—one of the major fields of study in universal logic—and draws inspiration from previous institution-theoretic developments related to logic programming such as [45, 10], and also [34]. We propose an axiomatic approach to logic programming, and intrinsically to logic programs, that is based on a three-level hierarchy of concepts meant to capture

1. the denotational semantics of logic programming, based on a notion of *generalized substitution system* that extends the concept of institution with appropriate abstractions of variables, substitutions, local sentences, interpretations (of variables), and satisfaction, all parameterized by the signature used,

2. the operational semantics of logic programming, supported by a notion of *logic-programming framework* that describes clauses and queries—two of the most important syntactic structures used to define and execute logic programs—and also goal-directed rules that allow the integration of a general form of resolution, and

3. the various constructions of logic programs through a notion of *logic-programming language* that defines programs as abstract objects characterized by signatures, sets of axioms (clauses) and classes of models; this allows us to uniformly accommodate not only representations of logic programs as sets of clauses (over a fixed signature), as in [32], but also a great variety of module systems such as those reported, for example, in [25, 43, 4, 15].

Under this conceptual structure, the first main result of the present paper is a variant of Herbrand's theorem for abstract logic-programming languages, which links the denotational and the operational semantics of a given language by reducing the satisfiability of a query with respect to a logic program to the search for a suitable correct-answer substitution. In addition to the fact that it is grounded on the more general setting of abstract logic-programming languages, this upgrades the corresponding institution-independent result from [10] in a non-trivial manner by relying on hypotheses that are applicable in a considerably wider range of contexts; in this sense, the logic-programming semantics of services advanced in [50] provides a prime example of a logic-programming language that does not fit into the framework put forward in [10]. The second main contribution of our paper is the formulation of a sound procedure for searching for solutions to queries, which is based on the computation of a series of intermediate results (namely computed substitutions) by means of resolution. This procedure is also shown to be complete, but only in a conditional form, under assumptions that reflect well-known properties of actual logical systems such as relational or equational first-order logic.

The paper is organized as follows: In Section 2 we briefly recall some of the most important category- and institution-theoretic concepts, notations, and terminology that are necessary for the subsequent developments presented in this work. Section 3 reviews the foundations of conventional logic programming, with explicit emphasis on the abstraction of first-order variables and substitutions, and on the mappings induced by signature morphisms. Section 4 introduces the basic notion of generalized substitution system and details the conditions that guarantee the invariance of the satisfaction of universally- and existentially-quantified sentences (defined over a given generalized substitution system) under change of notation. Section 5 is dedicated to the development of abstract logic programming, encompassing the notions of logic-programming framework and language, as well as the main results of the paper. Finally, in Section 6 we discuss how another notable variant of logic programming, (many-sorted) equational logic programming, can be defined as an abstract logic-programming language.

2 Preliminaries

Categories

We assume the reader is familiar with basic notions of category theory such as category, functor, and natural transformation. With a few exceptions, we use the terminology and the notations from [31]. In this sense, we denote by $|\mathbb{C}|$ the collection of objects of a category $\mathbb{C}$, by $\mathbb{C}(A,B)$ the collection of arrows from A to B, by $f\,;g$ the composition of arrows f and g in diagrammatic order, and by 1_A the identity arrow of an object A.

A note on foundations. Most of the constructions described in this paper rely on large or very large collections of objects, and thus some care must be taken with respect to the set-theoretic foundations of category theory. One may consider for instance the hierarchy of set universes discussed in [31], in which every universe is closed under the usual set-theoretic constructions, contains the elements of its elements, and belongs to the next universe in the hierarchy. In this way, we obtain concepts such as category and functor for every level of the hierarchy, which means that we cannot define, for example, the category of all sets, but rather a category of sets for every universe. In terms of notation, given an arbitrary but fixed level of the hierarchy of set universes, we denote by $\mathbb{S}\mathrm{et}$ its corresponding category of sets and functions, and by $\mathbb{C}\mathrm{at}$ its category of categories and functors, which belongs, of course, to a higher set-theoretic universe.

The most important category-theoretic notions for our work are those of comma category, Grothendieck construction, and generalized subfunctor. We only recall here a number of definitions and elementary properties, mainly for fixing the terminology and the notation to be used throughout the paper. The interested reader can find more detailed presentations in canonical texts on category theory such as [31, 1], as well as in works like [21, 48] on the applications of category theory to algebraic specification.

Definition 2.1 (Comma category)**.** Given two functors $F_1\colon \mathbb{C}_1 \to \mathbb{K}$ and $F_2\colon \mathbb{C}_2 \to \mathbb{K}$ with the same codomain, the *comma category* $F_1 \,/\, F_2$ is the category in which (a) the *objects* are triples $\langle A_1, f, A_2\rangle$, where $A_1 \in |\mathbb{C}_1|$, $A_2 \in |\mathbb{C}_2|$, and $f\colon F_1(A_1) \to F_2(A_2)$ is an arrow in $\mathbb{K}$, (b) the *arrows* $\langle A_1, f, A_2\rangle \to \langle A_1', f', A_2'\rangle$ are pairs $\langle g_1, g_2\rangle$, where $g_1\colon A_1 \to A_1'$ and $g_2\colon A_2 \to A_2'$ are arrows in $\mathbb{C}_1$ and $\mathbb{C}_2$, respectively, such that $f\,;F_2(g_2) = F_1(g_1)\,;f'$, and ($c$) the *composition* of arrows is defined componentwise.

The comma category $F_1 \,/\, F_2$ is often denoted $\mathbb{C}_1 \,/\, F_2$, $F_1 \,/\, \mathbb{C}_2$ or simply $\mathbb{C}_1 \,/\, \mathbb{C}_2$ if F_1, F_2, or both of them, respectively, correspond to inclusions of categories. A special case arises when F_1 is the constant functor with value $A \in |\mathbb{K}|$, and F_2 is the identity of $\mathbb{K}$. Then we denote the comma category $F_1 \,/\, F_2$ by $A \,/\, \mathbb{K}$, and the forgetful functor $A \,/\, \mathbb{K} \to \mathbb{K}$ by $|\_|_A$.

Fact 2.2. Every arrow $h\colon A \to A'$ in a category $\mathbb{K}$ induces a left-composition functor $h \,/\, \mathbb{K}\colon A' \,/\, \mathbb{K} \to A \,/\, \mathbb{K}$ that maps every object $\langle A', f, B\rangle$ of $A' \,/\, \mathbb{K}$ to $\langle A, h\,;f, B\rangle$.

Definition 2.3 (Indexed category)**.** For any category $\mathbb{I}$ of *indices*, an $\mathbb{I}$-*indexed category* is a functor $\mathbb{C}\colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$.

Definition 2.4 (Grothendieck construction)**.** Any indexed category $\mathbb{C}\colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ can be 'flattened' to a *Grothendieck category* $\mathbb{C}^\sharp$ in which (a) the *objects* are pairs $\langle i, A\rangle$, with $i \in |\mathbb{I}|$ and $A \in |\mathbb{C}(i)|$, (b) the *arrows* $\langle i, A\rangle \to \langle i', A'\rangle$ are pairs $\langle u, f\rangle$ such that $u\colon i \to i'$ is an arrow in $\mathbb{I}$, and $f\colon A \to \mathbb{C}(u)(A')$ is an arrow in $\mathbb{C}(i)$, and (c) the *composition* of arrows $\langle u, f\rangle$ and $\langle u', f'\rangle$ is defined as $\langle u\,;u', f\,;\mathbb{C}(u)(f')\rangle$.

Fact 2.5. Every $\mathbb{I}$-*indexed functor* F between indexed categories $\mathbb{C}, \mathbb{D}\colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$, i.e. every natural transformation $F\colon \mathbb{C} \Rightarrow \mathbb{D}$, determines a functor $F^\sharp\colon \mathbb{C}^\sharp \to \mathbb{D}^\sharp$ that maps every object $\langle i, A\rangle$ of $\mathbb{C}^\sharp$ to $\langle i, F_i(A)\rangle$, and every arrow $\langle u, f\rangle\colon \langle i, A\rangle \to \langle i', A'\rangle$ of $\mathbb{C}^\sharp$ to $\langle u, F_i(f)\rangle$. We obtain in this way a 'flattening' functor $(\_)^\sharp$ from the category $[\mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}]$ of $\mathbb{I}$-indexed categories to $\mathbb{C}\mathrm{at}$.

Definition 2.6 (Category of functors)**.** For any category $\mathbb{K}$, the category $[\_ \to \mathbb{K}]^\sharp$ of *functors into* $\mathbb{K}$ is the Grothendieck category defined by the functor $[\_ \to \mathbb{K}]\colon \mathbb{C}\mathrm{at}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ that maps any category $\mathbb{C}$ to the functor category $[\mathbb{C} \to \mathbb{K}]$, and any functor $U\colon \mathbb{C} \to \mathbb{C}'$ to the left-composition functor $U\_\colon [\mathbb{C}' \to \mathbb{K}] \to [\mathbb{C} \to \mathbb{K}]$.

This means that the objects of $[\_ \to \mathbb{K}]^\sharp$ are in essence functors $F\colon \mathbb{C} \to \mathbb{K}$ into $\mathbb{K}$, and that the morphisms from $F\colon \mathbb{C} \to \mathbb{K}$ to $F'\colon \mathbb{C}' \to \mathbb{K}$ are pairs $\langle U, \tau\rangle$, where U is a functor $\mathbb{C} \to \mathbb{C}'$, and τ is a natural transformation $F \Rightarrow U\,;F'$. Moreover, the composition of morphisms is defined by $\langle U, \tau\rangle\,;\langle U', \tau'\rangle = \langle U\,;U', \tau\,;(U \cdot \tau')\rangle$.

Fact 2.7. Functors $G\colon \mathbb{K} \to \mathbb{K}'$ determine appropriate natural transformations $[G]$ between $[\_ \to \mathbb{K}]$ and $[\_ \to \mathbb{K}']$; their components $[G]_{\mathbb{C}}$, for $\mathbb{C} \in |\mathbb{C}\mathrm{at}^{\mathrm{op}}|$, are the obvious right-composition functors $[\mathbb{C} \to \mathbb{K}] \to [\mathbb{C} \to \mathbb{K}']$. Hence, by Fact 2.5, every functor G between categories $\mathbb{K}$ and $\mathbb{K}'$ induces a functor $[G]^\sharp\colon [\_ \to \mathbb{K}]^\sharp \to [\_ \to \mathbb{K}']^\sharp$.

Definition 2.8 (Subfunctor)**.** For any two functors $F, G\colon \mathbb{C} \to \mathbb{S}\mathrm{et}$, F is said to be a *subfunctor* of G (denoted $F \subseteq G$) if there exists a natural transformation $F \Rightarrow G$ whose components are all set-theoretic inclusions.

Therefore, F is a subfunctor of G when for every object $A \in |\mathbb{C}|$, $F(A)$ is a subset of $G(A)$, and for every arrow $f\colon A \to A'$ in $\mathbb{C}$, $F(f)$ is the domain-codomain restriction of $G(f)$.

Definition 2.9 (Generalized subfunctor)**.** For any two functors $F, G\colon \mathbb{I} \to [\_ \to \mathbb{S}\mathrm{et}]^\sharp$, F is said to be a *generalized subfunctor* of G (denoted $F \subseteq G$) when $F(i)$ is a natural subfunctor of $G(i)$ for every $i \in |\mathbb{I}|$.

This means that for every object $i \in |\mathbb{I}|$, $F(i)$ is a subfunctor of $G(i)$, and for every arrow $u\colon i \to i'$ in $\mathbb{I}$, $F(u)$ and $G(u)$ make the following square commute.

$$\begin{array}{ccc} i & F(i)\colon \mathbb{C}_i \to \mathbb{S}\mathrm{et} & \xrightarrow{F(i) \subseteq G(i)} G(i)\colon \mathbb{C}_i \to \mathbb{S}\mathrm{et} \\ {\scriptstyle u}\downarrow & {\scriptstyle F(u)}\downarrow & \downarrow{\scriptstyle G(u)} \\ i' & F(i')\colon \mathbb{C}_{i'} \to \mathbb{S}\mathrm{et} & \xrightarrow[F(i') \subseteq G(i')]{} G(i')\colon \mathbb{C}_{i'} \to \mathbb{S}\mathrm{et} \end{array}$$

Institutions

The notion of institution emerged within computing science from the general concept of language [5]. It was introduced by Goguen and Burstall in [23], with the aim of formalizing the intuitive notion of logical system as a balanced interaction between its syntax and its semantics. The theory of institutions provides in this way a rigorous model-theoretic device for dealing with the increasing number of logics used as foundation for specification and programming languages.

In what follows we recall the basic notions and notations of institutions. Intuitively, an institution consists of a collection of signatures, each of which determines appropriate sets of sentences and collections of models, as well as a satisfaction relation between models and sentences, which is assumed to be invariant under change of notation.

Definition 2.10 (Institution)**.** An *institution* $\mathcal{I} = \langle \mathbb{S}\mathrm{ig}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, \models^{\mathcal{I}} \rangle$ consists of

- a category $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ of *signatures* and *signature morphisms*,
- a *sentence functor* $\mathrm{Sen}^{\mathcal{I}}\colon \mathbb{S}\mathrm{ig}^{\mathcal{I}} \to \mathbb{S}\mathrm{et}$, defining for every signature Σ the set $\mathrm{Sen}^{\mathcal{I}}(\Sigma)$ of Σ-*sentences*, and for every signature morphism $\varphi\colon \Sigma \to \Sigma'$ the *sentence-translation map* $\mathrm{Sen}^{\mathcal{I}}(\varphi)\colon \mathrm{Sen}^{\mathcal{I}}(\Sigma) \to \mathrm{Sen}^{\mathcal{I}}(\Sigma')$,
- a *model functor* $\mathrm{Mod}^{\mathcal{I}}\colon (\mathbb{S}\mathrm{ig}^{\mathcal{I}})^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$, defining for every signature Σ the category $\mathrm{Mod}^{\mathcal{I}}(\Sigma)$ of Σ-*models* and Σ-*model homomorphisms*, and for each signature morphism $\varphi\colon \Sigma \to \Sigma'$ the *reduct functor* $\mathrm{Mod}^{\mathcal{I}}(\varphi)\colon \mathrm{Mod}^{\mathcal{I}}(\Sigma') \to \mathrm{Mod}^{\mathcal{I}}(\Sigma)$, and
- a family of *satisfaction relations* $\models^{\mathcal{I}}_{\Sigma} \subseteq |\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \times \mathrm{Sen}^{\mathcal{I}}(\Sigma)$, indexed by signatures,

such that the following *satisfaction condition* holds:

$$M' \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\varphi)(\rho) \quad \text{if and only if} \quad \mathrm{Mod}^{\mathcal{I}}(\varphi)(M') \models^{\mathcal{I}}_{\Sigma} \rho,$$

for every signature morphism $\varphi\colon \Sigma \to \Sigma'$, Σ'-model M', and Σ-sentence ρ.

When there is no danger of confusion we may omit the superscripts or subscripts from the notations introduced above. For instance, when the institution $\mathcal{I}$ and the signature Σ can be easily inferred, we will often denote the satisfaction relation $\models^{\mathcal{I}}_{\Sigma}$ simply by $\models$. In addition, we will frequently denote the sentence translation $\mathrm{Sen}^{\mathcal{I}}(\varphi)$ by $\varphi(\_)$, and the reduct functor $\mathrm{Mod}^{\mathcal{I}}(\varphi)$ by $\_{\restriction_\varphi}$; and we will say that M is the φ-*reduct* of M' and that M' is a

φ-*expansion* of M whenever $M = M'{\restriction_\varphi}$. Finally, we will use the fact that the satisfaction relation extends in a natural way from sentences to sets of sentences, and we will say that a Σ-sentence ρ is a *semantic consequence* of a set of Σ-sentences E, denoted $E \models^{\mathcal{I}}_{\Sigma} \rho$, when every Σ-model that satisfies every sentence in E satisfies ρ as well.

Note that for any signature Σ of an institution $\mathcal{I}$, the satisfaction relation $\models^{\mathcal{I}}_{\Sigma}$ can be identified with a Boolean-valued function $\models^{\mathcal{I}}_{\Sigma} : \mathrm{Sen}^{\mathcal{I}}(\Sigma) \to [|\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \to 2]$ that determines whether a given sentence is satisfied by a model. In view of this observation, the satisfaction condition of $\mathcal{I}$ can be captured categorically by defining the satisfaction relations $\models^{\mathcal{I}}_{\Sigma}$ as components of a natural transformation

$$\models^{\mathcal{I}} : \mathrm{Sen}^{\mathcal{I}} \Rightarrow [|\mathrm{Mod}^{\mathcal{I}}(\_)| \to 2]$$

from $\mathrm{Sen}^{\mathcal{I}}$ to the functor $[|\mathrm{Mod}^{\mathcal{I}}(\_)| \to 2]$ that maps every signature $\Sigma \in |\mathbb{S}\mathrm{ig}^{\mathcal{I}}|$ to the collection of all functions from $|\mathrm{Mod}^{\mathcal{I}}(\Sigma)|$ to the two-element set 2.

$$\begin{array}{ccc} \Sigma & \mathrm{Sen}^{\mathcal{I}}(\Sigma) & \xrightarrow{\models^{\mathcal{I}}_{\Sigma}} [|\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \to 2] \\ {\scriptstyle\varphi}\downarrow & {\scriptstyle \mathrm{Sen}^{\mathcal{I}}(\varphi)}\downarrow & \downarrow{\scriptstyle \mathrm{Mod}^{\mathcal{I}}(\varphi);\_} \\ \Sigma' & \mathrm{Sen}^{\mathcal{I}}(\Sigma') & \xrightarrow[\models^{\mathcal{I}}_{\Sigma'}]{} [|\mathrm{Mod}^{\mathcal{I}}(\Sigma')| \to 2] \end{array}$$

A large number of logical systems have been formalized as institutions. Some examples related to the study of logic programming include (many-sorted) first-order logic [23], order-sorted Horn-clause logic with equality [26], category-based equational logic [8], higher-order logic [36], hidden algebra [24], constraint logic [9], and the rather recent logic of asynchronous relational networks [49]. Many others are thoroughly discussed in monographs on institutions and algebraic specifications such as [11, 44].

Institution comorphisms [27] capture the intuitive notion of embedding simpler logical systems into more complex ones. They were originally discussed in [34] under the name of *plain maps*, and in [46] under the name of *institution representations*.

Definition 2.11 (Comorphism of institutions)**.** Given two institutions $\mathcal{I}$ and $\mathcal{I}'$, with $\mathcal{I} = \langle \mathbb{S}\mathrm{ig}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, \models^{\mathcal{I}} \rangle$ and $\mathcal{I}' = \langle \mathbb{S}\mathrm{ig}^{\mathcal{I}'}, \mathrm{Sen}^{\mathcal{I}'}, \mathrm{Mod}^{\mathcal{I}'}, \models^{\mathcal{I}'} \rangle$, a *comorphism* $\mathcal{I} \to \mathcal{I}'$ is a triple $\langle \Phi, \alpha, \beta \rangle$, where

- Φ is a *signature functor* $\mathbb{S}\mathrm{ig}^{\mathcal{I}} \to \mathbb{S}\mathrm{ig}^{\mathcal{I}'}$,
- α is a natural transformation $\mathrm{Sen}^{\mathcal{I}} \Rightarrow \Phi ; \mathrm{Sen}^{\mathcal{I}'}$, and
- β is a natural transformation $\Phi^{\mathrm{op}} ; \mathrm{Mod}^{\mathcal{I}'} \Rightarrow \mathrm{Mod}^{\mathcal{I}}$,

such that the following *satisfaction condition* holds:

$$M' \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha_{\Sigma}(\rho) \quad \text{if and only if} \quad \beta_{\Sigma}(M') \models^{\mathcal{I}}_{\Sigma} \rho,$$

for every $\mathcal{I}$-signature Σ, $\Phi(\Sigma)$-model M', and Σ-sentence ρ.

Institution comorphisms compose in a natural, componentwise manner. Their composition is associative and has identities, thus giving a category co$\mathbb{I}$ns of institutions and institution comorphisms.

Institutions as Functors

The definition of institutions as functors into a category of local satisfaction systems called *rooms* or *twisted relations* is a well-known characterization that was first proposed in [23], and proved to be convenient for various studies on generalized institutions [22], heterogeneous logical systems [37, 38], and logic translations [39].

Definition 2.12 (Room)**.** The category $\mathbb{R}$oom of *rooms* and *corridors* is defined as follows:

The *objects* are *Boolean rooms*, i.e. triples $\langle S, \mathbb{M}, \models \rangle$ consisting of

- a set S of *sentences*,

– a category $\mathbb{M}$ of *models*, and

– a *satisfaction relation* $\models \subseteq |\mathbb{M}| \times S$.

The *morphisms* $\langle S, \mathbb{M}, \models \rangle \to \langle S', \mathbb{M}', \models' \rangle$, called *corridors*, are pairs $\langle \alpha, \beta \rangle$, where

– α is a *sentence-translation function* $S \to S'$, and

– β is a *model reduction functor* $\mathbb{M}' \to \mathbb{M}$,

such that the following satisfaction condition holds for all models $M' \in |\mathbb{M}'|$ and all sentences $\rho \in S$:

$$M' \models' \alpha(\rho) \quad \text{if and only if} \quad \beta(M') \models \rho.$$

The *composition* of morphisms is defined componentwise.

The ternary structure of rooms induces appropriate projections Sen and Mod into the categories of sets and categories, respectively, as well as a natural transformation $\models$ that captures the invariance of the satisfaction of sentences by models with respect to the change of rooms, as follows:

– $\mathrm{Sen} \colon \mathbb{R}\mathrm{oom} \to \mathbb{S}\mathrm{et}$ and $\mathrm{Mod} \colon \mathbb{R}\mathrm{oom}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ are the functors that map every room $\langle S, \mathbb{M}, \models \rangle$ to its underlying set of sentences S and category of models $\mathbb{M}$,

– $\models$ is the natural transformation $\mathrm{Sen} \Rightarrow [|\mathrm{Mod}(\_)| \to 2]$ whose components are given by the satisfaction relations of the considered rooms.

Then every institution $\mathcal{I}$ having $\mathbb{S}\mathrm{ig}$ as the category of signatures can be identified with the functor $\mathcal{I} \colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ that maps

– every signature Σ to $\langle \mathrm{Sen}^{\mathcal{I}}(\Sigma), \mathrm{Mod}^{\mathcal{I}}(\Sigma), \models^{\mathcal{I}}_{\Sigma} \rangle$, and

– every signature morphism φ to $\langle \mathrm{Sen}^{\mathcal{I}}(\varphi), \mathrm{Mod}^{\mathcal{I}}(\varphi) \rangle$.

Conversely, every functor $\mathcal{I} \colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ describes an institution whose category of signatures, sentence functor, model functor, and family of satisfaction relations are given by $\mathbb{S}\mathrm{ig}$, $\mathcal{I}\,;\mathrm{Sen}$, $\mathcal{I}^{\mathrm{op}}\,;\mathrm{Mod}$, and $\mathcal{I} \cdot \models$, respectively.

This one-to-one correspondence between institutions and functors into $\mathbb{R}\mathrm{oom}$ can be easily extended to maps of institutions by noticing that every comorphism $\langle \Phi, \alpha, \beta \rangle$ from $\mathcal{I}$ to $\mathcal{I}'$ can also be regarded as an arrow $\langle \Phi, \tau \rangle$ between the functors $\mathcal{I} \colon \mathbb{S}\mathrm{ig}^{\mathcal{I}} \to \mathbb{R}\mathrm{oom}$ and $\mathcal{I}' \colon \mathbb{S}\mathrm{ig}^{\mathcal{I}'} \to \mathbb{R}\mathrm{oom}$, where $\alpha = \tau \cdot \mathrm{Sen}$, and $\beta = \tau^{\mathrm{op}} \cdot \mathrm{Mod}$.

Fact 2.13. The categories $\mathbb{C}\mathrm{oIns}$ and $[\_ \to \mathbb{R}\mathrm{oom}]^{\sharp}$ are isomorphic.

3 The Logical System of Reference

Our first step towards a presentation of the foundations of logic programming that does not depend on any concrete logical system consists in determining an appropriate benchmark institution that will enable us to relate to concepts and results specific to conventional logic programming [32]. To this end, we recall that logic programming has been traditionally studied in the Horn sub-institution of $\underline{\mathrm{FOL}}^{1}_{\neq}$—the single-sorted variant of first-order logic without equality.

Signatures. A $\underline{\mathrm{FOL}}^{1}_{\neq}$*-signature* is a pair $\langle F, P \rangle$, where F and P are families $(F_n)_{n \in \omega}$ and $(P_n)_{n \in \omega}$ of disjoint sets of operation and relation symbols, indexed by arities. These symbols are by definition qualified by their arity; in this sense, we denote an operation or relation symbol ς of arity n by $\varsigma \colon n$. However, when there is no risk of confusion we may drop the additional qualification and denote $\varsigma \colon n$ simply by ς. The *signature morphisms* $\varphi \colon \langle F, P \rangle \to \langle F', P' \rangle$ reflect the structure of signatures, and consist of families $\varphi^{\mathrm{op}} = (\varphi^{\mathrm{op}}_n \colon F_n \to F'_n)_{n \in \omega}$ and $\varphi^{\mathrm{rel}} = (\varphi^{\mathrm{rel}}_n \colon P_n \to P'_n)_{n \in \omega}$ of functions between the corresponding sets of operation and relation symbols.

Sentences. The sentences are usual first-order sentences built over relational atoms as follows. For any $\underline{\mathrm{FOL}}^{1}_{\neq}$-signature $\langle F, P \rangle$, the set T_F of *F-terms* is the least set such that $\sigma(t_1, \ldots, t_n) \in \mathrm{T}_F$ whenever $\sigma \in F_n$ and $t_1, \ldots, t_n \in \mathrm{T}_F$. Then the set of $\langle F, P \rangle$*-sentences* is the smallest set that contains the *relational atoms* $\pi(t_1, \ldots, t_n)$,

where $\pi \in P_n$ and $t_1, \ldots, t_n \in \mathrm{T}_F$, and is closed under Boolean connectives[1] and quantification over sets of first-order variables. It should be noted that $\langle F, P\rangle$-variables are pairs (x, F_0), often denoted simply by their name, x, and that the sets of $\langle F, P\rangle$-variables are in fact collections of variables such that different variables have different names. Also, every set of $\langle F, P\rangle$-variables X determines an extended signature $\langle F \cup X, P\rangle$, which can be obtained from $\langle F, P\rangle$ by adding the variables of X as new constants.

The *translation of sentences* along a signature morphism $\varphi\colon \langle F, P\rangle \to \langle F', P'\rangle$ is defined inductively on the structure of sentences by replacing the symbols of $\langle F, P\rangle$ according to φ. To be more precise, φ determines a function $\varphi^{\mathrm{tm}}\colon \mathrm{T}_F \to \mathrm{T}_{F'}$ given by $\sigma(t_1, \ldots, t_n) \mapsto \varphi_n^{\mathrm{op}}(\sigma)(\varphi^{\mathrm{tm}}(t_1), \ldots, \varphi^{\mathrm{tm}}(t_n))$, which allows us to define the translation of $\langle F, P\rangle$-relational atoms $\pi(t_1, \ldots, t_n)$ as $\varphi_n^{\mathrm{rel}}(\pi)(\varphi^{\mathrm{tm}}(t_1), \ldots, \varphi^{\mathrm{tm}}(t_n))$. This definition can be straightforwardly extended to more complex sentences. For instance, the translation of universally-quantified $\langle F, P\rangle$-sentences is given by $(\forall X)\, \rho \mapsto (\forall X^\varphi)\, \varphi^X(\rho)$, where X^φ is the set of $\langle F', P'\rangle$-variables $\{(x, F_0') \mid (x, F_0) \in X\}$, and $\varphi^X : \langle F \cup X, P\rangle \to \langle F' \cup X^\varphi, P'\rangle$ is the canonical extension of φ such that $(\varphi^X)_0^{\mathrm{op}}(x, F_0) = (x, F_0')$ for every $(x, F_0) \in X$.

$$\begin{array}{ccc} \langle F, P\rangle & \xrightarrow{\ \varphi\ } & \langle F', P'\rangle \\ {\scriptstyle\subseteq}\downarrow & & \downarrow{\scriptstyle\subseteq} \\ \langle F \cup X, P\rangle & \xrightarrow[\ \varphi^X\ ]{} & \langle F' \cup X^\varphi, P'\rangle \end{array}$$

Models. Considering a $\underline{\mathrm{FOL}}^1_{\neq}$-signature $\langle F, P\rangle$, an $\langle F, P\rangle$-*model* M consists of

- a set $|M|$, called the *carrier set* of M, together with
- a function $M_\sigma\colon |M|^n \to |M|$ for each operation symbol $\sigma \in F_n$, and
- a subset $M_\pi \subseteq |M|^n$ for each relation symbol $\pi \in P_n$.

Homomorphisms $h\colon M_1 \to M_2$ are F-algebra homomorphisms, i.e. functions $h\colon |M_1| \to |M_2|$ satisfying $h(M_{1,\sigma}(m_1, \ldots, m_n)) = M_{2,\sigma}(h(m_1), \ldots, h(m_n))$ for all $\sigma \in F_n$ and all arguments $m_1, \ldots, m_n \in |M_1|$, such that $h(M_{1,\pi}) \subseteq M_{2,\pi}$ for every $\pi \in P_n$.

With respect to *model reducts*, for a morphism of signatures $\varphi\colon \langle F, P\rangle \to \langle F', P'\rangle$ and an $\langle F', P'\rangle$-model M', the φ-reduct $M'{\restriction_\varphi}$ is defined as the $\langle F, P\rangle$-model M with the same carrier set as M', such that $M_\varsigma = M'_{\varphi(\varsigma)}$ for every operation or relation symbol ς of $\langle F, P\rangle$.

The satisfaction relation. The *satisfaction* between models and sentences is the usual Tarskian satisfaction defined inductively on the structure of sentences and based on the evaluation of terms in models. For example, given an $\langle F, P\rangle$-model M and a universally-quantified $\langle F, P\rangle$-sentence $(\forall X)\, \rho$, $M \models_{\langle F, P\rangle} (\forall X)\, \rho$ if and only if $N \models_{\langle F \cup X, P\rangle} \rho$ for all expansions N of M along the inclusion $\langle F, P\rangle \subseteq \langle F \cup X, P\rangle$.

The approach that we follow in the present inquiry on the foundations of logic programming relies on abstract concepts of universally- and existentially-quantified sentences, whose definitions are independent of the logical system of choice. For this reason, our description of conventional logic programming is not based on the institution $\underline{\mathrm{FOL}}^1_{\neq}$ discussed above, but on its quantifier-free fragment (with sentences built from atoms by repeated applications of Boolean connectives), which we denote by $\text{QF-}\underline{\mathrm{FOL}}^1_{\neq}$.

An Institution of Substitutions

The category of substitutions. Let us fix a $\text{QF-}\underline{\mathrm{FOL}}^1_{\neq}$-signature $\langle F, P\rangle$, for example the following signature of natural numbers with addition (specified as a ternary predicate).

$$F_{\mathtt{NAT}} = \{\mathtt{0}\colon 0,\ \mathtt{s\_}\colon 1\}$$
$$P_{\mathtt{NAT}} = \{\mathtt{add}\colon 3\}$$

[1] For convenience, we assume that disjunctions, denoted $\bigvee E$, and conjunctions, denoted $\bigwedge E$, are defined over arbitrary finite sets of sentences E, and we abbreviate $\bigwedge\{\rho_1, \rho_2\}$ as $\rho_1 \wedge \rho_2$, and $\bigwedge \emptyset$ as true. In addition, when there is no danger of confusion, we may also abbreviate $\bigwedge\{\rho\}$ as ρ.

In this setting, variables, or more precisely, sets of variables, can be seen as signatures of a specialized logical system. A *signature of variables* is just a set X of $\langle F, P\rangle$-variables with distinct names, which, by definition, shares no elements with the set F_0 of constant operation symbols. The syntactic entities over a given signature of variables X are defined as the corresponding QF-$\underline{\mathrm{FOL}}^1_{\neq}$-expressions that can be formed based on the operation and relation symbols of $\langle F, P\rangle$, and on the variables of X considered as new constants. For example, the terms over X, or with variables from X, can be informally defined in an inductive manner as follows:

- every variable of X is a term over X,
- if $\sigma \in F_n$ and $t_1, \ldots, t_n$ are terms over X then $\sigma(t_1, \ldots, t_n)$ is a term over X as well.

Substitutions provide syntactic transformations on expressions that are defined over sets of variables. Given two signatures of variables X and Y, a *substitution* $\psi\colon X \to Y$ is a map $\psi\colon X \to \mathrm{T}_{F\cup Y}$ that associates a term with variables from Y to every variable of X. Note that any substitution $\psi\colon X \to Y$ can be canonically extended to a function $\psi^{\mathrm{tm}}\colon \mathrm{T}_{F\cup X} \to \mathrm{T}_{F\cup Y}$ defined by $x \mapsto \psi(x)$ for variables, and $\sigma(t_1, \ldots, t_n) \mapsto \sigma(\psi^{\mathrm{tm}}(t_1), \ldots, \psi^{\mathrm{tm}}(t_n))$ for compound terms. This allows us to define the composition of two substitutions $\psi\colon X \to Y$ and $\theta\colon Y \to Z$ as the map $\psi\,;\theta^{\mathrm{tm}}\colon X \to \mathrm{T}_{F\cup Z}$. It is easy to see that $\psi\,;\theta$ induces the term translation $\psi^{\mathrm{tm}}\,;\theta^{\mathrm{tm}}$, which entails that the composition of substitutions is associative. Moreover, it satisfies the identity laws by taking, for every signature of variables X, the identity 1_X as the function that encodes the variables of X as terms over X. Hence, the signatures of $\langle F, P\rangle$-variables together with their corresponding substitutions form a category $\mathbb{S}\mathrm{ubst}_{\langle F,P\rangle}$.

Sentences, models, and the satisfaction relation. Signatures of variables do not inherit only the terms of the extended QF-$\underline{\mathrm{FOL}}^1_{\neq}$-signatures, but also their sentences, models, and the satisfaction relation between them. For every signature of variables X we define

- the set of sentences $\mathrm{Sen}_{\langle F,P\rangle}(X)$ as $\mathrm{Sen}(F \cup X, P)$,
- the category of models $\mathrm{Mod}_{\langle F,P\rangle}(X)$ as $\mathrm{Mod}(F \cup X, P)$, and
- the satisfaction relation $\models_{\langle F,P\rangle,X}$ as $\models_{\langle F\cup X,P\rangle}$.

With respect to substitutions, notice that for every two signatures of variables X and Y, every substitution $\psi\colon X \to Y$ determines a sentence-translation map

$$\mathrm{Sen}_{\langle F,P\rangle}(\psi)\colon \mathrm{Sen}_{\langle F,P\rangle}(X) \to \mathrm{Sen}_{\langle F,P\rangle}(Y)$$

defined on atoms by $\pi(t_1, \ldots, t_n) \mapsto \pi(\psi^{\mathrm{tm}}(t_1), \ldots, \psi^{\mathrm{tm}}(t_n))$, and a model reduct functor

$$\mathrm{Mod}_{\langle F,P\rangle}(\psi)\colon \mathrm{Mod}_{\langle F,P\rangle}(Y) \to \mathrm{Mod}_{\langle F,P\rangle}(X)$$

that maps every Y-model N to the X-model $N\restriction_\psi$ given by $|N\restriction_\psi| = |N|$, $(N\restriction_\psi)_\varsigma = N_\varsigma$ for each symbol ς of $\langle F, P\rangle$, and $(N\restriction_\psi)_x = N_{\psi(x)}$ for each variable $x \in X$.

We have thus defined the four components of an institution of $\langle F, P\rangle$-substitutions: the category of signatures $\mathbb{S}\mathrm{ubst}_{\langle F,P\rangle}$, the sentence functor $\mathrm{Sen}_{\langle F,P\rangle}$, the model functor $\mathrm{Mod}_{\langle F,P\rangle}$, and the family of satisfaction relations $(\models_{\langle F,P\rangle,X})_{X\in|\mathbb{S}\mathrm{ubst}_{\langle F,P\rangle}|}$. Our construction is completed by the following result, originally discussed in [10], stating that satisfaction is invariant with respect to substitution of variables.

Proposition 3.1. *For every substitution $\psi\colon X \to Y$, Y-model N, and X-sentence ρ,*

$$N \models_{\langle F,P\rangle,Y} \mathrm{Sen}_{\langle F,P\rangle}(\psi)(\rho) \quad \textit{if and only if} \quad \mathrm{Mod}_{\langle F,P\rangle}(\psi)(N) \models_{\langle F,P\rangle,X} \rho. \qquad \blacksquare$$

Corollary 3.2. *For every QF-$\underline{\mathrm{FOL}}^1_{\neq}$-signature $\langle F, P\rangle$, the structure*

$$(\text{QF-}\underline{\mathrm{FOL}}^1_{\neq})_{\langle F,P\rangle} = \langle \mathbb{S}\mathrm{ubst}_{\langle F,P\rangle}, \mathrm{Sen}_{\langle F,P\rangle}, \mathrm{Mod}_{\langle F,P\rangle}, \models_{\langle F,P\rangle}\rangle$$

describes an institution—the institution of $\langle F, P\rangle$-substitutions. $\blacksquare$

Institutions of substitutions are not considered in isolation; instead, they are linked by various institution comorphisms induced by morphisms between their underlying first-order signatures.

Proposition 3.3. *Every morphism of* QF-$\underline{\mathrm{FOL}}^1_{\neq}$*-signatures* $\varphi\colon \langle F, P\rangle \to \langle F', P'\rangle$ *determines a comorphism* $\langle \Psi_\varphi, \alpha_\varphi, \beta_\varphi\rangle$ *between the institutions of* $\langle F, P\rangle$*- and* $\langle F', P'\rangle$*-substitutions, where*

- $\Psi_\varphi(X) = X^\varphi$ *and* $\Psi_\varphi(\psi)(x, F'_0) = (\varphi^Y)^{\mathrm{tm}}(\psi(x, F_0))$,
- $\alpha_{\varphi,X} = \mathrm{Sen}(\varphi^X)$, *and*
- $\beta_{\varphi,X} = \mathrm{Mod}(\varphi^X)$,

for each signature of $\langle F, P\rangle$*-variables* X*, substitution* $\psi\colon X \to Y$*, and variable* $(x, F'_0) \in X^\varphi$.

Proof. We begin by proving that Ψ_φ is a functor $\mathbb{S}\mathrm{ubst}_{\langle F,P\rangle} \to \mathbb{S}\mathrm{ubst}_{\langle F',P'\rangle}$. Following an inductive argument on the structure of terms, we infer that for every $\langle F, P\rangle$-substitution $\psi\colon X \to Y$, the translations $(\varphi^X)^{\mathrm{tm}}\,;\Psi_\varphi(\psi)^{\mathrm{tm}}$ and $\psi^{\mathrm{tm}}\,;(\varphi^Y)^{\mathrm{tm}}$ are equal.

$$\begin{array}{ccc} \mathrm{T}_{F\cup X} & \xrightarrow{(\varphi^X)^{\mathrm{tm}}} & \mathrm{T}_{F'\cup X^\varphi} \\ {\scriptstyle \psi^{\mathrm{tm}}}\downarrow & & \downarrow{\scriptstyle \Psi_\varphi(\psi)^{\mathrm{tm}}} \\ \mathrm{T}_{F\cup Y} & \xrightarrow[(\varphi^Y)^{\mathrm{tm}}]{} & \mathrm{T}_{F'\cup Y^\varphi} \end{array}$$

Then, based on this property, for any two $\langle F, P\rangle$-substitutions $\psi\colon X \to Y$ and $\theta\colon Y \to Z$,

$$\begin{array}{ll} \Psi_\varphi(\psi\,;\theta)(x, F_0) & \\ \quad = (\varphi^Z)^{\mathrm{tm}}((\psi\,;\theta)(x, F_0)) & \text{by the definition of } \Psi_\varphi(\psi\,;\theta) \\ \quad = (\varphi^Z)^{\mathrm{tm}}(\theta^{\mathrm{tm}}(\psi(x, F_0))) & \text{by the definition of composition in } \mathbb{S}\mathrm{ubst}_{\langle F,P\rangle} \\ \quad = \Psi_\varphi(\theta)^{\mathrm{tm}}((\varphi^Y)^{\mathrm{tm}}(\psi(x, F_0))) & \text{because } (\varphi^Y)^{\mathrm{tm}}\,;\Psi_\varphi(\theta)^{\mathrm{tm}} = \theta^{\mathrm{tm}}\,;(\varphi^Z)^{\mathrm{tm}} \\ \quad = \Psi_\varphi(\theta)^{\mathrm{tm}}(\Psi_\varphi(\psi)((x, F_0))) & \text{by the definition of } \Psi_\varphi(\psi) \\ \quad = (\Psi_\varphi(\psi)\,;\Psi_\varphi(\theta))(x, F_0) & \text{by the definition of composition in } \mathbb{S}\mathrm{ubst}_{\langle F',P'\rangle}. \end{array}$$

In a similar manner, it can be shown that Ψ_φ also preserves identity substitutions.

As regards the last two components of the comorphism, the naturality of α_φ and β_φ amounts to the fact that the following two squares commute for every $\langle F, P\rangle$-substitution $\psi\colon X \to Y$.

$$\begin{array}{ccc} \mathrm{Sen}_{\langle F,P\rangle}(X) & \xrightarrow{\varphi^X(\_)} & \mathrm{Sen}_{\langle F',P'\rangle}(\Psi_\varphi(X)) \\ {\scriptstyle \psi(\_)}\downarrow & & \downarrow{\scriptstyle \Psi_\varphi(\psi)(\_)} \\ \mathrm{Sen}_{\langle F,P\rangle}(Y) & \xrightarrow[\varphi^Y(\_)]{} & \mathrm{Sen}_{\langle F',P'\rangle}(\Psi_\varphi(Y)) \end{array} \qquad \begin{array}{ccc} \mathrm{Mod}_{\langle F,P\rangle}(X) & \xleftarrow{\_\restriction_{\varphi^X}} & \mathrm{Mod}_{\langle F',P'\rangle}(\Psi_\varphi(X)) \\ {\scriptstyle \_\restriction_\psi}\uparrow & & \uparrow{\scriptstyle \_\restriction_{\Psi_\varphi(\psi)}} \\ \mathrm{Mod}_{\langle F,P\rangle}(Y) & \xleftarrow[\_\restriction_{\varphi^Y}]{} & \mathrm{Mod}_{\langle F',P'\rangle}(\Psi_\varphi(Y)) \end{array}$$

This can be established with ease through a series of straightforward calculations. For example, in the case of atomic sentences $\pi(t_1, \ldots, t_n)$ over X we obtain

$$\begin{array}{ll} \Psi_\varphi(\psi)(\varphi^X(\pi(t_1, \ldots, t_n))) & \\ \quad = \Psi_\varphi(\psi)(\varphi^{\mathrm{rel}}_n(\pi)((\varphi^X)^{\mathrm{tm}}(t_1), \ldots)) & \text{by the definition of } \varphi^X(\_) \\ \quad = \varphi^{\mathrm{rel}}_n(\pi)(\Psi_\varphi(\psi)^{\mathrm{tm}}((\varphi^X)^{\mathrm{tm}}(t_1)), \ldots) & \text{by the definition of } \Psi_\varphi(\psi)(\_) \\ \quad = \varphi^{\mathrm{rel}}_n(\pi)((\varphi^Y)^{\mathrm{tm}}(\psi^{\mathrm{tm}}(t_1)), \ldots) & \text{because } (\varphi^X)^{\mathrm{tm}}\,;\Psi_\varphi(\psi)^{\mathrm{tm}} = \psi^{\mathrm{tm}}\,;(\varphi^Y)^{\mathrm{tm}} \\ \quad = \varphi^Y(\pi(\psi^{\mathrm{tm}}(t_1), \ldots)) & \text{by the definition of } \varphi^Y(\_) \\ \quad = \varphi^Y(\psi(\pi(t_1, \ldots, t_n))) & \text{by the definition of } \psi(\_). \end{array}$$

■

4 Substitution Systems

The institutions of substitutions discussed in Section 3 provide the most basic building blocks needed for writing logic programs: signatures of variables and sentences over those signatures. For instance, the following definition of the addition of natural numbers (which makes use of the dedicated clausal notation specific to logic programming)

$$\begin{aligned} &\mathtt{add}(\mathtt{0}, M, M) \xleftarrow[M]{} \\ &\mathtt{add}(\mathtt{s}\, M, N, \mathtt{s}\, P) \xleftarrow[M,N,P]{} \mathtt{add}(M, N, P) \end{aligned}$$

can be presented formally as the set given by the universal closures of the sentences true $\Rightarrow \mathtt{add}(\mathtt{0}, M, M)$ and $\mathtt{add}(M, N, P) \Rightarrow \mathtt{add}(\mathtt{s}\, M, N, \mathtt{s}\, P)$ defined over the signatures of $\langle F_{\mathtt{NAT}}, P_{\mathtt{NAT}}\rangle$-variables $\{M\}$ and $\{M, N, P\}$, respectively.

In order to describe the semantics of such definitions at the same level of abstraction, we need to make explicit the translations of sentences and the reductions of models that correspond to signature extensions such as $\langle F_{\mathtt{NAT}}, P_{\mathtt{NAT}}\rangle \subseteq \langle F_{\mathtt{NAT}} \cup \{M\}, P_{\mathtt{NAT}}\rangle$. A possible approach is to consider signatures of Σ-variables, for some signature Σ of a given logical system, as specific morphisms $X : \Sigma \to \Sigma(X)$. This treatment of (signatures of) variables was first outlined in [42] as part of an institutional approach to open formulae, and is related to many developments in institution theory such as institution-independent semantics for quantifiers [45], ultraproducts [11], general versions of Herbrand theorems [10], Birkhoff completeness [6], hybridization [33], structural induction [12], constructor-based logics [18], and forcing techniques [19, 17].

The framework that we propose for formalizing signatures of variables (and substitutions too) generalizes the aforementioned approach by taking into account only their corresponding extensions of rooms. This choice is motivated by the difficulties that may arise in defining variables as signature extensions in other logical systems of interest for logic programming, such as the institution of asynchronous relational networks [49]. In that case, signatures (and consequently signature morphisms as well) and variables are significantly different from a structural point of view: the former are logical systems satisfying given properties, while the latter are hypergraphs whose vertices and hyperedges are labelled with signatures and models of their underlying logical system; for this reason, variables cannot be faithfully represented through extensions of their base signatures.

Definition 4.1 (Substitution system)**.** A *substitution system* is a triple $\langle \mathbb{S}\mathrm{ubst}, G, \mathcal{S}\rangle$, usually denoted simply by $\mathcal{S}$, consisting of

- a category $\mathbb{S}\mathrm{ubst}$ of *signatures of variables* and *substitutions*,
- a room G of *ground sentences* and *models*, and
- a functor $\mathcal{S} : \mathbb{S}\mathrm{ubst} \to G/\mathbb{R}\mathrm{oom}$, defining for every signature of variables X the corridor $\mathcal{S}(X) : G \to G(X)$ from G to the room $G(X)$ of *X-sentences* and *X-models*.

Therefore, given a room $G = \langle \mathrm{Sen}(G), \mathrm{Mod}(G), \models_G\rangle$ of ground sentences and models, the signatures of variables are defined abstractly as objects X of a category $\mathbb{S}\mathrm{ubst}$ for which we consider (a) a set $\mathrm{Sen}(G(X))$ of X-sentences together with a sentence-extension map $\alpha_X : \mathrm{Sen}(G) \to \mathrm{Sen}(G(X))$, ($b$) a category $\mathrm{Mod}(G(X))$ of X-models together with a model-reduction functor $\beta_X : \mathrm{Mod}(G(X)) \to \mathrm{Mod}(G)$, and ($c$) a satisfaction relation $\models_{G(X)}$ between X-models and X-sentences, such that the following satisfaction condition holds for all X-models N and ground sentences ρ:

$$N \models_{G(X)} \alpha_X(\rho) \quad \text{if and only if} \quad \beta_X(N) \models_G \rho.$$

Similarly, substitutions $\psi : X \to Y$ are arrows in $\mathbb{S}\mathrm{ubst}$ for which we assume to exist (a) a sentence-translation map $\mathrm{Sen}(\psi) : \mathrm{Sen}(G(X)) \to \mathrm{Sen}(G(Y))$, and ($b$) a model-reduction functor $\mathrm{Mod}(\psi) : \mathrm{Mod}(G(Y)) \to \mathrm{Mod}(G(X))$, such that the following diagrams commute

$$\begin{array}{ccc} & \mathrm{Sen}(G) & \\ {\scriptstyle \alpha_X}\swarrow & & \searrow{\scriptstyle \alpha_Y} \\ \mathrm{Sen}(G(X)) & \xrightarrow[\mathrm{Sen}(\psi)]{} & \mathrm{Sen}(G(Y)) \end{array} \qquad \begin{array}{ccc} & \mathrm{Mod}(G) & \\ {\scriptstyle \beta_X}\nearrow & & \nwarrow{\scriptstyle \beta_Y} \\ \mathrm{Mod}(G(X)) & \xleftarrow[\mathrm{Mod}(\psi)]{} & \mathrm{Mod}(G(Y)) \end{array}$$

and for every Y-model N and X-sentence ρ,

$$N \models_{G(Y)} \mathrm{Sen}(\psi)(\rho) \quad \text{if and only if} \quad \mathrm{Mod}(\psi)(N) \models_{G(X)} \rho.$$

It should be noted that whenever the signatures of variables are defined as extensions of their base signatures (in a given institution), the corridors $\langle \mathrm{Sen}(\psi), \mathrm{Mod}(\psi)\rangle$ determined by substitutions $\psi\colon X \to Y$ correspond to the institution-independent concept of substitution defined in [10].

Example 4.2. Every QF-$\underline{\mathrm{FOL}}^1_{\neq}$-signature $\langle F, P\rangle$ gives rise to a substitution system

$$(\text{QF-FOL}^1_{\neq})_{\langle F,P\rangle}\colon \mathbb{S}\mathrm{ubst}_{\langle F,P\rangle} \to \text{QF-}\underline{\mathrm{FOL}}^1_{\neq}(F,P) \,/\, \mathbb{R}\mathrm{oom}^2$$

in which

- signatures of $\langle F, P\rangle$-variables X determine corridors given by the set-theoretic inclusions $\mathrm{Sen}(F,P) \subseteq \mathrm{Sen}_{\langle F,P\rangle}(X)$ and the reduct functors $\mathrm{Mod}_{\langle F,P\rangle}(X) \to \mathrm{Mod}(F,P)$ that forget the interpretation of variables, and
- substitutions $\psi\colon X \to Y$ are mapped to $\langle \mathrm{Sen}_{\langle F,P\rangle}(\psi), \mathrm{Mod}_{\langle F,P\rangle}(\psi)\rangle$.

Note that we can easily recover the institution of $\langle F, P\rangle$-substitutions described in Corollary 3.2 (regarded as a functor into $\mathbb{R}$oom) by means of a straightforward composition:

$$(\text{QF-}\underline{\mathrm{FOL}}^1_{\neq})_{\langle F,P\rangle} = (\text{QF-FOL}^1_{\neq})_{\langle F,P\rangle} \,;\, |\_|_{\text{QF-}\underline{\mathrm{FOL}}^1_{\neq}(F,P)}.$$

Working with substitution systems instead of the simpler institutions of substitutions means that we also need to consider an adequate notion of map of substitution systems.

Definition 4.3 (Morphism of substitution systems)**.** A *morphism* between substitution systems $\mathcal{S}\colon \mathbb{S}\mathrm{ubst} \to G \,/\, \mathbb{R}\mathrm{oom}$ and $\mathcal{S}'\colon \mathbb{S}\mathrm{ubst}' \to G' \,/\, \mathbb{R}\mathrm{oom}$ is triple $\langle \Psi, \kappa, \tau\rangle$, where

$$\begin{array}{ccc} \mathbb{S}\mathrm{ubst} & \xrightarrow{\ \mathcal{S}\ } & G \,/\, \mathbb{R}\mathrm{oom} \\ {\scriptstyle\Psi}\downarrow & \Downarrow \tau & \uparrow{\scriptstyle \kappa/\mathbb{R}\mathrm{oom}} \\ \mathbb{S}\mathrm{ubst}' & \xrightarrow[\ \mathcal{S}'\ ]{} & G' \,/\, \mathbb{R}\mathrm{oom} \end{array}$$

- Ψ is a functor $\mathbb{S}\mathrm{ubst} \to \mathbb{S}\mathrm{ubst}'$,
- κ is a corridor $G \to G'$, and
- τ is a natural transformation $\mathcal{S} \Rightarrow \Psi\,;\mathcal{S}'\,;(\kappa \,/\, \mathbb{R}\mathrm{oom})$.

Proposition 4.4. *For every morphism of* QF-$\underline{\mathrm{FOL}}^1_{\neq}$*-signatures* $\varphi\colon \langle F, P\rangle \to \langle F', P'\rangle$*, the comorphism* $\langle \Psi_\varphi, \alpha_\varphi, \beta_\varphi\rangle$ *given in Proposition 3.3 can be extended to a morphism of substitution systems*

$$\langle \Psi_\varphi, \kappa_\varphi, \tau_\varphi\rangle\colon (\text{QF-FOL}^1_{\neq})_{\langle F,P\rangle} \to (\text{QF-FOL}^1_{\neq})_{\langle F',P'\rangle}$$

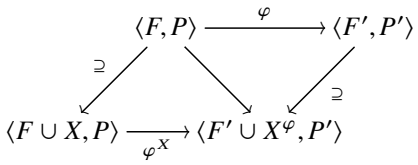
where κ_φ *is the corridor* QF-$\underline{\mathrm{FOL}}^1_{\neq}(\varphi) = \langle \mathrm{Sen}(\varphi), \mathrm{Mod}(\varphi)\rangle$*, and* $\tau_{\varphi,X} = \langle \alpha_{\varphi,X}, \beta_{\varphi,X}\rangle$ *for every* $X \in |\mathbb{S}\mathrm{ubst}_{\langle F,P\rangle}|$.

Proof. All we need to check is that, for every signature of $\langle F, P\rangle$-variables X, $\tau_{\varphi,X}$ is indeed an arrow in the comma category QF-$\underline{\mathrm{FOL}}^1_{\neq}(F,P) \,/\, \mathbb{R}\mathrm{oom}$. This follows directly from the fact that the equality

$$(\text{QF-FOL}^1_{\neq})_{\langle F,P\rangle}(X)\,;\tau_{\varphi,X} = \kappa_\varphi\,;(\text{QF-FOL}^1_{\neq})_{\langle F',P'\rangle}(\Psi_\varphi(X))$$

[2]The room QF-$\underline{\mathrm{FOL}}^1_{\neq}(F, P)$ is just the translation of the first-order signature $\langle F, P\rangle$ along the functor QF-$\underline{\mathrm{FOL}}^1_{\neq}\colon \mathbb{S}\mathrm{ig}^{\text{QF-}\underline{\mathrm{FOL}}^1_{\neq}} \to \mathbb{R}\mathrm{oom}$.

can be obtained by applying the functor QF-$\underline{\text{FOL}}^1_{\neq}$ to the commutative diagram below.



■

As expected, the composition of morphisms of substitution systems can be straightforwardly defined in terms of their components: for any morphisms $\langle \Psi, \kappa, \tau \rangle \colon \mathcal{S} \to \mathcal{S}'$ and $\langle \Psi', \kappa', \tau' \rangle \colon \mathcal{S}' \to \mathcal{S}''$ between substitution systems $\mathcal{S} \colon \mathbb{S}\text{ubst} \to G/\mathbb{R}\text{oom}$, $\mathcal{S}' \colon \mathbb{S}\text{ubst}' \to G'/\mathbb{R}\text{oom}$, and $\mathcal{S}'' \colon \mathbb{S}\text{ubst}'' \to G''/\mathbb{R}\text{oom}$, $\langle \Psi, \kappa, \tau \rangle ; \langle \Psi', \kappa', \tau' \rangle \colon \mathcal{S} \to \mathcal{S}''$ is given by

- the translation of signatures of variables $\Psi ; \Psi' \colon \mathbb{S}\text{ubst} \to \mathbb{S}\text{ubst}''$,
- the translation of ground sentences and models $\kappa ; \kappa' \colon G \to G''$, and
- the translation of sentences and models defined over signatures of variables

$$\tau ; (\Psi \cdot \tau' \cdot (\kappa \,/\, \mathbb{R}\text{oom})) \colon \mathcal{S} \Rightarrow (\Psi ; \Psi') ; \mathcal{S}'' ; (\kappa ; \kappa' \,/\, \mathbb{R}\text{oom}),$$

  where $(\tau ; (\Psi \cdot \tau' \cdot (\kappa \,/\, \mathbb{R}\text{oom})))_X = \tau_X ; \tau'_{\Psi(X)}$, for all signatures of variables X.

Together with obvious identities, the construction above yields a category $\mathbb{S}\text{ubst}\mathbb{S}\text{ys}$ with substitution systems as objects and morphisms of substitution systems as arrows.

Fact 4.5. The category $\mathbb{S}\text{ubst}\mathbb{S}\text{ys}$ arises from the Grothendieck construction for the functor $[\_ \to \_/\mathbb{R}\text{oom}] \colon (\mathbb{C}\text{at} \times \mathbb{R}\text{oom})^{\text{op}} \to \mathbb{C}\text{at}$ that maps

- every category $\mathbb{S}\text{ubst}$ and room G to the category of functors $[\mathbb{S}\text{ubst} \to G \,/\, \mathbb{R}\text{oom}]$, and
- every functor $\Psi \colon \mathbb{S}\text{ubst} \to \mathbb{S}\text{ubst}'$ and corridor $\kappa \colon G \to G'$ to the canonical composition functor $\Psi\_(\kappa \,/\, \mathbb{R}\text{oom}) \colon [\mathbb{S}\text{ubst}' \to G' \,/\, \mathbb{R}\text{oom}] \to [\mathbb{S}\text{ubst} \to G \,/\, \mathbb{R}\text{oom}]$.

We conclude the present section by noticing that the construction of first-order substitution systems is functorial, in the sense that it can be presented as a functor into $\mathbb{S}\text{ubst}\mathbb{S}\text{ys}$. We can thus define the following concept of generalized substitution system.

Definition 4.6 (Generalized substitution system)**.** *Generalized substitution systems* are objects of the category $[\_ \to \mathbb{S}\text{ubst}\mathbb{S}\text{ys}]^\sharp$ of functors into $\mathbb{S}\text{ubst}\mathbb{S}\text{ys}$.

Therefore, generalized substitution systems are functors $\mathcal{GS} \colon \mathbb{S}\text{ig} \to \mathbb{S}\text{ubst}\mathbb{S}\text{ys}$ from a category $\mathbb{S}\text{ig}$ of *signatures* and *signature morphisms* to $\mathbb{S}\text{ubst}\mathbb{S}\text{ys}$. Since this definition is rather compact, and lacks some of the transparency needed for dealing with logic-programming languages, let us first establish the notations and terminology that we will use throughout our work.

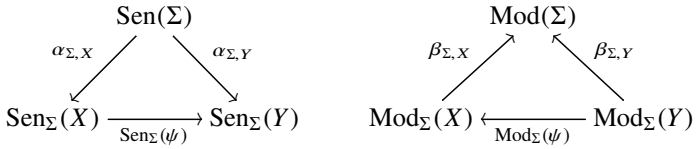
- Given a signature Σ of a generalized substitution system $\mathcal{GS}$, we will denote the (local) substitution system $\mathcal{GS}(\Sigma)$ by $\mathcal{GS}_\Sigma \colon \mathbb{S}\text{ubst}_\Sigma \to G_\Sigma \,/\, \mathbb{R}\text{oom}$, and will refer to the objects and morphisms of $\mathbb{S}\text{ubst}_\Sigma$ as *signatures of* Σ*-variables* and Σ*-substitutions*.

  The room G_Σ consists of the set $\text{Sen}(\Sigma)$ of *ground* Σ*-sentences*, the category $\text{Mod}(\Sigma)$ of Σ*-models*, and the Σ*-satisfaction relation* $\models_\Sigma \subseteq |\text{Mod}(\Sigma)| \times \text{Sen}(\Sigma)$.

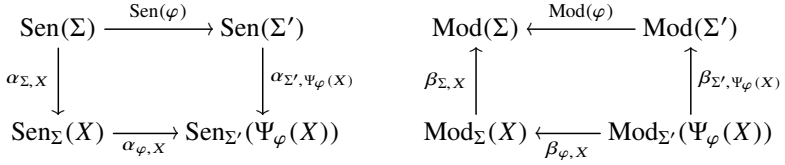
- On objects, $\mathcal{GS}_\Sigma$ maps every signature of Σ-variables X to the corridor $\mathcal{GS}_\Sigma(X) = \langle \alpha_{\Sigma,X}, \beta_{\Sigma,X} \rangle$ from G_Σ to the room $G_\Sigma(X) = \langle \text{Sen}_\Sigma(X), \text{Mod}_\Sigma(X), \models_{\Sigma,X} \rangle$ of X*-sentences* and X*-models*.

$$\alpha_{\Sigma,X} \colon \text{Sen}(\Sigma) \to \text{Sen}_\Sigma(X) \qquad \beta_{\Sigma,X} \colon \text{Mod}_\Sigma(X) \to \text{Mod}(\Sigma)$$

– On morphisms, $\mathcal{GS}_\Sigma$ maps every Σ-substitution $\psi\colon X \to Y$ to the corridor $\mathcal{GS}_\Sigma(\psi) = \langle \mathrm{Sen}_\Sigma(\psi), \mathrm{Mod}_\Sigma(\psi)\rangle$ from $G_\Sigma(X)$ to $G_\Sigma(Y)$, which satisfies, by definition, $\mathcal{GS}_\Sigma(X)\,;\mathcal{GS}_\Sigma(\psi) = \mathcal{GS}_\Sigma(Y)$.



– With respect to signature morphisms, every $\varphi\colon \Sigma \to \Sigma'$ determines a morphism of substitution systems $\mathcal{GS}_\varphi = \langle \Psi_\varphi, \kappa_\varphi, \tau_\varphi\rangle$ from $\mathcal{GS}_\Sigma$ to $\mathcal{GS}_{\Sigma'}$, where κ_φ is the corridor $\langle \mathrm{Sen}(\varphi), \mathrm{Mod}(\varphi)\rangle$ between G_Σ and $G_{\Sigma'}$, and for every signature of Σ-variables X, $\tau_{\varphi,X}$ is the corridor $\langle \alpha_{\varphi,X}, \beta_{\varphi,X}\rangle$ between $G_\Sigma(X)$ and $G_{\Sigma'}(\Psi_\varphi(X))$.



In addition, we adopt similar notational conventions as in the case of institutions. For example, we may use superscripts as in $\mathbb{S}\mathrm{ubst}_\Sigma^{\mathcal{GS}}$ in order to avoid potential ambiguities; or we may drop the subscripts of $\vDash_{\Sigma,X}$ when there is no danger of confusion. Likewise, we will often denote the functions $\mathrm{Sen}(\varphi)$, $\alpha_{\Sigma,X}$, and $\mathrm{Sen}_\Sigma(\psi)$ by $\varphi(\_)$, $X(\_)$, and $\psi(\_)$, respectively, and the functors $\mathrm{Mod}(\varphi)$, $\beta_{\Sigma,X}$, and $\mathrm{Mod}_\Sigma(\psi)$ by $\_\restriction_\varphi$, $\_\restriction_\Sigma$, and $\_\restriction_\psi$.

Example 4.7. The quantifier-free, single-sorted fragment of first-order logic without equality forms a generalized substitution system, which we denote by $\textsc{qf}$-$\mathrm{FOL}^1_{\neq}$.

$$\textsc{qf}\text{-}\mathrm{FOL}^1_{\neq}\colon \mathbb{S}\mathrm{ig}^{\underline{\textsc{qf}\text{-}\mathrm{FOL}^1_{\neq}}} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$$

Institutions of Quantified Sentences

The last ingredient needed to interpret logic programs is an appropriate concept of quantified sentence over a given generalized substitution system. To this purpose, we introduce universal and existential closures of sentences defined over signatures of variables by adapting the general institution-independent quantifiers of [45] to our framework of generalized substitution systems.

Definition 4.8 (Quantified sentence)**.** In any generalized substitution system $\mathcal{GS}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$, a *universally-quantified* Σ*-sentence* is a structure $(\forall X)\ \rho$, where X is a signature of Σ-variables and ρ is an X-sentence. The denotation of $(\forall X)\ \rho$ is given by the class of Σ-models whose X-expansions satisfy ρ. To be more precise, a Σ-model M is a model of $(\forall X)\ \rho$, denoted $M \vDash^{\mathrm{qs}}_\Sigma (\forall X)\ \rho$, if and only if for every X-model N such that $N\restriction_\Sigma = M$, $N \vDash_{\Sigma,X} \rho$.

Existentially-quantified Σ*-sentences* $(\exists X)\ \rho$ are introduced in a similar manner. Their semantics is, as expected, existential rather than universal: the denotation of a sentence $(\exists X)\ \rho$ is given by the class of Σ-models that admit X-expansions satisfying ρ.

Let $\mathrm{QSen}(\Sigma)$, or $\mathrm{QSen}^{\mathcal{GS}}(\Sigma)$ when we want to make explicit the underlying generalized substitution system, be the set of quantified sentences (universal or existential) over a signature Σ of a generalized substitution system $\mathcal{GS}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$. It is straightforward to see that QSen can be extended to a functor $\mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{et}$ by defining $\mathrm{QSen}(\varphi)((QX)\ \rho)$ as $(Q\Psi_\varphi(X))\ \alpha_{\varphi,X}(\rho)$ for every morphism of signatures $\varphi\colon \Sigma \to \Sigma'$ and every quantified Σ-sentence $(QX)\ \rho$, where $Q \in \{\forall, \exists\}$.

Fact 4.9. For any generalized substitution system $\mathcal{GS}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$, QSen is a *quantified-sentence functor* $\mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{et}$.

In order to reason about programs defined over different signatures, it is necessary to ensure that the translation of quantified sentences is consistent with the reduction of models—in the sense that the satisfaction relation is preserved. As in many institutions that capture logical systems with quantifiers, this condition relies on the essential property of model amalgamation, i.e. on the possibility of combining models of different but related signatures, provided that they have a common reduct to a given shared signature. The form of model amalgamation used here was first introduced in [3]; other early papers include [43, 14]. Its importance was also emphasized in works on heterogeneous specifications [47, 4], which are closely related to our setting.

Definition 4.10 (Model amalgamation)**.** A generalized substitution system $\mathcal{GS}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$ has *(weak) model amalgamation* when for every signature morphism $\varphi\colon \Sigma \to \Sigma'$ and every signature of Σ-variables X, the diagram depicted below is a (weak) pullback.

$$\begin{array}{ccc} |\mathrm{Mod}(\Sigma)| & \xleftarrow{\_\restriction_\varphi} & |\mathrm{Mod}(\Sigma')| \\ {\scriptstyle \_\restriction_\Sigma}\uparrow & & \uparrow{\scriptstyle \_\restriction_{\Sigma'}} \\ |\mathrm{Mod}_\Sigma(X)| & \xleftarrow[\beta_{\varphi,X}]{} & |\mathrm{Mod}_{\Sigma'}(\Psi_\varphi(X))| \end{array}$$

This means that for every Σ'-model M' and every X-model N such that $M'\restriction_\varphi = N\restriction_\Sigma$ there exists a $\Psi_\varphi(X)$-model N', called the *amalgamation* of M' and N, that satisfies $N'\restriction_{\Sigma'} = M'$ and $\beta_{\varphi,X}(N') = N$. Whenever this holds, the following commutative square, which subsumes the diagram above, is said to be a *(weak) model-amalgamation square*.

$$\begin{array}{ccc} G_\Sigma & \xrightarrow{\kappa_\varphi} & G_{\Sigma'} \\ {\scriptstyle \mathcal{GS}_\Sigma(X)}\downarrow & & \downarrow{\scriptstyle \mathcal{GS}_{\Sigma'}(\Psi_\varphi(X))} \\ G_\Sigma(X) & \xrightarrow[\tau_{\varphi,X}]{} & G_{\Sigma'}(\Psi_\varphi(X)) \end{array}$$

In generalized substitution systems such as QF-$\mathrm{FOL}^1_{\neq}$ (as well as QF-$\mathrm{FOL}_{=}$, discussed in Section 6), for every signature morphism $\varphi\colon \langle F, P\rangle \to \langle F', P'\rangle$ and every signature of $\langle F, P\rangle$-variables X, the commutative square of interest for model amalgamation can be obtained by taking the image through QF-$\underline{\mathrm{FOL}}^1_{\neq}$ of the following diagram of signature morphisms.

$$\begin{array}{ccc} \langle F, P\rangle & \xrightarrow{\varphi} & \langle F', P'\rangle \\ {\scriptstyle \subseteq}\downarrow & & \downarrow{\scriptstyle \subseteq} \\ \langle F \cup X, P\rangle & \xrightarrow[\varphi^X]{} & \langle F' \cup X^\varphi, P'\rangle \end{array}$$

It is well known that such diagrams describe pushouts, as well as the fact that every pushout of first-order signatures gives rise to a model-amalgamation square (details can be found, for example, in [11, 44]). These considerations lead to the following result.

Proposition 4.11. *The generalized substitution system* QF-$\mathrm{FOL}^1_{\neq}$ *has model amalgamation.* ∎

The model-amalgamation property allows us to prove that the translations of quantified sentences and the reductions of models preserve the satisfaction relation.

Proposition 4.12. *Let $\mathcal{GS}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$ be a generalized substitution system that has weak model amalgamation. Then for every signature morphism $\varphi\colon \Sigma \to \Sigma'$, every quantified Σ-sentence $(Q\,X)\ \rho$, and every Σ'-model M',*

$$M' \models^{\mathrm{qs}}_{\Sigma'} \mathrm{QSen}(\varphi)((Q\,X)\ \rho) \quad \textit{if and only if} \quad \mathrm{Mod}(\varphi)(M') \models^{\mathrm{qs}}_{\Sigma} (Q\,X)\ \rho.$$

Proof. Since the two kinds of quantified sentences can be treated similarly, we focus here only on the case of universal sentences. Let us thus consider a signature morphism $\varphi\colon \Sigma \to \Sigma'$, a universally-quantified Σ-sentence $(\forall X)\ \rho$, and a Σ'-model M'.

The proof of the 'if' part is simpler and it does not require the model-amalgamation property. Assume that $\mathrm{Mod}(\varphi)(M') \models^{\mathrm{qs}}_{\Sigma} (\forall X)\, \rho$, and let N' be an arbitrary $\Psi_\varphi(X)$-expansion of M'. Since $\kappa_\varphi ; \mathcal{GS}_{\Sigma'}(\Psi_\varphi(X)) = \mathcal{GS}_\Sigma(X) ; \tau_{\varphi,X}$ (because $\langle \Psi_\varphi, \kappa_\varphi, \tau_\varphi \rangle$ is a morphism of substitution systems), it follows that $\beta_{\varphi,X}(N')$ is an X-expansion of $\mathrm{Mod}(\varphi)(M')$. As a result, $\beta_{\varphi,X}(N') \models_{\Sigma,X} \rho$, which implies, by the satisfaction condition for $\tau_{\varphi,X}$, that $N' \models_{\Sigma', \Psi_\varphi(X)} \alpha_{\varphi,X}(\rho)$. Given that, by definition, $\mathrm{QSen}(\varphi)((\forall X)\, \rho)$ is the universally-quantified sentence $(\forall \Psi_\varphi(X))\, \alpha_{\varphi,X}(\rho)$, we conclude that $M' \models^{\mathrm{qs}}_{\Sigma'} \mathrm{QSen}(\varphi)((\forall X)\, \rho)$.

For the 'only if' part, assume that M' is a model of $\mathrm{QSen}(\varphi)((\forall X)\, \rho)$, and let N be an X-expansion of $\mathrm{Mod}(\varphi)(M')$, which means that $\mathrm{Mod}(\varphi)(M') = \beta_{\Sigma,X}(N)$. We need to show that N satisfies ρ. Since $\mathcal{GS}$ has weak model amalgamation, we deduce that there exists a $\Psi_\varphi(X)$-model N' such that $\beta_{\Sigma', \Psi_\varphi(X)}(N') = M'$ and $\beta_{\varphi,X}(N') = N$. This means that N' is a $\Psi_\varphi(X)$-expansion of M', and thus $N' \models_{\Sigma', \Psi_\varphi(X)} \alpha_{\varphi,X}(\rho)$ because, by hypothesis, $M' \models^{\mathrm{qs}}_{\Sigma'} (\forall \Psi_\varphi(X))\, \alpha_{\varphi,X}(\rho)$. Therefore, by the satisfaction condition for $\tau_{\varphi,X}$, we have $\beta_{\varphi,X}(N') \models_{\Sigma,X} \rho$. ∎

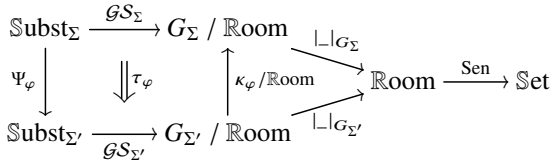
Corollary 4.13. *For any generalized substitution system* $\mathcal{GS} \colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$ *that has weak model amalgamation, the structure* $\mathcal{GS}^{\mathrm{qs}} = \langle \mathbb{S}\mathrm{ig}, \mathrm{QSen}, \mathrm{Mod}, \models^{\mathrm{qs}} \rangle$ *describes an institution*—the institution of quantified sentences over $\mathcal{GS}$. ∎

5 Abstract Logic Programming

As we have seen, generalized substitution systems provide us with a framework that is rich and flexible enough for capturing some of the most essential aspects of logic programming. However, since they make no assumptions on the structure of sentences defined over signatures of variables, we cannot distinguish between clauses and queries, nor can we describe unifiable sentences, which would further enable the search for solutions to queries by means of a suitable concept of resolution. To overcome these limitations, we extend the notion of generalized substitution system with appropriate functors that define local clauses and queries, as well as binary inference rules that allow the integration of the declarative and operational semantics of logic programming. The following property gives a very useful concise presentation of the local sentences defined by a generalized substitution system.

Fact 5.1. Every generalized substitution system $\mathcal{GS} \colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$ determines a *local-sentence functor* $\mathrm{LSen} \colon \mathbb{S}\mathrm{ig} \to [\_ \to \mathbb{S}\mathrm{et}]^\sharp$, also denoted $\mathrm{LSen}^{\mathcal{GS}}$ when we want to make explicit the generalized substitution system, that maps

- every signature Σ to the pair $\langle \mathbb{S}\mathrm{ubst}_\Sigma, \mathcal{GS}_\Sigma ; |\_|_{G_\Sigma} ; \mathrm{Sen} \rangle$, usually identified with the second component, the functor $\mathcal{GS}_\Sigma ; |\_|_{G_\Sigma} ; \mathrm{Sen} \colon \mathbb{S}\mathrm{ubst}_\Sigma \to \mathbb{S}\mathrm{et}$, and
- every signature morphism $\varphi \colon \Sigma \to \Sigma'$ to $\langle \Psi_\varphi, \tau_\varphi \cdot (|\_|_{G_\Sigma} ; \mathrm{Sen}) \rangle$.



This means that for every signature Σ, $\mathrm{LSen}(\Sigma)$ describes the sets $\mathrm{LSen}(\Sigma)(X) = \mathrm{Sen}_\Sigma(X)$ of *local* Σ*-sentences*, where X is a signature of Σ-variables, together with *translations of local sentences* $\mathrm{LSen}(\Sigma)(\psi) = \mathrm{Sen}_\Sigma(\psi)$, where ψ is a Σ-substitution. In addition, for every signature morphism $\varphi \colon \Sigma \to \Sigma'$, $\mathrm{LSen}(\varphi)$ describes the translation of Σ-variables $\Psi_\varphi \colon \mathbb{S}\mathrm{ubst}_\Sigma \to \mathbb{S}\mathrm{ubst}_{\Sigma'}$ and the family of natural maps $\alpha_{\varphi,X} \colon \mathrm{Sen}_\Sigma(X) \to \mathrm{Sen}_{\Sigma'}(\Psi_\varphi(X))$, indexed by signatures of Σ-variables X.

We can now define the main concept underlying our approach to logic programming.

Definition 5.2 (Logic-programming framework)**.** A *logic-programming framework* is a tuple $\mathcal{F} = \langle \mathcal{GS}, \mathrm{C}, \mathrm{Q}, \Vdash \rangle$, where

- $\mathcal{GS}$ is a generalized substitution system that has weak model amalgamation,

– C and Q are generalized subfunctors of LSen, defining the sets $\mathrm{C}_\Sigma(X) \subseteq \mathrm{Sen}_\Sigma(X)$ and $\mathrm{Q}_\Sigma(X) \subseteq \mathrm{Sen}_\Sigma(X)$ of *X-clauses* and *X-queries*, respectively, for every signature Σ and every signature of Σ-variables X, and

– $\Vdash$ is a generalized subfunctor of $(\mathrm{Q} \times \mathrm{C}) \times \mathrm{Q}$, defining the set $\Vdash_{\Sigma,X} \subseteq (\mathrm{Q}_\Sigma(X) \times \mathrm{C}_\Sigma(X)) \times \mathrm{Q}_\Sigma(X)$ of *X-goal-directed rules*—where $(\rho_1, \gamma, \rho_2) \in \Vdash_{\Sigma,X}$ is usually written $\rho_1, \gamma \Vdash_{\Sigma,X} \rho_2$—for every signature Σ and every signature of Σ-variables X,

such that the following *soundness property* holds:

$$\rho_1, \gamma \Vdash_{\Sigma,X} \rho_2 \quad \text{implies} \quad \{\rho_2, \gamma\} \vDash_{\Sigma,X} \rho_1,$$

for every signature Σ, signature of Σ-variables X, X-queries ρ_1 and ρ_2, and every X-clause γ.

Example 5.3. Conventional first-order logic programming is defined over the generalized substitution system $_{\mathrm{QF}}\text{-FOL}^1_{\neq}$ as follows:

– the set $\mathrm{C}_{\langle F,P\rangle}(X)$ of X-clauses consists of all implications of the form $\bigwedge H \Rightarrow C$, where H is a finite set of relational atoms and C is an atom,

– the set $\mathrm{Q}_{\langle F,P\rangle}(X)$ of X-queries consists of all finite conjunctions of atoms $\bigwedge Q$, and

– the goal-directed rules are given by $\bigwedge(\{C\} \cup Q), (\bigwedge H \Rightarrow C) \Vdash_{\langle F,P\rangle,X} \bigwedge(Q \cup H)$,

for every signature $\langle F, P\rangle$ and every signature of $\langle F, P\rangle$-variables X.

One can easily see that the (local) first-order clauses, queries, and goal-directed rules are preserved along both signature morphisms and substitutions; this means, for example, that the translation of a clause along a signature morphism, when regarded as a sentence over the domain signature of that morphism, is a clause defined over the codomain signature of the morphism. Moreover, the goal-directed rules correspond, in essence, to applications of *modus ponens*, and thus they are sound. As a result, the constructions above define a logic-programming framework, which we denote by $\mathcal{FOL}^1_{\neq}$.

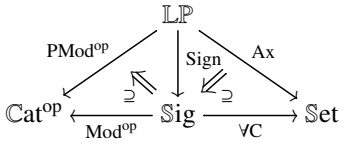
The definitions of clauses and queries are rather straightforward.

Definition 5.4 (Clause and query)**.** Let Σ be a signature of the underlying generalized substitution system $\mathcal{GS}$ of a logic-programming framework $\langle \mathcal{GS}, \mathrm{C}, \mathrm{Q}, \Vdash\rangle$. A Σ-*clause* is a universally-quantified sentence $(\forall X)\, \gamma$ over Σ such that $\gamma \in \mathrm{C}_\Sigma(X)$. Similarly, a Σ-*query* is an existentially-quantified sentence $(\exists X)\, \rho$ over Σ such that $\rho \in \mathrm{Q}_\Sigma(X)$.

We obtain in this way two subfunctors $\forall\mathrm{C}$ (defining the clauses) and $\exists\mathrm{Q}$ (defining the queries) of the sentence functor QSen of the institution of quantified sentences over $\mathcal{GS}$.

Following recent developments in the theory of structured specifications [13], we propose an axiomatic approach to logic-programming languages, in which the programs are treated as abstract entities, each of them defining a signature (in a given logic-programming framework), a class of models—that provides the denotational semantics of the program—and an appropriate set of clauses—that supports the operational semantics of the program. In addition, programs are related by morphisms that induce appropriate reductions of models and translations of clauses, defined in terms of their projections to the underlying logic-programming framework.

Definition 5.5 (Logic-programming language)**.** Given a framework $\mathcal{F} = \langle \mathcal{GS}, \mathrm{C}, \mathrm{Q}, \Vdash\rangle$, a *logic-programming language* $\mathcal{L} = \langle \mathbb{LP}, \mathrm{Sign}, \mathrm{PMod}, \mathrm{Ax}\rangle$ over $\mathcal{F}$ consists of



– a category $\mathbb{LP}$ of *logic programs* and *morphisms of logic programs*,

– a *signature functor* $\mathrm{Sign}\colon \mathbb{LP} \to \mathbb{S}\mathrm{ig}$, and

– subfunctors $\mathrm{PMod} \subseteq \mathrm{Sign}^{\mathrm{op}}\,;\mathrm{Mod}$ and $\mathrm{Ax} \subseteq \mathrm{Sign}\,;\forall\mathrm{C}$, defining for every program P the category $\mathrm{PMod}(P)$ of *P-models* and the set $\mathrm{Ax}(P)$ of *P-clauses*,

such that for every logic program P and every model M of P, $M \models^{\mathrm{qs}}_{\mathrm{Sign}(P)} \mathrm{Ax}(P)$.

By way of notation, when there is no risk of confusion, we will often denote logic programs P by $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$, where $\Sigma = \mathrm{Sign}(P)$ and $\Gamma = \mathrm{Ax}(P)$, emphasizing in this way their corresponding signatures and sets of clauses; moreover, we may denote simply by ν the signature morphism $\mathrm{Sign}(\nu)$ determined by a morphism of programs $\nu\colon P \to P'$. We may also indicate that a Σ-model M is a model of P by writing $M \models^{\mathrm{lp}}_{\Sigma} P$ in place of $M \in |\mathrm{PMod}(P)|$, and that a quantified Σ-sentence $(QX)\,\rho$ is semantically entailed by P, meaning that $M \models^{\mathrm{qs}}_{\Sigma} (QX)\,\rho$ for all P-models M, by writing $P \models^{\mathrm{lp}}_{\Sigma} (QX)\,\rho$.

Example 5.6. Most of the mechanisms used in defining (structured) specifications can also be used to define logic programs. For instance, one of the simplest and most common descriptions of logic programs is as *presentations* [23, 14] over a logic-programming framework $\mathcal{F} = \langle \mathcal{GS}, \mathrm{C}, \mathrm{Q}, \Vdash \rangle$ such as $\mathcal{FOL}^{1}_{\neq}$. In this case, the programs defined by the resulting logic-programming language—denoted $\mathcal{F}^{\mathrm{pres}}$—are pairs $\langle \Sigma, \Gamma \rangle$ of signatures Σ of $\mathcal{GS}$ and sets of Σ-clauses Γ, while the morphisms $\varphi\colon \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ are merely signature morphisms $\varphi\colon \Sigma \to \Sigma'$ such that $\Gamma' \models^{\mathrm{qs}}_{\Sigma'} \varphi(\Gamma)$. The functors Sign, Ax, and PMod have their usual interpretations: for every presentation $\langle \Sigma, \Gamma \rangle$, (*a*) $\mathrm{Sign}(\Sigma, \Gamma)$ is the signature Σ, (*b*) $\mathrm{Ax}(\Sigma, \Gamma)$ is the set of Σ-clauses Γ, and (*c*) $\mathrm{PMod}(\Sigma, \Gamma)$ is the full subcategory of $\mathrm{Mod}(\Sigma)$ given by the models of Σ that satisfy Γ.

Under this formalism, based on the first-order signature $\langle F_{\mathtt{NAT}}, P_{\mathtt{NAT}} \rangle$ considered in Section 3 and on the clauses presented in Section 4, the logic program over $(\mathcal{FOL}^{1}_{\neq})^{\mathrm{pres}}$ that defines the addition of natural numbers can be described as follows:

signature

op $\mathtt{0}\colon 0$

op $\mathtt{s\_}\colon 1$

pred $\mathtt{add}\colon 3$

axioms

$\mathtt{add}(\mathtt{0}, M, M) \xleftarrow[M]{}$

$\mathtt{add}(\mathtt{s}\,M, N, \mathtt{s}\,P) \xleftarrow[M,N,P]{} \mathtt{add}(M, N, P)$

For the remaining part of this section we will assume $\mathcal{L} = \langle \mathbb{LP}, \mathrm{Sign}, \mathrm{PMod}, \mathrm{Ax} \rangle$ to be an arbitrary but fixed logic-programming language over a framework $\mathcal{F} = \langle \mathcal{GS}, \mathrm{C}, \mathrm{Q}, \Vdash \rangle$.

Herbrand's Theorem

From a model-theoretic perspective, a logic program $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ gives a positive answer to a Σ-query $(\exists X)\,\rho$ if and only if $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ semantically entails $(\exists X)\,\rho$, which means, of course, that every model of $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ admits an X-expansion that satisfies ρ. In practice, this kind of answer is not always adequate for the particular task at hand, as we may be interested in finding the actual 'values' for X—independent of any choice of model of $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$—that meet all the requirements described by ρ.

Definition 5.7 (Solution)**.** For any logic program $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$, a Σ-substitution $\psi\colon X \to Y$ is a $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$-*solution*, or *correct* $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$-*answer*, to a Σ-query $(\exists X)\,\rho$ if Y is conservative, i.e. if the functor $\beta_{\Sigma,Y}\colon \mathrm{Mod}_{\Sigma}(Y) \to \mathrm{Mod}(\Sigma)$ is surjective on objects,[3] and $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall Y)\,\psi(\rho)$.

Therefore, we have to consider two notions of answer to a query based on a logic program: the first one corresponds to the denotational semantics of the logic-programming language, while the second provides the necessary foundations for its operational semantics.

Herbrand's theorem ensures that the two notions are equivalent. First, it reduces the semantic entailment of the considered query to satisfaction in the initial model of the logic program; then, it shows that any expansion

[3] Note that, in first-order logic, a (non-empty) signature of $\langle F, P \rangle$-variables is conservative if and only if F_0 is not empty.

of the initial model that satisfies the underlying local sentence of the query gives rise to a solution, and vice versa. These properties are well known in the literature for various logical systems (e.g. [32, 26, 24]), and they were first investigated in an institution-independent setting in [10]. The result we develop here upgrades the ones from [10] by considering a different set of hypotheses, based on the existence of certain reachable models instead of representable signature morphisms and substitutions; moreover, it can be utilized even in situations in which the signatures of variables cannot be described as extensions of their base signatures. These new assumptions are significantly more permissive as they allow us to apply the present theory to logical systems like those discussed in [49, 50], that do not fit into the framework proposed in [10]—which, in fact, can be shown to be a concrete realization of the conceptual structure that we put forward here.

The following concept of reachable model extends (non-trivially) the homonymous one from [19] to the present setting of generalized substitution systems by eliminating the need for an initial signature of variables with the same sentences, models, and satisfaction relation as its corresponding base signature.

Definition 5.8 (Reachable model)**.** Let Σ be a signature and X a signature of Σ-variables in a generalized substitution system. A Σ-model M is said to be *X-reachable* when for every X-expansion N of M there exists a Σ-substitution $\psi\colon X \to Y$ such that

- Y is conservative, and
- the canonical map $\_\restriction_\Sigma\colon N \,/\, \mathrm{Mod}_\Sigma(\psi) \to M \,/\, \beta_{\Sigma,Y}$ determined by the reduct functor $\beta_{\Sigma,X}$ is surjective on objects.

Hence, given N and ψ as above, a Σ-model M is X-reachable if every homomorphism $h\colon M \to N_1\restriction_\Sigma$ from M to the Σ-reduct of an Y-model N_1 admits an X-expansion of the form $f\colon N \to N_1\restriction_\psi$.

In many concrete examples of institutions (see, e.g. [19]), a model M is reachable (with respect to some signature of variables) if and only if all its elements are interpretations of terms, i.e. if the unique homomorphism $0_\Sigma \to M$ from the initial model of the signature of M to M is epi. In particular, for first-order logic, the initial model $0_{\langle F,P\rangle,\Gamma}$ of a set Γ of $\langle F,P\rangle$-clauses is X-reachable for every signature of $\langle F,P\rangle$-variables X: every expansion $N_{\langle F,P\rangle,\Gamma}$ of $0_{\langle F,P\rangle,\Gamma}$ yields a substitution $\psi\colon X \to \emptyset$ to the empty signature of variables (which is conservative, and has the same models as $\langle F,P\rangle$) such that $0_{\langle F,P\rangle,\Gamma}\restriction_\psi = N_{\langle F,P\rangle,\Gamma}$; as a result, every homomorphism $h\colon 0_{\langle F,P\rangle,\Gamma} \to N_1\restriction_{\langle F,P\rangle}$ in $\mathrm{Mod}(F,P)$ admits an X-expansion $N_{\langle F,P\rangle,\Gamma} \to N_1\restriction_\psi$ given by the reduct $h\restriction_\psi$.

Proposition 5.9. *Every logic program $\langle\langle F,P\rangle,\Gamma\rangle$ of $(\mathcal{FOL}^1_{\neq})^{\mathrm{pres}}$ admits an initial model $0_{\langle F,P\rangle,\Gamma}$ that is reachable with respect to all signatures of $\langle F,P\rangle$-variables.* ∎

In addition to the existence of particular reachable models, we also require (as in [10]) that model homomorphisms preserve the satisfaction of the local sentence upon which the query under consideration is based.

Definition 5.10 (Preservation of satisfaction)**.** Given a signature Σ and a signature of Σ-variables X in a generalized substitution system $\mathcal{GS}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$, an X-homomorphism $h\colon N_1 \to N_2$ is said to *preserve the satisfaction* of an X-sentence ρ if $N_1 \models_{\Sigma,X} \rho$ implies $N_2 \models_{\Sigma,X} \rho$.

Note that, by definition, in the case of QF-$\mathrm{FOL}^1_{\neq}$, all homomorphisms preserve the satisfaction of all relational atoms. This property can be easily extended to arbitrary conjunctions of atoms, and thus to local queries of $\mathcal{FOL}^1_{\neq}$.

Fact 5.11. In $\mathcal{FOL}^1_{\neq}$, all homomorphisms preserve the satisfaction of all local queries.

Theorem 5.12 (Herbrand's theorem)**.** *For every logic program $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ and every Σ-query $(\exists X)\,\rho$ such that (a) $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ has an X-reachable initial model $0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle}$, and (b) the satisfaction of ρ is preserved by X-homomorphisms, the following statements are equivalent:*

1. $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle \models^{\mathrm{lp}}_\Sigma (\exists X)\,\rho$,
2. $0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \models^{\mathrm{qs}}_\Sigma (\exists X)\,\rho$,
3. $(\exists X)\,\rho$ *admits a* $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$*-solution.*

Proof.

1 ⇒ 2. Obvious, since $0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \models^{\mathrm{lp}}_{\Sigma} \langle\!\langle\Sigma,\Gamma\rangle\!\rangle$.

2 ⇒ 3. By hypothesis, there exists an X-expansion $N_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle}$ of $0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle}$ so that $N_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \models_{\Sigma,X} \rho$. Since $0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle}$ is assumed to be X-reachable, we know there exists a substitution $\psi\colon X \to Y$ (determined by $N_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle}$) satisfying the two properties listed in Definition 5.8. Hence, we only need to show that $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall Y)\, \psi(\rho)$.

Let us thus consider a model M of $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ and a Y-expansion N of M. Based on the initiality of $0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle}$, we obtain a (unique) Σ-homomorphism $h\colon 0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \to M = N\restriction_{\Sigma}$, which can be lifted, by the surjectivity of the map $\_\restriction_{\Sigma}\colon N_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \,/\, \mathrm{Mod}_{\Sigma}(\psi) \to 0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \,/\, \beta_{\Sigma,Y}$, to an X-homomorphism $f\colon N_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \to N\restriction_{\psi}$. Since f preserves the satisfaction of ρ (by hypothesis) and $N_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle} \models_{\Sigma,X} \rho$, it follows that $N\restriction_{\psi} \models_{\Sigma,X} \rho$, which implies, by the satisfaction condition for ψ, that $N \models_{\Sigma,Y} \psi(\rho)$. Therefore, $M \models^{\mathrm{qs}}_{\Sigma} (\forall Y)\, \psi(\rho)$.

3 ⇒ 1. Assume that $\psi\colon X \to Y$ is a $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$-solution to $(\exists X)\, \rho$, and that M is a model of $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$. This means that $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall Y)\, \psi(\rho)$, and hence $M \models^{\mathrm{qs}}_{\Sigma} (\forall Y)\, \psi(\rho)$. In addition, Y is conservative, from which we deduce that there exists a Y-expansion N of M such that $N \models_{\Sigma,Y} \psi(\rho)$. It follows by the satisfaction condition for ψ that $N\restriction_{\psi}$ is an X-expansion of M such that $N\restriction_{\psi} \models_{\Sigma,X} \rho$. Consequently, $M \models^{\mathrm{qs}}_{\Sigma} (\exists X)\, \rho$. ∎

Remark 5.13. The additional hypotheses referring to the existence of a reachable initial model of $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ and to the preservation of the satisfaction of ρ by homomorphisms are used only in the proof of the 'completeness part' of Theorem 5.12, i.e. for the implication 1 ⇒ 3;[4] the 'soundness part', corresponding to the implication 3 ⇒ 1, holds for every logic program $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ and any Σ-query $(\exists X)\, \rho$.

Operational Semantics

One of the most important—and distinctive—features of the concept of logic-programming language is that it allows us to make effective use of the resolution inference rule, and in a very natural way, to give an operational semantics to conventional logic programming. Consider, for example, the logic program $\langle\langle F_{\mathtt{NAT}}, P_{\mathtt{NAT}}\rangle, \Gamma\rangle$ described in Example 5.6 and the query $(\exists\{X_1\})\, \mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_1)$, sometimes written as

$$\underset{X_1}{\vdash} \mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_1)$$

using a notation similar to that of clauses. We can compute a solution to the query $(\exists\{X_1\})\, \mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_1)$ using the clauses of Γ and the goal-directed rules of $\mathcal{FOL}^1_{\neq}$ as follows: we first derive the query $(\exists\{X_2\})\, \mathtt{add}(\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_2)$, based on the second clause of Γ, the substitutions $\theta_1\colon \{X_1\} \to \{X_2\}$ and $\psi_1\colon \{M,N,P\} \to \{X_2\}$ given by $X_1 \mapsto \mathtt{s}\, X_2$, $M \mapsto \mathtt{0}$, $N \mapsto \mathtt{s}\,\mathtt{0}$ and $P \mapsto X_2$, and the following goal-directed rule over $\{X_2\}$;

$$\underbrace{\mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, \mathtt{s}\, X_2)}_{\theta_1(\mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_1))}, \underbrace{\mathtt{add}(\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_2) \Rightarrow \mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, \mathtt{s}\, X_2)}_{\psi_1(\mathtt{add}(M,N,P) \Rightarrow \mathtt{add}(\mathtt{s}\, M, N, \mathtt{s}\, P))} \Vdash_{\langle F_{\mathtt{NAT}}, P_{\mathtt{NAT}}\rangle, X_2} \mathtt{add}(\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_2)$$

by iterating these constructions, we can further derive the trivial query $(\exists\emptyset)\, \mathtt{true}$ using the first clause, the substitutions $\theta_2\colon \{X_2\} \to \emptyset$ and $\psi_2\colon \{M\} \to \emptyset$ given by $X_2 \mapsto \mathtt{s}\,\mathtt{0}$ and $M \mapsto \mathtt{s}\,\mathtt{0}$, and the goal-directed rule over $\emptyset$ detailed below;

$$\underbrace{\mathtt{add}(\mathtt{0}, \mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0})}_{\theta_2(\mathtt{add}(\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_2))}, \underbrace{\mathtt{true} \Rightarrow \mathtt{add}(\mathtt{0}, \mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0})}_{\psi_2(\mathtt{true} \Rightarrow \mathtt{add}(\mathtt{0}, M, M))} \Vdash_{\langle F_{\mathtt{NAT}}, P_{\mathtt{NAT}}\rangle, \emptyset} \mathtt{true}$$

finally, we compose the substitutions θ_1 and θ_2 (which are usually computed through term unification) to obtain a solution to $(\exists\{X_1\})\, \mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_1)$.

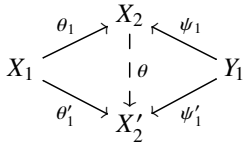
This procedure does not depend on any particular details of $\mathcal{FOL}^1_{\neq}$, and can be conveniently defined within the abstract framework of logic-programming languages.

Definition 5.14 (Resolution)**.** Let $(\exists X_1)\, \rho_1$ be a query and $(\forall Y_1)\, \gamma_1$ a clause over a signature Σ. A Σ-query $(\exists X_2)\, \rho_2$ is said to be *derived by resolution* from $(\exists X_1)\, \rho_1$ and $(\forall Y_1)\, \gamma_1$ using the *computed substitution* $\theta_1\colon X_1 \to X_2$ if there exists a substitution $\psi_1\colon Y_1 \to X_2$ such that $\theta_1(\rho_1), \psi_1(\gamma_1) \Vdash_{\Sigma,X_2} \rho_2$.

[4] Moreover, it would suffice to assume that the model $0_{\langle\!\langle\Sigma,\Gamma\rangle\!\rangle}$ is weakly initial.

Unification. Note that resolution describes not only the derivation of new and (presumably) simpler queries from appropriate pairs of queries and clauses, but also how partial answers to the original queries can be computed by means of sentence unification. In this sense, for any signature Σ and signatures of Σ-variables X_1 and Y_1, two sentences $\rho_1 \in \mathrm{Q}_\Sigma(X_1)$ and $\gamma_1 \in \mathrm{C}_\Sigma(Y_1)$ are said to be *unifiable* if there exists a pair $\langle \theta_1, \psi_1 \rangle$ of substitutions $\theta_1 \colon X_1 \to X_2$ and $\psi_1 \colon Y_1 \to X_2$, called the *unifier* of ρ_1 and γ_1, such that $\theta_1(\rho_1), \psi_1(\gamma_1) \Vdash_{\Sigma, X_2} \rho_2$ for some X_2-sentence $\rho_2 \in \mathrm{Q}_\Sigma(X_2)$.

It is also possible to distinguish between the various levels of generality of the unifiers. Given two unifiers $\langle \theta_1, \psi_1 \rangle$ and $\langle \theta_1', \psi_1' \rangle$ of ρ_1 and γ_1 as depicted below, we say that $\langle \theta_1', \psi_1' \rangle$ is an *instance* of $\langle \theta_1, \psi_1 \rangle$, or that $\langle \theta_1, \psi_1 \rangle$ is *more general* than $\langle \theta_1', \psi_1' \rangle$, if there exists a substitution θ such that $\theta_1 ; \theta = \theta_1'$ and $\psi_1 ; \theta = \psi_1'$. Then the unifiers of ρ_1 and γ_1 can be defined as objects of a subcategory of the category of cospans of Σ-substitutions.



It should be noted however that, under the present formalization, the most general unifiers, defined as initial objects in their corresponding category (along the lines of [20]), cannot be guaranteed to exist. Even in the case of $\mathcal{FOL}^1_{\neq}$, one cannot find, for example, a most general unifier of $\mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_1) \wedge \mathtt{add}(\mathtt{s}\,\mathtt{s}\,\mathtt{0}, \mathtt{0}, X_1)$ and $\mathtt{add}(M, N, P) \Rightarrow \mathtt{add}(\mathtt{s}\,M, N, \mathtt{s}\,P)$—although one exists for $\mathtt{add}(\mathtt{s}\,\mathtt{0}, \mathtt{s}\,\mathtt{0}, X_1)$ and $\mathtt{add}(M, N, P) \Rightarrow \mathtt{add}(\mathtt{s}\,M, N, \mathtt{s}\,P)$, as well as for $\mathtt{add}(\mathtt{s}\,\mathtt{s}\,\mathtt{0}, \mathtt{0}, X_1)$ and $\mathtt{add}(M, N, P) \Rightarrow \mathtt{add}(\mathtt{s}\,M, N, \mathtt{s}\,P)$. This does not restrict the applicability of the theory proposed here, since our abstract notion of resolution corresponds in fact to an extended variant of first-order resolution, in which any unifier may give rise to a derivation, not necessarily the most general ones.

The scope of Definition 5.14 can be easily broaden to accommodate sets of clauses: given a set Γ of clauses over a signature Σ, a Σ-query $(\exists X_2)\, \rho_2$ is said to be *derived by resolution* from $(\exists X_1)\, \rho_1$ and Γ using the *computed substitution* $\theta_1 \colon X_1 \to X_2$, written

$$(\exists X_1)\, \rho_1 \longrightarrow\!\!\!\!\!\rightarrow_{\Gamma, \theta_1} (\exists X_2)\, \rho_2,$$

if $(\exists X_2)\, \rho_2$ is derived by resolution from $(\exists X_1)\, \rho_1$ and $(\forall Y_1)\, \gamma_1$ using the computed substitution θ_1, for some Σ-clause $(\forall Y_1)\, \gamma_1 \in \Gamma$. This gives us a family $(\longrightarrow\!\!\!\!\!\rightarrow_{\Gamma, \theta})_{\theta \in \mathbb{S}\mathrm{ubst}_\Sigma}$ of *one-step derivation relations generated by* Γ, whose union is denoted by $\longrightarrow\!\!\!\!\!\rightarrow_\Gamma$.

The rest of this subsection is devoted to the composition of one-step derivations, which will be shown to provide a general procedure for resolving queries. To this purpose, let us first investigate the soundness of the one-step derivation relations.

Proposition 5.15. *Let $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ be a logic program, and $(\exists X_1)\, \rho_1$ and $(\exists X_2)\, \rho_2$ two Σ-queries. Then for every inference step $(\exists X_1)\, \rho_1 \longrightarrow\!\!\!\!\!\rightarrow_{\Gamma, \theta_1} (\exists X_2)\, \rho_2$ and every solution $\psi \colon X_2 \to Y$ to the query $(\exists X_2)\, \rho_2$, the substitution $\theta_1 ; \psi$ is a solution to $(\exists X_1)\, \rho_1$.*

Proof. Assume that $(\exists X_2)\, \rho_2$ is derived by resolution from $(\exists X_1)\, \rho_1$ and Γ using the computed substitution $\theta_1 \colon X_1 \to X_2$, and that $\psi \colon X_2 \to Y$ is a Γ-solution to $(\exists X_2)\, \rho_2$. The latter implies that the codomain Y of $\theta_1 ; \psi$ is conservative. Therefore, we only need to prove that $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \vDash^{\mathrm{lp}}_\Sigma (\forall Y)\, (\theta_1 ; \psi)(\rho_1)$.

Let M be a model of $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ and N a Y-expansion of M. By the definition of the one-step derivation relation $\longrightarrow\!\!\!\!\!\rightarrow_{\Gamma, \theta_1}$ we know there exists a clause $(\forall Y_1)\, \gamma_1 \in \Gamma$ and a substitution $\psi_1 \colon Y_1 \to X_2$ such that $\theta_1(\rho_1), \psi_1(\gamma_1) \Vdash_{\Sigma, X_2} \rho_2$. This allows us to deduce, based on the soundness of the goal-directed rules, that $\{\rho_2, \psi_1(\gamma_1)\} \vDash_{\Sigma, X_2} \theta_1(\rho_1)$. Furthermore, since the semantic consequence is preserved by translation along signature morphisms (which, in turn, is an immediate consequence of the invariance of truth under change of notation), we obtain $\{\psi(\rho_2), (\psi_1 ; \psi)(\gamma_1)\} \vDash_{\Sigma, Y} (\theta_1 ; \psi)(\rho_1)$. This means that in order to conclude our proof it suffices to show that N satisfies both $\psi(\rho_2)$ and $(\psi_1 ; \psi)(\gamma_1)$.

In the case of the first relation, since ψ is a $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$-solution to $(\exists X_2)\, \rho_2$, $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \vDash^{\mathrm{lp}}_\Sigma (\forall Y)\, \psi(\rho_2)$. It follows that $M \vDash^{\mathrm{qs}}_\Sigma (\forall Y)\, \psi(\rho_2)$, which further implies $N \vDash_{\Sigma, Y} \psi(\rho_2)$.

In the case of the second relation, by the general properties of substitution systems, we know that $N\!\upharpoonright_{\psi_1 ; \psi}$ is a Y_1-expansion of M. As a result, $N\!\upharpoonright_{\psi_1 ; \psi} \vDash_{\Sigma, Y_1} \gamma_1$ because $M \vDash^{\mathrm{qs}}_\Sigma (\forall Y_1)\, \gamma_1$, and thus, by the satisfaction condition for $\psi_1 ; \psi$, $N \vDash_{\Sigma, X_2} (\psi_1 ; \psi)(\gamma_1)$. ∎

The search for solutions proceeds by means of a sequence of one-step derivations.

Definition 5.16 (Derivation)**.** For any set Γ of clauses over Σ, and any Σ-queries $(\exists X_1)\,\rho_1$ and $(\exists X_n)\,\rho_n$, a Γ-*derivation* of $(\exists X_n)\,\rho_n$ from $(\exists X_1)\,\rho_1$ is a chain of one-step derivations

$$(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow_{\Gamma,\theta_1} (\exists X_2)\,\rho_2 \longrightarrow\!\!\!\!\rightarrow_{\Gamma,\theta_2} (\exists X_3)\,\rho_3 \;\cdots\; (\exists X_{n-1})\,\rho_{n-1} \longrightarrow\!\!\!\!\rightarrow_{\Gamma,\theta_{n-1}} (\exists X_n)\,\rho_n.$$

Whenever such a chain exists, the query $(\exists X_n)\,\rho_n$ is said to be *derived* from $(\exists X_1)\,\rho_1$ and Γ using the *computed substitution* $\theta = \theta_1\,;\theta_2\,;\cdots;\theta_{n-1}$, written $(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta} (\exists X_n)\,\rho_n$. By definition, if the chain depicted above is empty, $(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,1_{X_1}} (\exists X_1)\,\rho_1$. We obtain in this way a family $(\longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta})_{\theta\in\mathbb{S}\mathrm{ubst}_\Sigma}$ of *derivation relations generated by* Γ.

Remark 5.17. The union of $(\longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta})_{\theta\in\mathbb{S}\mathrm{ubst}_\Sigma}$ is the reflexive and transitive closure of $\longrightarrow\!\!\!\!\rightarrow_\Gamma$.

Proposition 5.15 can be generalized without difficulty from one-step to arbitrary derivations by a straightforward induction on the length of the derivation.

Corollary 5.18. *Let $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ be a logic program, and $(\exists X_1)\,\rho_1$ and $(\exists X_n)\,\rho_n$ two Σ-queries. Then for every derivation $(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta} (\exists X_n)\,\rho_n$ and every solution $\psi\colon X_n \to Y$ to $(\exists X_n)\,\rho_n$, the substitution $\theta\,;\psi$ is a solution to $(\exists X_1)\,\rho_1$.* ∎

All we require now in order to define computed answers is a concept of trivial query, which is meant to characterize the successful termination of the search procedure.

Definition 5.19 (Trivial query)**.** Let Σ be a signature. A Σ-query $(\exists Y)\,\top$ is said to be *trivial* if and only if Y is conservative and every Y-model satisfies $\top$.

Fact 5.20. In $\mathcal{FOL}^1_{\neq}$, a query is trivial if and only if it is of the form $(\exists Y)$ true.[5]

An immediate consequence of the definition above is that every trivial query corresponds to a local tautology (and even more, the query itself is a tautology). As a result, for any logic program $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ and any trivial Σ-query $(\exists Y)\,\top$, $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle \vDash^{\mathrm{lp}}_\Sigma (\forall Y)\,\top$. This amounts to describing the identity 1_Y as a $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$-solution to $(\exists Y)\,\top$.

Fact 5.21. Every trivial query $(\exists Y)\,\top$ over the signature Σ of a logic program $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ admits a $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$-solution—the identity substitution 1_Y.

Definition 5.22 (Computed answer)**.** Given a program $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$, a Σ-substitution $\theta\colon X \to Y$ is a *computed* $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$-*answer* to a Σ-query $(\exists X)\,\rho$ if there exists a trivial query $(\exists Y)\,\top$ such that $(\exists X)\,\rho \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta} (\exists Y)\,\top$.

Corollary 5.18 and Fact 5.21 lead us to the first main result of this subsection.

Theorem 5.23 (Soundness of resolution)**.** *For any logic program $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$ and any Σ-query $(\exists X)\,\rho$, every computed $\langle\!\langle\Sigma,\Gamma\rangle\!\rangle$-answer to $(\exists X)\,\rho$ is a solution to $(\exists X)\,\rho$.* ∎

Completeness

As expected, completeness is more difficult to obtain, and requires additional hypotheses. To simplify the proof, we consider two lemmas, each of which introduces a new property to be satisfied by the query at hand or by the considered logic program. The first lemma allows us to derive by resolution any translation (along a substitution) of a given query, while the second reduces the search for solutions to a sequence of elementary inferences in a local institution of substitutions—which, in the case of first-order logic-programming, involves no quantifiers.

Definition 5.24 (Identity clause)**.** Let $(\exists X)\,\rho$ be a query over a signature Σ. A Σ-clause $(\forall X)\,\gamma$ is said to be an *identity* of $(\exists X)\,\rho$ if and only if γ is a tautology and $\rho,\gamma \Vdash_{\Sigma,X} \rho$.

Fact 5.25. In $\mathcal{FOL}^1_{\neq}$, every non-trivial query $(\exists X)\ \bigwedge Q$ admits an identity $(\forall X)\,\pi(t_1,\ldots,t_n) \Rightarrow \pi(t_1,\ldots,t_n)$, where $\pi(t_1,\ldots,t_n)$ is an atom in Q.

[5] Note that, to ensure the conservativity of non-empty signatures of variables, the considered first-order signature has to define at least one constant.

Lemma 5.26. *Consider a signature Σ, a Σ-query $(\exists X)\,\rho$, and an identity $(\forall X)\,\gamma$ of $(\exists X)\,\rho$. Then for every Σ-substitution $\psi\colon X \to Y$ there exists a one-step derivation of $(\exists Y)\,\psi(\rho)$ from $(\exists X)\,\rho$ and $(\forall X)\,\gamma$ having ψ as the computed substitution.*

$$(\exists X)\,\rho \longrightarrow\!\!\!\!\rightarrow_{\{(\forall X)\,\gamma\},\psi} (\exists Y)\,\psi(\rho)$$

Proof. Since $(\forall X)\,\gamma$ is the identity of $(\exists X)\,\rho$, we know that $\rho, \gamma \Vdash_{\Sigma,X} \rho$. This implies $\psi(\rho), \psi(\gamma) \Vdash_{\Sigma,Y} \psi(\rho)$ because, by the functoriality of $\Vdash$, the goal-directed rules are preserved along substitutions. It follows that $\langle \psi, \psi \rangle$ is a unifier of ρ and γ, which allows us to conclude that $(\exists Y)\,\psi(\rho)$ can be derived by resolution from $(\exists X)\,\rho$ and $(\forall X)\,\gamma$ using the computed substitution ψ. ∎

The second lemma is based on the following concept of instance of a set of clauses, which, together with the generalization of the goal-directed rules to sets of local clauses, provides an essential characterization of the derivation relation.

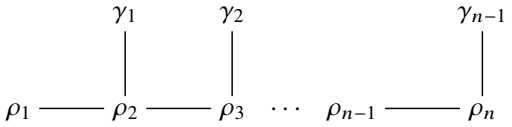
Definition 5.27 (Instance of a set of clauses)**.** Given a set Γ of clauses over Σ and a signature of Σ-variables X, the *X-instance of* Γ is defined as the set $X(\Gamma)$ of all X-sentences that can be obtained by translating the local clauses of Γ along substitutions into X.

$$X(\Gamma) = \{\psi(\gamma) \in \mathrm{Sen}_\Sigma(X) \mid (\forall Y)\,\gamma \in \Gamma \text{ and } \psi\colon Y \to X\}$$

Definition 5.28. For every signature Σ and signature of Σ-variables X, $\Vdash_{\Sigma,X}$ can be extended to a set $\Vdash^*_{\Sigma,X}$ of goal-directed rules between X-queries and sets of X-clauses as follows: given two queries $\rho_1, \rho_n \in \mathrm{Q}_\Sigma(X)$ and a (possibly empty) set of clauses $G \subseteq \mathrm{C}_\Sigma(X)$,

$$\rho_1, G \Vdash^*_{\Sigma,X} \rho_n$$

if and only if there exists an alternating sequence $\rho_1\,\gamma_1\,\rho_2\,\gamma_2\,\rho_3\,\ldots\,\rho_{n-1}\,\gamma_{n-1}\,\rho_n$ of queries $\rho_i \in \mathrm{Q}_\Sigma(X)$ and clauses $\gamma_i \in G$ such that $\rho_i, \gamma_i \Vdash_{\Sigma,X} \rho_{i+1}$ for each $1 \leq i < n$.



This gives rise to a generalized subfunctor $\Vdash^*$ of $(\mathrm{Q} \times \mathrm{C}\,;[\mathcal{P}]^\sharp) \times \mathrm{Q}$.[6]

Proposition 5.29. *For all sets Γ of clauses over a signature Σ, queries $(\exists X_1)\,\rho_1$ and $(\exists X_n)\,\rho_n$ over Σ, and all (computed) substitutions $\theta\colon X_1 \to X_n$,*

$$(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta} (\exists X_n)\,\rho_n \quad \textit{if and only if} \quad \theta(\rho_1), X_n(\Gamma) \Vdash^*_{\Sigma,X_n} \rho_n.$$

Proof. We prove the result by induction on the length of the derivation. Since for the base case the conclusion follows directly from the definitions of $\longrightarrow\!\!\!\!\rightarrow^*_\Gamma$ and $\Vdash^*$, we will focus entirely on the more interesting case corresponding to the induction step.

For the 'if' part, assume there exists an X_n-query ρ_2 and an X_n-clause $\psi_1(\gamma_1) \in X_n(\Gamma)$, further indicating the existence of a clause $(\forall Y_1)\,\gamma_1 \in \Gamma$ and of a substitution $\psi_1\colon Y_1 \to X_n$, such that $\theta(\rho_1), \psi_1(\gamma_1) \Vdash_{\Sigma,X_n} \rho_2$ and $\rho_2, X_n(\Gamma) \Vdash^*_{\Sigma,X_n} \rho_n$. It follows by the definition of resolution and by the induction hypothesis that $(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow_{\Gamma,\theta} (\exists X_n)\,\rho_2$ and $(\exists X_n)\,\rho_2 \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,1_{X_n}} (\exists X_n)\,\rho_n$; therefore, by the composition of these derivations, $(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta} (\exists X_n)\,\rho_n$.

For the 'only if' part, let $\theta_1\,;\theta_2$ be a factorization of θ such that $(\exists X_1)\,\rho_1 \longrightarrow\!\!\!\!\rightarrow_{\Gamma,\theta_1} (\exists X_2)\,\rho_2$ and $(\exists X_2)\,\rho_2 \longrightarrow\!\!\!\!\rightarrow^*_{\Gamma,\theta_2} (\exists X_n)\,\rho_n$. Then, by the induction hypothesis, $\theta_2(\rho_2), X_n(\Gamma) \Vdash^*_{\Sigma,X_n} \rho_n$. We also know, by the definition of resolution, that there exists a clause $(\forall Y_1)\,\gamma_1 \in \Gamma$ and a substitution $\psi_1\colon Y_1 \to X_2$ such that $\theta_1(\rho_1), \psi_1(\gamma_1) \Vdash_{\Sigma,X_2} \rho_2$. Based on the functoriality of $\Vdash^*$, this implies that $(\theta_1\,;\theta_2)(\rho_1), (\psi_1\,;\theta_2)(\gamma_1) \Vdash_{\Sigma,X_n} \theta_2(\rho_2)$, and thus, since $(\psi_1\,;\theta_2)(\gamma_1)$ belongs to $X_n(\Gamma)$ and $\theta_2(\rho_2), X_n(\Gamma) \Vdash^*_{\Sigma,X_n} \rho_n$, we conclude that $\theta(\rho_1), X_n(\Gamma) \Vdash^*_{\Sigma,X_n} \rho_n$. ∎

[6] We recall from Fact 2.7 and Definition 5.2 that $\mathrm{C}\,;[\mathcal{P}]^\sharp$ is the functor that maps every signature Σ and every signature of Σ-variables X to the set $\mathcal{P}(\mathrm{C}_\Sigma(X))$ of sets of X-clauses.

Let us recall that, in general, the derivation of queries proceeds by selecting, at each step, a new rule—over a new signature of variables—to be applied to the current goal. Provided that we know the result of the derivation, Proposition 5.29 allows us to reduce the derivation of queries to applications of goal-directed rules that are defined over the same signature of variables. In view of this characterization, the soundness of resolution can be interpreted locally as described in the following corollary.

Corollary 5.30. *Let $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ be a logic program, X a conservative signature of Σ-variables, and ρ an X-query. Then for every trivial X-query $\top$,*

$$\rho, X(\Gamma) \Vdash^{*}_{\Sigma,X} \top \quad \textit{implies} \quad \langle\!\langle \Sigma, \Gamma \rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall X)\, \rho.$$

Proof. Assume $\top$ to be a trivial query over X such that $\rho, X(\Gamma) \Vdash^{*}_{\Sigma,X} \top$. By Proposition 5.29, it follows that $(\exists X)\, \rho \longrightarrow\!\!\!\!\rightarrow^{*}_{\Gamma, 1_X} (\exists X)\, \top$. This means that the identity 1_X is a computed $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$-answer to $(\exists X)\, \rho$ (because X is conservative), and thus, by Theorem 5.23, 1_X is also a solution to $(\exists X)\, \rho$. As a result, $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall X)\, \rho$. ∎

With respect to the completeness of resolution, we are interested in the converse of the implication discussed in the corollary above. This may hold for certain programs in logic-programming languages of interest, but it cannot be guaranteed in general.

Definition 5.31 (Query-completeness)**.** A logic program $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ is said to be *query-complete* if for every conservative signature of Σ-variables X, and every X-query ρ,

$$\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall X)\, \rho \quad \text{implies} \quad \rho, X(\Gamma) \Vdash^{*}_{\Sigma,X} \top$$

for some trivial X-query $\top$. In addition, the logic-programming language $\mathcal{L}$ is *query-complete* when all logic programs of $\mathcal{L}$ have this property.

The following result is well known in the literature; it can be found, for instance, in a slightly different form, in [32]. The result is based on the observation that for every signature of variables X over a first-order signature $\langle F, P \rangle$, and for every set Γ of clauses over $\langle F, P \rangle$, the X-expansion N of the free $\langle \langle F, P \rangle, \Gamma \rangle$-model over X given by $N_x = x$ for every $x \in X$ satisfies a relational atom ρ if and only if there exist a clause $(\forall Y)\ \bigwedge H \Rightarrow C$ in Γ and a substitution $\psi : Y \to X$ such that $\psi(C) = \rho$ and $N \vDash_{\langle F, P \rangle, X} \bigwedge \psi(H)$.

Therefore, if $\langle \langle F, P \rangle, \Gamma \rangle \models^{\mathrm{lp}}_{\langle F, P \rangle} (\forall X)\, \rho$, then $N \vDash_{\langle F, P \rangle, X} \rho$, and thus there exists a clause $\psi(\gamma) \in X(\Gamma)$ such that $\rho, \psi(\gamma) \Vdash_{\Sigma, X} \rho_1$, where $N \vDash_{\langle F, P \rangle, X} \rho_1$. By iterating this construction, we obtain an alternating sequence of X-queries and clauses as in Definition 5.28; furthermore, since $N\restriction_{\langle F, P \rangle}$ is the free $\langle \langle F, P \rangle, \Gamma \rangle$-model over X, it is always possible to assemble such a sequence that terminates in a trivial query, i.e. in true. Consequently, $\rho, X(\Gamma) \Vdash^{*}_{\Sigma,X}$ true. A more detailed presentation of this result can be found in [32].

Proposition 5.32. *The logic-programming language $(\mathcal{FOL}^{1}_{\neq})^{\mathrm{pres}}$ is query-complete.* ∎

We can now obtain our second lemma as a direct consequence of Proposition 5.29.

Lemma 5.33. *Suppose that $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ is a query-complete logic program. Then for every Σ-query $(\exists X)\, \rho$ such that X is conservative and $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall X)\, \rho$, there exists a trivial query $(\exists X)\, \top$ that can be derived from $(\exists X)\, \rho$ and Γ, with the identity substitution 1_X as the computed answer.*

$$(\exists X)\, \rho \longrightarrow\!\!\!\!\rightarrow^{*}_{\Gamma, 1_X} (\exists X)\, \top$$

Proof. Let $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ be a program and $(\exists X)\, \rho$ a Σ-query such that $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \models^{\mathrm{lp}}_{\Sigma} (\forall X)\, \rho$. By query-completeness, there exists a trivial X-query $\top$ such that $\rho, X(\Gamma) \Vdash^{*}_{\Sigma,X} \top$. Hence, by Proposition 5.29, we can derive the trivial query $(\exists X)\, \top$ from $(\exists X)\, \rho$ and Γ using the computed substitution 1_X. ∎

Theorem 5.34 (Completeness of resolution)**.** *Assume that $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$ is a query-complete logic program and that $(\exists X)\, \rho$ is a Σ-query that admits an identity $(\forall X)\, \gamma \in \Gamma$. Then every $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$-solution to $(\exists X)\, \rho$ is also a computed $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$-answer to $(\exists X)\, \rho$.*

Proof. Let $\psi\colon X \to Y$ be a $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle$-solution to $(\exists X)\,\rho$. By Lemma 5.26 we know that $(\exists Y)\,\psi(\rho)$ can be derived from $(\exists X)\,\rho$ and its identity, $(\forall X)\,\gamma$, using the substitution ψ. Therefore, since $(\forall Y)\,\gamma \in \Gamma$, the query $(\exists Y)\,\psi(\rho)$ can also be derived from $(\exists X)\,\rho$ and Γ.

$$(\exists X)\,\rho \longrightarrow\!\!\!\!\rightarrow_{\Gamma,\psi} (\exists Y)\,\psi(\rho)$$

In addition, by Definition 5.7, the signature of variables Y is conservative, and $\langle\!\langle \Sigma, \Gamma \rangle\!\rangle \vDash^{\mathrm{lp}}_{\Sigma} (\forall Y)\,\psi(\rho)$. Hence, by Lemma 5.33, we can derive a trivial query $(\exists Y)\,\top$ from $(\exists Y)\,\psi(\rho)$ and Γ using the computed substitution 1_Y.

$$(\exists Y)\,\psi(\rho) \longrightarrow\!\!\!\!\rightarrow^{*}_{\Gamma,1_Y} (\exists Y)\,\top$$

Composing the two derivation chains outlined above yields a derivation of $(\exists Y)\,\top$ from $(\exists X)\,\rho$ using the substitution ψ. Consequently, ψ is a computed answer to $(\exists X)\,\rho$. ∎

6 Equational Logic Programming Revisited

Let us now focus on another prominent example of logic-programming language. Equational logic programming [25] integrates the machinery of its relational counterpart within algebraic specification in order to solve equations over abstract data types that are provided by given specifications. This is accomplished by replacing (*a*) the underlying single-sorted relational variant of first-order logic (without equality) with the many-sorted equational variant, (*b*) resolution with paramodulation, and (*c*) presentations (in the definition of logic programs) with program modules that are adequate for defining abstract data types. In this setting, the computation of the sum of `s 0` and `s 0` considered in Section 5 can be triggered by a query of the form

$$\frac{}{X_1 : \mathtt{Nat}} \vdash \mathtt{s}\,\mathtt{0} + \mathtt{s}\,\mathtt{0} = X_1$$

meant to be solved over a logic program that consists of two modules: `NAT`, defining the natural numbers as an abstract data type, and `ADD`, introducing the addition operation.

module `NAT` = **free**
  sort `Nat`
  op `0`: → `Nat`
  op `s_`: `Nat` → `Nat`

module `ADD` = `NAT` **then**
  op `_ + _`: `Nat Nat` → `Nat`
  clause $\mathtt{0} + M = M \leftarrow \frac{}{M : \mathtt{Nat}}$
  clause $(\mathtt{s}\,M) + N = \mathtt{s}\,(M + N) \leftarrow \frac{}{M, N : \mathtt{Nat}}$

The Generalized Substitution System

Equational logic programming is defined over the generalized substitution system QF-FOL$_=$ of the quantifier-free fragment of many-sorted first-order equational logic. Since most of the definitions and properties to check are straightforward adaptations of the definitions and properties discussed for QF-FOL$^{!}_{\neq}$ to the equational many-sorted setting of QF-FOL$_=$, here we only briefly review some of the most important concepts for our presentation. A more in-depth discussion of the components of QF-FOL$_=$ can be found, for example, in [26],[7] or in the recent monographs [11, 44].

Signatures. The *signatures* of QF-FOL$_=$ are pairs $\langle S, F \rangle$, where S is a set of *sorts* and F is a family $(F_{w\to s})_{w\in S^*, s\in S}$ of sets of *operation symbols*, indexed by *arities* and *sorts*. *Signature morphisms* $\varphi\colon \langle S, F \rangle \to \langle S', F' \rangle$ are defined

[7] It should be noted that in [26] the authors consider an even more general setting of order-sorted equational logic, with subsorts and overloading of operation symbols.

by functions $\varphi^{\mathrm{st}}\colon S \to S'$ between the sets of sorts, and families of functions $\varphi^{\mathrm{op}}_{w\to s}\colon F_{w\to s} \to F'_{\varphi^{\mathrm{st}}(w)\to\varphi^{\mathrm{st}}(s)}$, where $w \in S^*$ and $s \in S$.

Signatures of variables and substitutions. For every signature $\langle S, F\rangle$, a *signature of variables* X is a family of sets X_s, for $s \in S$, of variables $(x, F_{\varepsilon\to s})$,[8] often denoted $x\colon s$, where x is the name of the variable (distinct from the names of other variables in X) and s is its sort. In order to define substitutions, let us first recall that, for each sort $s \in S$, the set $\mathrm{T}_{F,s}$ of *F-terms* of sort s is the least set such that $\sigma(t_1,\ldots,t_n)\colon s \in \mathrm{T}_{F,s}$ for all operation symbols $\sigma \in F_{s_1\cdots s_n\to s}$ and all terms $t_i \in \mathrm{T}_{F,s_i}$. Then *substitutions* $\psi\colon X \to Y$ are families of maps $\psi_s\colon X_s \to \mathrm{T}_{F\cup Y,s}$, indexed by sorts $s \in S$, assigning a term over the extended signature $\langle S, F \cup Y\rangle$ to every variable of X.

Sentences, models, and the satisfaction relation. Similarly to the relational setting, the *sentences* over a QF-FOL$_=$-signature $\langle S, F\rangle$ are built from *equational atoms* $l = r$, where l and r are F-terms having the same sort, by iteration of the usual Boolean connectives. With respect to the semantics of $\langle S, F\rangle$, the *models* M of $\langle S, F\rangle$ interpret each sort $s \in S$ as a set M_s, called the *carrier set* of s in M, and each operation symbol $\sigma \in F_{s_1\cdots s_n\to s}$ as a function $M_\sigma\colon M_{s_1} \times \cdots \times M_{s_n} \to M_s$. *Homomorphisms* $h\colon M_1 \to M_2$ are families of functions $(h_s\colon M_{1,s} \to M_{2,s})_{s\in S}$ such that $h_s(M_{1,\sigma}(m_1,\ldots,m_n)) = M_{2,\sigma}(h_{s_1}(m_1),\ldots,h_{s_n}(m_n))$ for all operation symbols $\sigma \in F_{s_1\cdots s_n\to s}$ and all arguments $m_i \in M_{s_i}$.

$$\begin{array}{ccc} M_{1,s_1}\times\cdots\times M_{1,s_n} & \xrightarrow{M_{1,\sigma}} & M_{1,s} \\ {\scriptstyle h_{s_1}\times\cdots\times h_{s_n}}\downarrow & & \downarrow{\scriptstyle h_s} \\ M_{2,s_1}\times\cdots\times M_{2,s_n} & \xrightarrow{M_{2,\sigma}} & M_{2,s} \end{array}$$

Finally, the satisfaction relation is defined by induction on the structure of sentences, based on the evaluation of terms in models. For instance, an $\langle S, F\rangle$-model M satisfies an equational atom $l = r$ if and only if the terms l and r yield the same value in M.

Just as in the case of QF-FOL$^{\mathrm{I}}_{\neq}$, the signatures of variables inherit the sentences, the models, and the satisfaction relation of their corresponding extended first-order signatures. This means that for every signature of $\langle S, F\rangle$-variables X, $\mathrm{Sen}_{\langle S,F\rangle}(X)$, $\mathrm{Mod}_{\langle S,F\rangle}(X)$, and $\models_{\langle S,F\rangle,X}$ are defined as $\mathrm{Sen}(S, F \cup X)$, $\mathrm{Mod}(S, F \cup X)$, and $\models_{\langle S,F\cup X\rangle}$, respectively.

The following result relies on arguments similar to those of Proposition 4.11, based on the fact that the translation of variables along signature morphisms determines pushouts of first-order signatures, which in turn induce model-amalgamation squares.

Proposition 6.1. *The generalized substitution system* QF-FOL$_=$ *has model amalgamation.* ∎

Paramodulation

Paramodulation originated with the work of Robinson and Wos [40] as a refinement of resolution that is suitable for dealing with clauses and queries defined over first-order logic with equality. The definition of its corresponding goal-directed rules relies on the following notions of context and substitution of a term in a given context [44, 2].

For any sort s of a QF-FOL$_=$-signature $\langle S, F\rangle$, and for any signature of $\langle S, F\rangle$-variables X, an *X-context for s* is a term c over an extended signature of variables of the form $X \cup \{\Box\colon s\}$ that contains precisely one occurrence of the new variable $\Box$. The *substitution of a term $t \in \mathrm{T}_{F\cup X,s}$ in c*, denoted $c[t]$, is defined as the translation of c along the substitution $(\Box \mapsto t)\colon X \cup \{\Box\colon s\} \to X$ that maps every variable $x \in X$ to the term x, and $\Box$ to t.

Clauses and queries. Given a QF-FOL$_=$-signature $\langle S, F\rangle$, the local *clauses* over a signature of $\langle S, F\rangle$-variables X are (as in the relational case) implications of the form $\bigwedge H \Rightarrow (l = r)$, where H is a finite set of equational atoms and $l = r$ is an equational atom.

Similarly, the local *queries* over X are just finite conjunctions of equational atoms $\bigwedge Q$.

[8] We denote the empty arity by ε; hence, $F_{\varepsilon\to s}$ is the set of *constant-operation symbols* of $\langle S, F\rangle$ of sort s.

Goal-directed rules. Every signature of $\langle S, F\rangle$-variables X determines rules of the form

$$\bigwedge(\{c[l] = t\} \cup Q), (\bigwedge H \Rightarrow (l = r)) \Vdash_{\langle S,F\rangle,X} \bigwedge(\{c[r] = t\} \cup Q \cup H)$$

or

$$\bigwedge(\{t = c[l]\} \cup Q), (\bigwedge H \Rightarrow (l = r)) \Vdash_{\langle S,F\rangle,X} \bigwedge(\{t = c[r]\} \cup Q \cup H)$$

where Q and H are finite sets of equational atoms over X, l, and r are terms over X of some sort s, c is an X-context for s, and t is term over X with the same sort as c.

It is easy to verify that the above constructions satisfy all the necessary properties of Definition 5.2. Thus, they give rise to a (many-sorted) equational logic-programming framework that we denote by $\mathcal{FOL}_{=}$. Moreover, the abstract concept of resolution captures in this setting an extended version of what is known as paramodulation: a query $(\exists X_2) \bigwedge Q_2$ (over some signature $\langle S, F\rangle$) is derived by paramodulation in one step from another query $(\exists X_1) \bigwedge Q_1$ and a clause $(\forall Y_1) \bigwedge H_1 \Rightarrow (l_1 = r_1)$ using the computed substitution $\theta_1 : X_1 \to X_2$ if and only if there exists a substitution $\psi_1 : Y_1 \to X_2$ such that

$$\bigwedge \theta_1(Q_1), (\bigwedge \psi_1(H_1) \Rightarrow (\psi_1^{\mathrm{tm}}(l_1) = \psi_1^{\mathrm{tm}}(r_1))) \Vdash_{\langle S,F\rangle,X_2} \bigwedge Q_2.$$

The extension is in this case twofold: first, the derivation is not limited to most general unifiers (similarly to $\mathcal{FOL}^{1}_{\neq}$); second, the term $\psi_1^{\mathrm{tm}}(l_1)$ need not be equal with the θ_1-translation of a subterm in Q_1, but with a subterm in the θ_1-translation of Q_1.

All we need now in order to compute answers to queries defined over $\mathcal{FOL}_{=}$ is an analogue of Fact 5.20 providing an adequate characterization of trivial queries. To this end, notice first that a signature of $\langle S, F\rangle$-variables X is conservative if and only if there exists at least one F-term for each sort $s \in S$ such that $\mathrm{T}_{F\cup X,s}$ is not empty.

Fact 6.2. Let Y be a signature of $\langle S, F\rangle$-variables in $\mathcal{FOL}_{=}$ such that $\mathrm{T}_{F,s} = \emptyset$ implies $\mathrm{T}_{F\cup Y,s} = \emptyset$ for every $s \in S$. Then an $\langle S, F\rangle$-query $(\exists Y)\ \top$ is trivial if and only if $\top$ is a conjunction of equalities of the form $t = t$.

To illustrate the use of paramodulation, let Γ be the set consisting of the two clauses defined by the module `ADD` together with the identity clause $(\forall \emptyset)$ true $\Rightarrow$ $(\mathtt{0} = \mathtt{0})$. Then the sum of $\mathtt{s}\,\mathtt{0}$ and $\mathtt{s}\,\mathtt{0}$ can be computed according to the following chain of derivations:

$$\begin{aligned}
&(\exists\{X_1 : \mathtt{Nat}\})\ \mathtt{s}\,\mathtt{0} + \mathtt{s}\,\mathtt{0} = X_1 \\
&\quad \twoheadrightarrow_{\Gamma, X_1 \mapsto X_2} (\exists\{X_2 : \mathtt{Nat}\})\ \mathtt{s}\,(\mathtt{0} + \mathtt{s}\,\mathtt{0}) = X_2 && \text{using the first clause of } \Gamma \\
&\qquad && \quad \text{and the substitution } M \mapsto \mathtt{0}, N \mapsto \mathtt{s}\,\mathtt{0} \\
&\quad \twoheadrightarrow_{\Gamma, X_2 \mapsto X_3} (\exists\{X_3 : \mathtt{Nat}\})\ \mathtt{s}\,\mathtt{s}\,\mathtt{0} = X_3 && \text{using the second clause of } \Gamma \\
&\qquad && \quad \text{and the substitution } M \mapsto \mathtt{s}\,\mathtt{0} \\
&\quad \twoheadrightarrow_{\Gamma, X_3 \mapsto \mathtt{s}\,\mathtt{s}\,\mathtt{0}} (\exists\emptyset)\ \mathtt{s}\,\mathtt{s}\,\mathtt{0} = \mathtt{s}\,\mathtt{s}\,\mathtt{0} && \text{using the third (and last) clause of } \Gamma.
\end{aligned}$$

Program modules

In order to accommodate both the so-called loose semantics specific to relational logic programming, and the free semantics necessary for specifying abstract data types, equational logic programs may be defined by means of program modules like `NAT` and `ADD` that are built from (finite) presentations over $\mathcal{FOL}_{=}$, by iteration of *structuring operators* such as union, translation, and free semantics [43]. Various other operators dedicated, for example, to the derivation or to the extension of modules (often parameterized by classes of signature morphisms or of homomorphisms) have been considered in the specification literature [4, 44, 15]. To keep the presentation simple, we focus here only on the three aforementioned operators.

The equational logic-programming language $\mathcal{FOL}^{\mathrm{struc}}_{=}$ defines *programs* as 'terms' formed from the programs of $\mathcal{FOL}^{\mathrm{pres}}_{=}$ by repeated applications of the union, translation, and free-semantics operators listed below—together with the appropriate images of the resulting program modules under the functors Sign, Ax, and PMod. Similarly to the case of presentations, the *morphisms of programs* $\nu : P \to P'$ are morphisms of signatures $\nu : \mathrm{Sign}(P) \to \mathrm{Sign}(P')$ such that $M'\restriction_{\nu} \in |\mathrm{PMod}(P)|$ for every model $M' \in |\mathrm{PMod}(P')|$.

Union. For any two program modules P_1 and P_2 having the same signature Σ, $P_1 \cup P_2$ is also a program module, with

$$\begin{aligned}\mathrm{Sign}(P_1 \cup P_2) &= \Sigma\\ \mathrm{Ax}(P_1 \cup P_2) &= \mathrm{Ax}(P_1) \cup \mathrm{Ax}(P_2)\\ \mathrm{PMod}(P_1 \cup P_2) &= \mathrm{PMod}(P_1) \cap \mathrm{PMod}(P_2).\end{aligned}$$

Translation. For any program module P and any signature morphism $\varphi\colon \mathrm{Sign}(P) \to \Sigma'$, **translate** P **by** φ is also a program module, with

$$\begin{aligned}\mathrm{Sign}(\textbf{translate } P \textbf{ by } \varphi) &= \Sigma'\\ \mathrm{Ax}(\textbf{translate } P \textbf{ by } \varphi) &= \varphi(\mathrm{Ax}(P))\\ \mathrm{PMod}(\textbf{translate } P \textbf{ by } \varphi) &= \mathrm{Mod}(\varphi)^{-1}(\mathrm{PMod}(P)).\end{aligned}$$

Free semantics. For any two program modules P and P', and any signature morphism $\varphi\colon \mathrm{Sign}(P) \to \mathrm{Sign}(P')$, **free** P' **over** P **through** φ is also a program module, with

$$\begin{aligned}\mathrm{Sign}(\textbf{free } P' \textbf{ over } P \textbf{ through } \varphi) &= \mathrm{Sign}(P')\\ \mathrm{Ax}(\textbf{free } P' \textbf{ over } P \textbf{ through } \varphi) &= \mathrm{Ax}(P')\\ \mathrm{PMod}(\textbf{free } P' \textbf{ over } P \textbf{ through } \varphi) &= \text{the full subcategory of } \mathrm{PMod}(P') \text{ given by}\\ &\qquad \text{the models } M' \text{ that are free with respect to } \varphi \text{ over some model } M \text{ of } P.\end{aligned}$$

To describe the modules NAT and ADD, note that, for any program module P, **free** P is an abbreviation for **free** P **over** $\emptyset$ **through** ι, where ι is the inclusion $\emptyset \subseteq \mathrm{Sign}(P)$, and that, for any other program module P' such that $\mathrm{Sign}(P)$ is a subsignature of $\mathrm{Sign}(P')$, P **then** P' is an abbreviation for (**translate** P **by** ι) $\cup$ P', where ι is the inclusion $\mathrm{Sign}(P) \subseteq \mathrm{Sign}(P')$.

The soundness of paramodulation with respect to a given equational logic program follows from Theorem 5.23. As regards completeness, note that, unlike $(\mathcal{FOL}^{1}_{\neq})^{\mathrm{pres}}$, the logic-programming language $\mathcal{FOL}^{\mathrm{struc}}_{=}$ cannot be guaranteed to be query-complete due to its reliance upon the free-semantics operator [44]. However, it is known that every program module built only with the union and translation operators is semantically equivalent with a program defined over $\mathcal{FOL}^{\mathrm{pres}}_{=}$ [15], and moreover, that logic programs given as presentations $\langle \Sigma, \Gamma \rangle$ are query-complete whenever the term rewriting system generated by Γ is convergent, i.e. both confluent and terminating [2]. These observations lead to the following result.

Proposition 6.3. *Let P be a logic program defined over $\mathcal{FOL}^{\mathrm{struc}}_{=}$ using only the union and translation operators, such that the term rewriting system generated by $\mathrm{Ax}(P)$ is convergent. Then P is query-complete.* ■

7 Conclusions

We have advanced an abstract axiomatic theory of logic programming by identifying and examining in detail in an institution-theoretic setting two of the most basic principles of the paradigm: (*a*) the fact that each logic program has a rigorous mathematical semantics given by a class of models, and often by a 'standard' model of that class, determining the specific set of queries that can be positively answered, and (*b*) the existence of a sound (and in some cases complete) goal-directed procedure for computing answer-substitutions that confirm the validity of the positive answer received by a query. In this way, the present study unifies the relational and the equational variants of logic programming, and we can expect it to integrate with ease many other derived forms such as higher-order [35], behavioural [24], or even service-oriented logic programming [49]. The first two share many similarities with the equational variant presented in Section 6. The latter, however, is significantly different from all the other forms of logic programming mentioned above—even though the logic programs (known as repositories) and their corresponding operational semantics are defined likewise to their relational counterparts.

This is mainly due to the fact that the signatures of variables (formalized in this case as labelled hypergraphs) can no longer be treated as extensions of signatures (which provide the structures that can be used as labels for the hypergraphs underlying the signatures of variables). Consequently, the service-oriented substitutions cannot be captured by the institution-independent concept of substitution considered in [10], nor can they be expressed as generalized forms of signature morphisms as in [18], for example. Nevertheless, they can be easily shown to form the components of a generalized substitution system as defined in Section 4.

Our efforts have focused mainly on the development of a simple, yet sufficiently rich theoretical framework to allow the investigation of properties related to the satisfaction of quantified sentences, the generalization of Herbrand's theorem to abstract logic-programming languages, and even more, the definition of a sound and (conditionally) complete procedure for computing solutions to queries. This procedure relies on exploring a potentially infinite search space of queries, related through substitutions computed by means of resolution, in search of queries that are trivial, i.e. known to admit the simplest possible solutions. Its implementation is not complete because we do not commit to any particular search strategy, which could make use, for example, of most general unifiers and backtracking. However, it should be noted that under the present formalization of the search space, any such strategy would be sound, and moreover, complete if it ensured that the reachability of all trivial queries is maintained.

Apart from the obvious need to consider abstractly various other forms of logic programming, an interesting direction for future research is to examine the preservation and the reflection of answers along morphisms of logic programs (following the lines of [10]), as this may give us the possibility to search for solutions to queries in restricted contexts that correspond to 'subprograms', and then translate these solutions back to the original setting. Another open problem of practical importance is the development of an appropriate concept of map of logic-programming languages to capture, for instance, the encoding of relational logic programming into its equational correspondent based on the representation of relations as Boolean-valued operations (see, e.g. [11]). Even though from a denotational perspective we obtain an immediate answer in the form of the notion of morphism of generalized substitution systems (suggested in Definition 4.6), from an operational point of view the answer does not appear to be equally obvious since different logic-programming frameworks may be founded on highly different kinds of goal-directed rules.

Acknowledgements

The authors would like to thank Răzvan Diaconescu and Uwe Wolter for many useful discussions that led to the present form of this paper. This research has been supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-ID-PCE-2011-3-0439.

References

[1] Jiri Adámek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories: The Joy of Cats*. Dover books on mathematics. Dover Publications, 2009. reprint.

[2] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

[3] Edward K. Blum and Francesco Parisi-Presicce. The semantics of shared submodules specifications. In Hartmut Ehrig, Christiane Floyd, Maurice Nivat, and James W. Thatcher, editors, *Theory and Practice of Software Development*, volume 185 of *Lecture Notes in Computer Science*, pages 359–373. Springer, 1985.

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

[5] Rod M. Burstall and Joseph A. Goguen. The semantics of CLEAR, a specification language. In Dines Bjørner, editor, *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1979.

[6] Mihai Codescu and Daniel Găină. Birkhoff completeness in institutions. *Logica Universalis*, 2(2):277–309, 2008.

[7] Alain Colmerauer, Henry Kanoui, Philippe Roussel, and Robert Pasero. Un systéme de communication homme-machine en francais. Technical report, Groupe de Intelligence Artificielle, Faculté des Sciences de Luminy, Université de Aix-Marseille II, Marseille, 1973.

[8] Răzvan Diaconescu. Completeness of category-based equational deduction. *Mathematical Structures in Computer Science*, 5(1):9–40, 1995.

[9] Răzvan Diaconescu. Category-based constraint logic. *Mathematical Structures in Computer Science*, 10:373–407, 5 2000.

[10] Răzvan Diaconescu. Herbrand theorems in arbitrary institutions. *Information Processing Letters*, 90(1):29–37, 2004.

[11] Răzvan Diaconescu. *Institution-Independent Model Theory*. Studies in Universal Logic. Birkhäuser, 2008.

[12] Răzvan Diaconescu. Structural induction in institutions. *Information and Computation*, 209(9):1197–1222, 2011.

[13] Răzvan Diaconescu. An axiomatic approach to structuring specifications. *Theoretical Computer Science*, 433:20–42, 2012.

[14] Răzvan Diaconescu, Joseph A. Goguen, and Petros Stefaneas. Logical support for modularisation. In Gérard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge University Press, 1993.

[15] Răzvan Diaconescu and Ionuţ Ţuţu. On the algebra of structured specifications. *Theoretical Computer Science*, 412(28):3145–3174, 2011.

[16] José L. Fiadeiro, Antónia Lopes, and Laura Bocchi. An abstract model of service discovery and binding. *Formal Aspects of Computing*, 23(4):433–463, 2011.

[17] Daniel Găină. Forcing, downward Löwenheim-Skolem and Omitting Types theorems, institutionally. *Logica Universalis*, pages 1–30, 2013.

[18] Daniel Găină, Kokichi Futatsugi, and Kazuhiro Ogata. Constructor-based logics. *Journal of Universal Computer Science*, 18(16):2204–2233, 2012.

[19] Daniel Găină and Marius Petria. Completeness by forcing. *Journal of Logic and Computation*, 20(6):1165–1186, 2010.

[20] Joseph A. Goguen. What is unification? *Resolution of Equations in Algebraic Structures*, 1:217–261, 1989.

[21] Joseph A. Goguen and Rod M. Burstall. Some fundamental algebraic tools for the semantics of computation. Part 1: Comma categories, colimits, signatures and theories. *Theoretical Computer Science*, 31:175–209, 1984.

[22] Joseph A. Goguen and Rod M. Burstall. A study in the foundations of programming methodology: specifications, institutions, charters and parchments. In David H. Pitt, Samson Abramsky, Axel Poigné, and David E. Rydeheard, editors, *Category Theory and Computer Programming*, volume 240 of *Lecture Notes in Computer Science*, pages 313–333. Springer, 1986.

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

[24] Joseph A. Goguen, Grant Malcolm, and Tom Kemp. A hidden Herbrand theorem: combining the object and logic paradigms. *Journal of Logic and Algebraic Programming*, 51(1):1–41, 2002.

[25] Joseph A. Goguen and José Meseguer. Eqlog: Equality, types, and generic modules for logic programming. In *Logic Programming: Functions, Relations, and Equations*, pages 295–363. Prentice Hall, 1986.

[26] Joseph A. Goguen and José Meseguer. Models and equality for logical programming. In Hartmut Ehrig, Robert A. Kowalski, Giorgio Levi, and Ugo Montanari, editors, *Theory and Practice of Software Development*, volume 250 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 1987.

[27] Joseph A. Goguen and Grigore Roşu. Institution morphisms. *Formal Aspects of Computing*, 13(3–5):274–307, 2002.

[28] Jacques Herbrand. Investigations in proof theory. *From Frege to Gödel: A Source Book in Mathematical Logic*, 1879–1931:525–581, 1879.

[29] Robert A. Kowalski. Predicate logic as programming language. In *IFIP Congress*, pages 569–574, 1974.

[30] Robert A. Kowalski and Donald Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2(3/4):227–260, 1971.

[31] Saunders Mac Lane. *Categories for the working mathematician*. Graduate texts in mathematics. Springer, 1998.

[32] John W. Lloyd. *Foundations of logic programming*. Symbolic computation: Artificial intelligence. Springer, 1987.

[33] Manuel A. Martins, Alexandre Madeira, Răzvan Diaconescu, and Luís Soares Barbosa. Hybridization of institutions. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Algebra and Coalgebra in Computer Science*, volume 6859 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2011.

[34] José Meseguer. General logics. In Heinz-Dieter Ebbinghaus, José Fernández-Prida, Manuel Garrido, Daniel Lascar, and Mario Rodriquez-Artalejo, editors, *Logic Colloquium '87*, volume 129 of *Studies in Logic and the Foundations of Mathematics Series*, pages 275–329. Elsevier, 1989.

[35] José Meseguer. Multiparadigm logic programming. In Hélène Kirchner and Giorgio Levi, editors, *Algebraic and Logic Programming*, volume 632 of *Lecture Notes in Computer Science*, pages 158–200. Springer, 1992.

[36] Bernhard Möller, Andrzej Tarlecki, and Martin Wirsing. Algebraic specifications of reachable higher-order algebras. In Donald Sannella and Andrzej Tarlecki, editors, *Abstract Data Types*, volume 332 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 1987.

[37] Till Mossakowski. Comorphism-based Grothendieck logics. In Krzysztof Diks and Wojciech Rytter, editors, *Mathematical Foundations of Computer Science 2002*, volume 2420 of *Lecture Notes in Computer Science*, pages 593–604. Springer, 2002.

[38] Till Mossakowski. Institutional 2-cells and Grothendieck institutions. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 124–149. Springer, 2006.

[39] Till Mossakowski, Răzvan Diaconescu, and Andrzej Tarlecki. What is a logic translation? *Logica Universalis*, 3(1), 2009.

[40] George Robinson and Lawrence Wos. Paramodulation and theorem-proving in first-order theories with equality. In Bernard Meltzer and Donald Michie, editors, *Machine intelligence*, volume 4, pages 135–150. Edinburgh University Press, 1983.

[41] John A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[42] Donald Sannella and Andrzej Tarlecki. Building specifications in an arbitrary institution. In Gilles Kahn, David B. MacQueen, and Gordon D. Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 337–356. Springer, 1984.

[43] Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76(2/3):165–210, 1988.

[44] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2011.

[45] Andrzej Tarlecki. Quasi-varieties in abstract algebraic institutions. *Journal of Computer and System Sciences*, 33(3):333–360, 1986.

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

[47] Andrzej Tarlecki. Towards heterogeneous specifications. In Dov M. Gabbay and Maarten van Rijke, editors, *Frontiers of Combining Systems*, volume 2, pages 337–360. Research Studies Press, 2000.

[48] Andrzej Tarlecki, Rod M. Burstall, and Joseph A. Goguen. Some fundamental algebraic tools for the semantics of computation. Part 3: Indexed categories. *Theoretical Computer Science*, 91(2):239–264, 1991.

[49] Ionuţ Ţuţu and José L. Fiadeiro. A logic-programming semantics of services. In Reiko Heckel and Stefan Milius, editors, *Algebra and Coalgebra in Computer Science*, volume 8089 of *Lecture Notes in Computer Science*, pages 299–313. Springer, 2013.

[50] Ionuţ Ţuţu and José L. Fiadeiro. Service-oriented logic programming. 2014. submitted.