A Categorical Semantics of Relational First-Order Logic*

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
Wolfgang.Schreiner@risc.jku.at

Valerie Novitzká William Steingartner
Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice, Slovakia
Valerie.Novitzka@tuke.sk
William.Steingartner@tuke.sk

March 11, 2019

Abstract

We present a categorical formalization of a variant of first-order logic. Unlike other texts on this topic, the goal of this paper is to give a very transparent and self-contained account without requiring more background than basic logic and set theory. Our focus is to show how the semantics of first order formulas can be derived from their usual deduction rules. For understanding the core ideas, it is not necessary to investigate the internal term structure of atomic formulas, thus we abstract atomic formulas to (syntactically opaque) relations; in this sense our variant of first-order logic is "relational". While the derived semantics is based on categorical principles, it is nevertheless "constructive" in that it describes explicit computations of the truth values of formulas. We demonstrate this by modeling the categorical semantics in the RISCAL (RISC Algorithm Language) system which allows us to validate the core propositions by automatically checking them in finite models.

*Supported by the Austrian OEAD WTZ program and the Slovak SRDA agency under the contract SK 14/2018 "SemTech — Semantic Technologies for Computer Science Education" and by the Johannes Kepler University Linz, Linz Institute of Technology (LIT), Project LOGTECHEDU "Logic Technology for Computer Science Education".

1 Introduction

Most introductions to first-order logic first define the syntax of formulas, then formalize their meaning in the form established in the 1930s by Tarski [19] (essentially what we today call in programming language theory a "denotational semantics" [9]), then introduce a deduction calculus, and finally show the soundness and completeness of this calculus with respect to the semantics: if a formula can be derived in the calculus, it is true according to the semantics, and vice versa. These relationships between truth and derivability have to be established, because there is no self-evident link between the semantics of a formula and the deduction rules associated to it. Historically, deduction actually came first; the soundness of a deduction calculus was established by showing that it could not lead to apparent inconsistencies, i.e., that not a formula and its negation could be derived. It was Tarski who first gave a meaning to formulas that was independent of deduction.

However, like it did since the 1940s to many other mathematical areas, category theory [3, 6, 15], the general theory of mathematical structures, can shed an alternative light also to first-order logic. It does so by considering logical notions as special instances of "universal" constructions, where a value of interest is determined

- first, by depicting the core property that the value shall satisfy and
- second, by a criterion to choose among all values that satisfy the property a canonical one.

It was eventually recognized that by such universal constructions the semantics of the connectives of propositional logic could be determined directly from their associated introduction and elimination rules. However, it took until the late 1960s until Lawvere gained the fundamental insight that this idea could be also applied to the quantifiers of first-order logic [5], thus establishing a direct relationship between its semantics and its proof calculus.

However, this insight has not yet got a foothold in basic texts on logic and its basic education. A main reason may be that the corresponding material is found mostly in texts on category theory and its applications where it is dispersed among examples of the application of categorical notions without a clear central presentation. Furthermore, the general treatment of first-order logic with terms and variables requires a complex machinery [4] which is much beyond the scope of basic introductions. Reasonably compact introductions can be found, e.g., in Section 2.1.10 of [1], in [7], in Section 1.6 of [2] (however, in the context of type theory rather than classical first-order logic), in Section 9.5 of [3] (the treatment of quantifiers only), and in Section 7.1.12 of [15] (again only the treatment of quantifiers).

The goal of the present elaboration is to give a compact introduction to a categorical version of first-order logic that is fully self-contained, only introduces the categorical notions relevant for the stated purpose, and presents them from the point of view of the intended application. For this purpose it elaborates a simple but completely formalized syntactic and semantic framework of first-order logic that represents the background of the discussion, without gaps and inconsistencies (hopefully). As a deliberate decision, this framework does not address the syntax and semantics of terms but abstracts atomic formulas to opaque relations; this allows to focus the discussion on the essentials. However, to really describe a reasonably close relative of first-order logic, this framework is (in contrast to other presentations) not based on relations of fixed arity, i.e.,

with a fixed number of variables; rather we consider relations of infinite arity, i.e., with infinitely many variables. However, only finitely many variables may influence the truth value of the relation, which represents the effect that a classical atomic formula can only reference finitely many variables. The overall result is (as we hope) a slick and elegant presentation.

The remainder of this paper is structured as follows: In Section 2 we define a term-free variant of first-order logic and give it a semantics in the usual style on the basis of set-theoretic notions. In Section 3, we introduce those categorical notions that are necessary for understanding the following elaboration and discuss their relationships. The core of this paper is Section 4 where we elaborate the categorical formulation of the semantics of our variant of first-order logic. In Section 5 we demonstrate that this semantics is constructive by modeling it in the RISCAL system [8], which allows us to automatically check the core propositions in finite models. Section 6 concludes our presentation and gives an outlook on future work.

2 A Relational First-Order Logic

In this section, we introduce a simplified variant of first-order logic that abstracts from the syntactic structure of atomic formulas and thus copes without the concept of terms, constants, function symbols, predicate symbols, and all of the associated semantic machinery. Towards this goal, atomic formulas are replaced by relations over assignments (maps of variables to values) that are constrained to only depend on a finite number of variables; we will call such relations "predicates". Consequently also the semantics of every non-atomic formula is a predicate.

We begin with some standard notions.

Axiom 1 (Variables and Values)**.** Let *Var* denote an arbitrary infinite and enumerable set; we call the elements of this set *variables*. Furthermore, let *Val* denote an arbitrary non-empty set; we call the elements of this set *values*.

Definition 1 (Assignments)**.** We define $Ass := Var \to Val$ as the set of all mappings of variables to values; we call the elements of this set *assignments*. Thus for every assignment $a \in Ass$ and every variable $x \in Var$, we have $a(x) \in Val$.

Definition 2 (Updates)**.** Let $a \in Ass$ be an assignment, $x \in Var$ a variable, and $v \in Val$ a value. We define the *update* assignment $a[x \mapsto v] \in Ass$ as follows:

$$a[x \mapsto v](y) := \begin{cases} v & \text{if } x = y \\ a[x] & \text{otherwise} \end{cases}$$

Consequently, $a[x \mapsto v]$ is identical to a except that it maps variable x to value v.

Proposition 1 (Update Properties)**.** *Let $a \in Ass$ be an assignment, $x, y \in Var$ variables, and $v, v_1, v_2 \in Var$ values. Then we have the following properties:*

$$a[x \mapsto a(x)] = a$$
$$a[x \mapsto v_1][x \mapsto v_2] = a[x \mapsto v_2]$$
$$x \neq y \Rightarrow a[x \mapsto v_1][y \mapsto v_2] = a[y \mapsto v_2][x \mapsto v_1]$$

Proof. Directly from the definitions. □

The properties of assignments listed above (and only these) will be of importance in the subsequent proofs.

Now we turn to the fundamental semantic notions.

Definition 3 (Relations)**.** We define $Rel := \mathcal{P}(Ass)$ as the set of all sets of assignments; we call the elements of this set "relations". Consequently, a relation is a set of assignments.

Definition 4 (Variable Independence)**.** We state that relation $R \in Rel$ *is independent of* variable $x \in Var$, written as $R \perp\!\!\!\perp x$, if and only if the following holds:

$$\forall a \in Ass, v_1, v_2 \in Val.\ a[x \mapsto v_1] \in R \Leftrightarrow a[x \mapsto v_2] \in R$$

Consequently, if $R \perp\!\!\!\perp x$, the value of x in any assignment a does not influence whether a is in R. We say that R *depends on* x if $R \perp\!\!\!\perp x$ does *not* hold.

We transfer the central syntactic property of atomic formulas (they can only refer to finitely many variables) to its semantic counterpart.

Definition 5 (Predicates)**.** A relation $R \in Rel$ is a *predicate*, if it only depends on finitely many variables. We denote by $Pred$ the set of all predicates and by $Pred_x := \{P \in Pred \mid P \perp\!\!\!\perp x\}$ the subset of all predicates that are independent of x.

Now we are ready to introduce the central entities of our discourse.

Definition 6 (Abstract Syntax of Formulas)**.** We define For as that smallest set of abstract syntax trees in which every element $F \in For$ is generated by an application of a rule of the following context-free grammar (where $P \in Pred$ denotes an arbitrary predicate and $x \in Var$ denotes an arbitrary variable):

$$\begin{aligned} F ::= {} & P \mid \top \mid \bot \\ & \mid \neg F \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid F_1 \rightarrow F_2 \mid F_1 \leftrightarrow F_2 \\ & \mid \forall x.\ F \mid \exists x.\ F \end{aligned}$$

We call the elements of this set *formulas*.

In this definition, the role of a classic atomic predicate $p(t_1, \ldots, t_n)$ with argument terms $t_1, \ldots, t_k$ in which n variables $x_1, \ldots, x_n$ occur freely is abstracted to a predicate P that depends on variables $x_1, \ldots, x_n$.

Now we establish the relationship between the syntax and semantics of formulas.

Definition 7 (Semantics of Formulas)**.** Let $F \in \mathit{For}$ be a formula. We define the relation $[\![F]\!] \in \mathit{Rel}$, called the *semantics* of F, by induction on the structure of F:

$$
\begin{aligned}
[\![P]\!] &:= \{a \in \mathit{Ass} \mid a \in P\} \\
[\![\top]\!] &:= \{a \in \mathit{Ass} \mid \text{true}\} \\
[\![\bot]\!] &:= \{a \in \mathit{Ass} \mid \text{false}\} \\
[\![\neg F]\!] &:= \{a \in \mathit{Ass} \mid a \notin [\![F]\!]\} \\
[\![F_1 \wedge F_2]\!] &:= \{a \in \mathit{Ass} \mid a \in [\![F_1]\!] \text{ and } a \in [\![F_2]\!]\} \\
[\![F_1 \vee F_2]\!] &:= \{a \in \mathit{Ass} \mid a \in [\![F_1]\!] \text{ or } a \in [\![F_2]\!]\} \\
[\![F_1 \to F_2]\!] &:= \{a \in \mathit{Asss} \mid a \notin [\![F_1]\!] \text{ or } a \in [\![F_2]\!]\} \\
[\![F_1 \leftrightarrow F_2]\!] &:= \{a \in \mathit{Ass} \mid (a \in [\![F_1]\!] \text{ and } a \in [\![F_2]\!]) \text{ or } (a \notin [\![F_1]\!] \text{ and } a \notin [\![F_2]\!])\} \\
[\![\forall x.\, F]\!] &:= \{a \in \mathit{Ass} \mid a[x \mapsto v] \in [\![F]\!], \text{ for all } v \in \mathit{Val}\} \\
[\![\exists x.\, F]\!] &:= \{a \in \mathit{Ass} \mid a[x \mapsto v] \in [\![F]\!], \text{ for some } v \in \mathit{Val}\}
\end{aligned}
$$

Above definition is well-defined in that clearly every formula denotes a relation. To show that formulas indeed denote predicates some more work is required.

Proposition 2 (Quantified Formulas and Variable Independence)**.** *For every variable $x \in \mathit{Var}$ and formula $F \in \mathit{For}$, we have $[\![\forall x.\, F]\!] \perp\!\!\!\perp x$ and $[\![\exists x.\, F]\!] \perp\!\!\!\perp x$, i.e., the semantics of quantified formulas does not depend on x.*

Proof. We prove this proposition by reductio ad absurdum.

First assume that $[\![\forall x.\, F]\!]$ depends on x. Then we have some assignment $a \in [\![\forall x.\, F]\!]$ and some values $v_1, v_2 \in \mathit{Val}$ such that $a[x \mapsto v_1] \in [\![\forall x.\, F]\!]$ and $a[x \mapsto v_2] \notin [\![\forall x.\, F]\!]$. From $a[x \mapsto v_2] \notin [\![\forall x.\, F]\!]$ we have some $v \in \mathit{Val}$ with $a[x \mapsto v_2][x \mapsto v] \notin [\![F]\!]$ and thus $a[x \mapsto v] \notin [\![F]\!]$. However, $a[x \mapsto v_1] \in [\![\forall x.\, F]\!]$ implies $a[x \mapsto v_1][x \mapsto v] \in [\![F]\!]$ and thus $a[x \mapsto v] \in [\![F]\!]$, which represents a contradiction.

Now assume that $[\![\exists x.\, F]\!]$ depends on x. Then we have some assignment $a \in [\![\exists x.\, F]\!]$ and some values $v_1, v_2 \in \mathit{Val}$ such that $a[x \mapsto v_1] \in [\![\exists x.\, F]\!]$ and $a[x \mapsto v_2] \notin [\![\exists x.\, F]\!]$. From $a[x \mapsto v_1] \in [\![\exists x.\, F]\!]$ we have some $v \in \mathit{Val}$ with $a[x \mapsto v_1][x \mapsto v] \in [\![F]\!]$ and thus $a[x \mapsto v] \in [\![F]\!]$. However, $a[x \mapsto v_2] \notin [\![\exists x.\, F]\!]$ implies $a[x \mapsto v_2][x \mapsto v] \notin [\![F]\!]$ and thus $a[x \mapsto v] \notin [\![F]\!]$, which represents a contradiction. □

Proposition 3 (Formula Semantics and Predicates)**.** *For every formula $F \in \mathit{For}$, we have $[\![F]\!] \in \mathit{Pred}$, i.e., the semantics of F is a predicate.*

Proof. The proof proceeds by induction over the structure of F.

- If $F = P$, we have $[\![F]\!] = \{a \in \mathit{Ass} \mid a \in P\} = P \in \mathit{Pred}$.
- If $F \in \{\top, \bot\}$, there are no x, a, v_1, v_2 such that $a[x \mapsto v_1] \in [\![F]\!]$ and $a[x \mapsto v_2] \notin [\![F]\!]$, because for $F = \top$ the second condition must be false and for $F = \bot$ the first one; thus F does not depend on any variable.

- If $F = \neg F_1$, by the induction hypothesis, we may assume that $[\![F_1]\!]$ depends only on the variables in some finite variable set X. From the definition of $[\![F]\!]$, it is then easy to show that also $[\![\neg F_1]\!]$ depends only on the variables in X.

- If $F \in \{F_1 \wedge F_2, F_1 \vee F_2, F_1 \rightarrow F_2, F_1 \leftrightarrow F_2\}$, we may assume by the induction hypothesis that $[\![F_1]\!]$ depends only on the variables in some finite set X_1 while P_2 only depends on the variables in some finite set X_2. From the definition of $[\![F]\!]$, it is then easy to show that $[\![F]\!]$ depends only on the variables in the finite set $X_1 \cup X_2$.

- If $F \in \{\forall x.\ F_1, \exists x.\ F_1\}$, we may assume by the induction hypothesis that $[\![F_1]\!]$ only depends on the variables in some finite variable set X. We are now going to show that $[\![F]\!]$ only depends on the variables in the finite set $X \backslash \{x\}$. Actually we assume that this is not the case and show a contradiction. From this assumption and Proposition 2, we have a variable $y \neq x \wedge y \notin X$ on which F depends; thus we have an assignment a and values v_1, v_2 such that $a[y \mapsto v_1] \in [\![F]\!]$ and $a[y \mapsto v_2] \notin [\![F]\!]$.

  If $F = \forall x.\ F_1$, from $a[y \mapsto v_2] \notin [\![F]\!]$ we have a $v \in \mathit{Val}$ with $a[y \mapsto v_2][x \mapsto v] \notin [\![F_1]\!]$ and thus (since $y \neq x$) $a[x \mapsto v][y \mapsto v_2] \notin [\![F_1]\!]$. From $a[y \mapsto v_1] \in [\![F]\!]$ we know $a[y \mapsto v_1][x \mapsto v] \in [\![F_1]\!]$ and thus $a[x \mapsto v][y \mapsto v_1] \in [\![F_1]\!]$. Thus $[\![F_1]\!]$ depends on a variable $y \notin X$ which contradicts the induction assumption.

  If $F = \exists x.\ F_1$, from $a[y \mapsto v_1] \in [\![F]\!]$ we have a $v \in \mathit{Val}$ with $a[y \mapsto v_1][x \mapsto v] \in [\![F_1]\!]$ and thus (since $y \neq x$) $a[x \mapsto v][y \mapsto v_1] \in [\![F_1]\!]$. From $a[y \mapsto v_2] \notin [\![F]\!]$ we know $a[y \mapsto v_2][x \mapsto v] \notin [\![F_1]\!]$ and thus $a[x \mapsto v][y \mapsto v_2] \notin [\![F_1]\!]$. Thus $[\![F_1]\!]$ depends on a variable $y \notin X$ which contradicts the induction assumption.

This completes our proof. □

In the following we transfer the classical model-theoretic notions to our framework.

Definition 8 (Satisfaction)**.** Let $a \in \mathit{Ass}$ be an assignment and $F \in \mathit{For}$ be a formula. We define $a \models F$ (read: *a satisfies F*) as follows:

$$a \models F :\Leftrightarrow a \in [\![F]\!]$$

Definition 9 (Validity)**.** Let $F \in \mathit{For}$ be a formula. We define $\models F$ (read: F is *valid*) as follows:

$$\models F :\Leftrightarrow \forall a \in \mathit{Ass}.\ a \models F$$

Definition 10 (Logical Consequence)**.** Let $F, G \in \mathit{For}$ be formulas. We define $F \models G$ (read: G is a *logical consequence* of F) as follows:

$$F \models G :\Leftrightarrow \forall a \in \mathit{Ass}.\ a \models F \Rightarrow a \models G$$

Definition 11 (Logical Equivalence)**.** Let $F, G \in \mathit{For}$ be formulas. We define $F \equiv G$ (read: F and G are *logically equivalent*) as follows:

$$F \equiv G :\Leftrightarrow \forall a \in \mathit{Ass}.\ a \models F \Leftrightarrow a \models G$$

Proposition 4 (Logical Consequence and Logical Equivalence)**.** *Let $F, G \in For$ be formulas. Then we have the following equivalences:*

- $(F \models G) \Leftrightarrow (\models F \to G)$
- $(F \models G) \Leftrightarrow (\llbracket F \rrbracket \subseteq \llbracket G \rrbracket)$
- $(F \equiv G) \Leftrightarrow (\models F \leftrightarrow G)$
- $(F \equiv G) \Leftrightarrow (\llbracket F \rrbracket = \llbracket G \rrbracket)$

Proof. Directly from the definitions. □

Thus logical consequence on the meta-level coincides with implication on the formula level and with the subset relation on the semantic level. Furthermore, logical equivalence on the meta-level coincides with equivalence on the formula level and with the equality relation on the semantic level.

In the following, we establish a set-theoretic interpretation of the logical operations of our formula language.

Definition 12 (Complement)**.** We define the *complement* $\overline{R} \in Rel$ of relation $R \in Rel$ as the relation $\overline{R} := Ass \backslash R$. Consequently an assignment is in $\overline{R}$ if and only if it is not in R.

Proposition 5 (Propositional Semantics as Set Operations)**.** *Let $F, F_1, F_2 \in For$ be formulas. We then have the following equalities:*

$$\begin{aligned}
\llbracket P \rrbracket &= P \\
\llbracket \top \rrbracket &= Ass \\
\llbracket \bot \rrbracket &= \emptyset \\
\llbracket \neg F \rrbracket &= \overline{\llbracket F \rrbracket} \\
\llbracket F_1 \wedge F_2 \rrbracket &= \llbracket F_1 \rrbracket \cap \llbracket F_2 \rrbracket \\
\llbracket F_1 \vee F_2 \rrbracket &= \llbracket F_1 \rrbracket \cup \llbracket F_2 \rrbracket \\
\llbracket F_1 \to F_2 \rrbracket &= \overline{\llbracket F_1 \rrbracket} \cup \llbracket F_2 \rrbracket \\
\llbracket F_1 \leftrightarrow F_2 \rrbracket &= (\llbracket F_1 \rrbracket \cap \llbracket F_2 \rrbracket) \cup (\overline{\llbracket F_1 \rrbracket} \cap \overline{\llbracket F_2 \rrbracket})
\end{aligned}$$

Proof. Directly from the definition of the semantics. □

While above results are quite intuitive, a corresponding set-theoretic interpretation of quantified formulas is not. In the following we only state the plain result without indication of how it can be intuitively understood; we will delegate this explanation to Section 4 where the categorical framework will provide us with the adequate insight.

Proposition 6 (Quantifier Semantics as Set Operations)**.** *Let $F \in For$ be a formula. We then have the following equalities:*

$$\begin{aligned}
\llbracket \forall x.\, F \rrbracket &= \bigcup \{P \in Pred \mid P \perp\!\!\!\perp x \wedge P \subseteq \llbracket F \rrbracket\} \\
\llbracket \exists x.\, F \rrbracket &= \bigcap \{P \in Pred \mid P \perp\!\!\!\perp x \wedge \llbracket F \rrbracket \subseteq P\}
\end{aligned}$$

In other words, $[\![\forall x.\ F]\!]$ is the weakest predicate P ("weakest" in the sense of the largest set) that is independent of x and that satisfies the property $P \subseteq [\![F]\!]$ while $[\![\exists x.\ F]\!]$ is the strongest predicate P ("strongest" in the sense of the smallest set) that is independent of x and that satisfies the property $[\![F]\!] \subseteq P$.

Proof. First we take arbitrary $a \in Ass$ and show

$$a \in [\![\forall x.\ F]\!] \Leftrightarrow \big(\exists P \in Pred.\ P \perp\!\!\!\perp x \wedge P \subseteq [\![F]\!] \wedge a \in P\big)$$

$\Rightarrow$: We assume $a \in [\![\forall x.\ F]\!]$ and prove for $P := [\![\forall x.\ F]\!]$

$$P \in Pred \tag{a}$$
$$P \perp\!\!\!\perp x \tag{b}$$
$$P \subseteq [\![F]\!] \tag{c}$$
$$a \in P \tag{d}$$

From Proposition 3, we have (a). From Proposition 2, we have (b). From $a \in [\![\forall x.\ F]\!]$ we have (d). To show (c), we take arbitrary assignment $a_0 \in P$ and show $a_0 \in [\![F]\!]$. From $a_0 \in P$, we know $a_0[x \mapsto v] \in [\![F]\!]$ for $v := a_0(x)$. Since $a_0[x \mapsto a_0(x)] = a_0$, we thus know $a_0 \in [\![F]\!]$.

$\Leftarrow$: We assume for some $P \in Pred$

$$P \perp\!\!\!\perp x \tag{1}$$
$$P \subseteq [\![F]\!] \tag{2}$$
$$a \in P \tag{3}$$

and prove $a \in [\![\forall x.\ F]\!]$. For this we take arbitrary $v \in Val$ and prove $a[x \mapsto v] \in [\![F]\!]$. From (2), it suffices to show $a[x \mapsto v] \in P$. From (1), we know

$$\forall v_1, v_2 \in Val.\ a[x \mapsto v_1] \in P \Leftrightarrow a[x \mapsto v_2] \in P \tag{4}$$

From (3) and $a = a[x \mapsto a(x)]$, we know $a[x \mapsto v_0] \in P$ for $v_0 := a(x)$. Thus with (4) we know $a[x \mapsto v] \in P$.

Now we take arbitrary $a \in Ass$ and show

$$a \in [\![\exists x.\ F]\!] \Leftrightarrow \big(\forall P \in Pred.\ P \perp\!\!\!\perp x \wedge [\![F]\!] \subseteq P \Rightarrow a \in P\big)$$

$\Rightarrow$: We assume $a \in [\![\exists x.\ F]\!]$ and take arbitrary but fixed $P \in Pred$ for which we assume

$$P \perp\!\!\!\perp x \tag{5}$$
$$[\![F]\!] \subseteq P \tag{6}$$

Our goal is to show $a \in P$. From $a \in [\![\exists x.\ F]\!]$, we know $a[x \mapsto v] \in [\![F]\!]$ for some $v \in Val$. From (6), we thus know $a[x \mapsto v] \in P$. From (5), we thus know $a[x \mapsto a(x)] \in P$. Since $a[x \mapsto a(x)] = a$, we thus know $a \in P$.

$\Leftarrow$: We assume

$$\forall P \in Pred.\ P \perp\!\!\!\perp x \wedge [\![F]\!] \subseteq P \Rightarrow a \in P \tag{7}$$

and prove $a \in [\![\exists x.\ F]\!]$. From (7) instantiated with $P := [\![\exists x.\ F]\!]$ and Propositions 3 and 2, it suffices to prove $[\![F]\!] \subseteq [\![\exists x.\ F]\!]$. Take arbitrary assignment $a_0 \in [\![F]\!]$. Since $a_0[x \mapsto a(x)] = a$, we thus have $a_0[x \mapsto v] \in [\![F]\!]$ for $v := a(x)$ and thus $a_0 \in [\![\exists x.\ F]\!]$. □

3 Category Theory

In this section we discuss those aspects of category theory that are relevant for the subsequent categorical formulation of our relational first order logic.

Basic Notions We begin with the basic notions of category theory.

Definition 13 (Category)**.** A category $\mathcal{C}$ is is a triple $\langle O, A, \circ \rangle$ of the following components:

- A class O of elements called $\mathcal{C}$*-objects* or just *objects*.
- A class A of elements called $\mathcal{C}$*-arrows* or just *arrows*. Each arrow has a *source* object and a *target* object from O; we write $f\colon a \to b$ to indicate that f is an arrow with source a and target b. We write $\mathcal{C}(a, b)$ to denote the class of all arrows of A with source a and target b (called the *hom-class* of all arrows from a to b). For every object x in O, A contains an arrow $id_x\colon x \to x$ called the *identity* arrow for x.
- A binary operation $\circ$ on arrows called *composition*. For all arrows $f\colon a \to b$ and $g\colon b \to c$ we have $(g \circ f)\colon a \to c$. Furthermore, the composition satisfies the following axioms:
  - Associativity: $(h \circ g) \circ f = h \circ (g \circ f)$, for all arrows $f\colon a \to b, g\colon b \to c, h\colon c \to d$.
  - Identity: $id_b \circ f = f = f \circ id_a$, for all arrows $f\colon a \to b$.

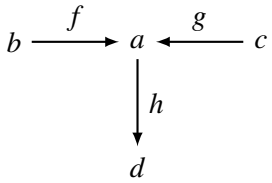
Example 1 (Category of Sets)**.** We introduce the category *SET* which contains as objects all sets and every triple $\langle f^\lambda, A, B \rangle$ as arrow $f\colon A \to B$ where f^λ is a (total) function from set A to set B. The identity arrow $id_A\colon A \to A$ is defined by $id_A^\lambda\colon A \to A,\ id_A^\lambda(x) := x$; composition is defined by $(g \circ f)^\lambda\colon A \to C,\ (g \circ f)^\lambda(x) := g(f(x))$ for arrows $g\colon B \to C$ and $f\colon A \to B$. □

Example 2 (Category of Subsets)**.** Given a set S we introduce the category $SUB(S)$ which contains as objects all subsets of S and every tuple $\langle A, B \rangle$ as arrow $f\colon A \to B$ for which $A \subseteq B$ holds; thus a unique arrow with source A and target B exists if and only if $A \subseteq B$. The identity arrow is defined as $id_A := \langle A, A \rangle$; composition is defined as $g \circ f := \langle A, C \rangle$ for arrows $g\colon B \to C$ and $f\colon A \to B$. □

Categories are often depicted in the form of diagrams where

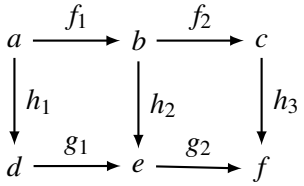
- every point in the diagram represents an object of the category;
- every arrow between two points in the diagram represents in the category an arrow between the corresponding two objects.

These diagrams do typically not show the identity arrows and arrows whose existence is implied by the composition of the depicted arrows. For example, take the following diagram:



The category depicted by this diagram has four objects a, b, c, d and nine arrows: the identities id_a, id_b, id_c, id_d, the depicted arrows $f\colon b \to a$, $g\colon c \to a$, $h\colon a \to d$, and the implied arrows $(h \circ f)\colon b \to d$, $(h \circ g) : c \to d$.

Categorical propositions are often depicted in the form of "commutative diagrams" where every connected "path" of one ore more arrows in the diagram represents in the category the composition of these arrows; if two paths in the diagram have the same start point and the same end point, they represent in the category the same arrow (i.e., the compositions denoted by the paths yield the same result). For example, take the following commutative diagram:



This diagram states the propositions $h_2 \circ f_1 = g_1 \circ h_1$ and $h_3 \circ f_2 = g_2 \circ h_2$ which imply the proposition $h_3 \circ f_2 \circ f_1 = g_2 \circ g_1 \circ h_1$.

Some more fundamental notions follow.

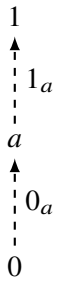
Definition 14 (Isomorphism)**.** Let $\mathcal{C}$ be a category and a, b be $\mathcal{C}$-objects a, b. Then we have $a \simeq b$ (read: a and b are *isomorphic*) if there are $\mathcal{C}$-arrows $f\colon a \to b$ and $g\colon b \to a$, called *isomorphisms*, such that $g \circ f = id_a$ and $f \circ g = id_b$.

Definition 15 (Subcategory)**.** A category $\mathcal{C}$ is a *subcategory* of category $\mathcal{D}$ if every $\mathcal{C}$-object is also a $\mathcal{D}$-object, every $\mathcal{C}$-arrow is also a $\mathcal{D}$-arrow, every identity arrow in $\mathcal{C}$ is also an identity arrow in $\mathcal{D}$, and $g \circ_C f = g \circ_D f$ for all $\mathcal{C}$-arrows $f\colon a \to b$ and $g\colon b \to c$, where $\circ_C$ denotes the composition in $\mathcal{C}$ and $\circ_D$ denotes the composition in $\mathcal{D}$.

Object Constructions We are now introducing constructions of categorical objects that will subsequently play an important role in the categorical formulation of relational first-order logic.

Definition 16 (Initial and Final Objects)**.** Let $\mathcal{C}$ be a category. A $\mathcal{C}$-object 0 is *initial* if for every $\mathcal{C}$-object a there exists exactly one arrow $0_a : 0 \to a$. A $\mathcal{C}$-object 1 is *final* if for every $\mathcal{C}$-object a there exists exactly one arrow $1_a : a \to 1$.

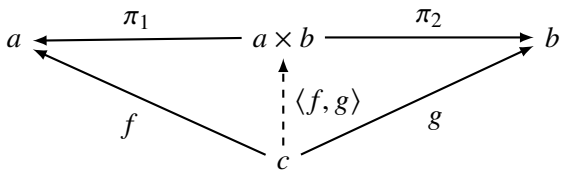
The following diagram illustrates the arrows of an initial object 0 and a final object 1 with respect to an arbitrary object a:



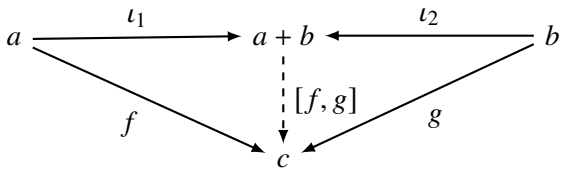
This construction of initial/final objects is "universal" in the sense that it describes a class of entities (objects and accompanying arrows) that share a common property and picks from this class an entity whose characterizing property is the existence of exactly one arrow from/to every entity of this class. This defines the entity uniquely up to isomorphism. Further instances of such constructions will be given later.

Example 3 (Initial and Final Object)**.** In the category *SET* of all sets (introduced in Example 1), the only initial object 0 is the empty set $\emptyset$, where arrow $0_A\colon 0 \to A$ is determined by the function $0^\lambda_A\colon \emptyset \to A$, $0^\lambda_A := \emptyset$ that is not defined on any argument. Every singleton set $\{e\}$ with arbitrary element e is final with arrow $1_A\colon A \to 1$ determined by the function $1^\lambda_A : A \to \{e\}$, $1^\lambda_A(x) := e$. In the category $SUB(S)$ of all subsets of set S (introduced in Example 2), the only initial object 0 is the empty set $\emptyset$ with arrow $0_A := \langle \emptyset, A\rangle$. The only final element 1 is the set S with arrow $1_A := \langle A, S\rangle$. □

Definition 17 (Product and Coproduct)**.** Let $\mathcal{C}$ be a category. Then the triple $\langle a \times b, \pi_1, \pi_2\rangle$ is a *product* of $\mathcal{C}$-objects a and b if $a \times b$ is a $\mathcal{C}$-object, the *product object*, with arrows $\pi_1\colon a \times b \to a$ and $\pi_2\colon a \times b \to b$, the *projections*, such that for every triple $\langle c, f, g\rangle$ with $\mathcal{C}$-object c and arrows $f\colon c \to a$ and $g\colon c \to b$ there exists exactly one arrow $\langle f, g\rangle\colon c \to a \times b$ such that the following diagram commutes:



Dually, the triple $(a + b, \iota_1, \iota_2)$ is a *coproduct* of $\mathcal{C}$-objects a and b if $a + b$ is a $\mathcal{C}$-object, the *coproduct* object, with arrows $\iota_1\colon a \to a + b$ and $\iota_2\colon b \to a + b$, the *injections*, such that for every triple $\langle c, f, g\rangle$ with $\mathcal{C}$-object c and arrows $f\colon a \to c$ and $g\colon b \to c$ there exists exactly one arrow $[f, g]\colon a + b \to c$ such that the following diagram commutes:



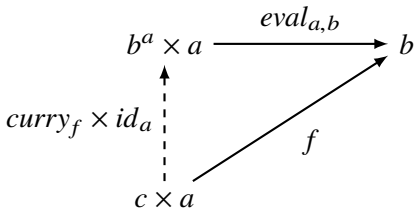
The product and the coproduct are thus defined by universal constructions analogous to those of the final and the initial element, respectively; thus also products and coproducts are uniquely defined up to isomorphism.

Example 4 (Product and Coproduct)**.** In the category *SET* of all sets (introduced in Example 1), a product object $A \times B$ is the set of all pairs $\{\langle a, b\rangle \mid a \in A \wedge b \in B\}$ with the projections determined by functions $\pi^\lambda_1(\langle a, b\rangle) := a$ and $\pi^\lambda_2(\langle a, b\rangle) := b$. A coproduct object $A + B$ is the disjoint union $\{\langle a, 1\rangle \mid a \in A\} \cup \{\langle b, 2\rangle \mid b \in B\}$ with the injections defined by the functions $\iota^\lambda_1(a) := \langle a, 1\rangle$ and $\iota^\lambda_2(b) := \langle b, 2\rangle$.

In the category $SUB(S)$ of all subsets of set S (introduced in Example 2), a product object $A \times B$ is the intersection $A \cap B$ with the existence of the projections $\pi_1 \colon A \cap B \to A$ and $\pi_2 \colon A \cap B \to A$ expressing the facts $A \cap B \subseteq A$ and $A \cap B \subseteq B$, respectively. A coproduct object $A + B$ is the union $A \cup B$ with the existence of the injections $\iota_1 \colon A \to A \cup B$ and $\iota_2 \colon B \to A \cup B$ expressing the facts $A \subseteq A \cup B$ and $B \subseteq A \cup B$, respectively. □

Definition 18 (Product Arrow)**.** Let $\mathcal{C}$ be a category with products $\langle a_1 \times a_2, \pi_1, \pi_2 \rangle$ and $\langle b_1 \times b_2, \pi_1', \pi_2' \rangle$ and arrows $f \colon a_1 \to b_1$ and $g \colon a_2 \to b_2$, respectively. Then the *product arrow* $f \times g \colon a_1 \times a_2 \to b_1 \times b_2$ is the arrow $\langle f \circ \pi_1, g \circ \pi_2 \rangle$.

Definition 19 (Exponential)**.** Let $\mathcal{C}$ be a category in which for all $\mathcal{C}$-objects there exists a product object. Then the tuple $\langle b^a, eval_{a,b} \rangle$ is an *exponential* of $\mathcal{C}$-objects a and b if b^a is a $\mathcal{C}$-object, the *exponential object*, with arrow $eval_{a,b} \colon b^a \times a \to b$, the *evaluation arrow*, such that for every $\mathcal{C}$-object c with arrow $f \colon c \times a \to b$ there exists exactly one arrow $curry_f \colon c \to b^a$, the *currying arrow*, such that the following diagram commutes:



Since also the exponential is defined by a universal construction, it is uniquely defined up to isomorphism.

Example 5 (Exponential)**.** In the category *SET* of all sets (introduced in Example 1), for *SET*-objects (i.e., sets) A, B, the exponential object B^A is the set $\{f \colon A \to B\}$ of all functions from A to B. The evaluation arrow is defined by the function $eval^{\lambda}_{A,B}(f, a) := f(a)$ for $f \colon A \to B$ and $a \in A$, the currying arrow by the function $curry^{\lambda}_f(c) := \lambda a.f(c, a)$ for $f \colon C \times A \to B$ and $c \in C$. Thus, given objects $c \in C, a \in A, b \in B$ and function $f \colon c \times a \to b$, we can construct a function $curry^{\lambda}_f(c)$ such that for arbitrary object a, we have $eval^{\lambda}_{A,B}(curry^{\lambda}_f(c), a) = f(c, a)$.

In the category $SUB(S)$ of all subsets of set S (introduced in Example 2), the exponential object B^A is the set $\overline{A} \cup B = (S \backslash A) \cup B$. The existence of the evaluation arrow $eval_{A,B} := \langle B^A \cap A, B \rangle$ expresses the fact $B^A \cap A \subseteq B$. The existence of the currying arrow defined as $curry_f := \langle C, B^A \rangle$ states $C \subseteq B^A$. Thus, given set C and $f \colon C \times A \to B$, i.e., a derivation goal $C \cap A \subseteq B$, we can construct $curry_f \colon C \to B^A$, i.e., the "missing assumption" $C \subseteq B^A$ which allows us to deduce $C \cap A \subseteq B$ using $eval_{A,B} : B^A \cap A \to B$, i.e., the "universal knowledge" $B^A \cap A \subseteq B$. Interpreting sets as propositions, B^A as the implication $A \to B$, the intersection $\cap$ as the conjunction $\wedge$, and the subset relation $\subseteq$ as the "entails" relation $\models$, above elaboration describes in a nutshell the inference rule

$$\frac{curry_f \colon \quad C \vdash A \to B}{f \colon \quad C \wedge A \vdash B}$$

where the "universal knowledge"

$$eval_{A,B} \colon (A \to B) \wedge A \vdash B$$

is essentially the *modus ponens* rule. □

Functors and Adjunction Moving on from individual categories, we will now discuss some concepts that address relationships between categories.

Definition 20 (Functor)**.** Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A *functor* $F\colon \mathcal{C} \to \mathcal{D}$ is a map that takes every $\mathcal{C}$-object a to a $\mathcal{D}$-object $F(a)$ and every $\mathcal{C}$-arrow $f\colon a \to b$ to a $\mathcal{D}$-arrow $F(f)\colon F(a) \to F(b)$ such that

- $F(id_a) = id_{F(a)}$ for every $\mathcal{C}$-object a, and
- $F(g \circ_C f) = F(g) \circ_D F(f)$ for all $\mathcal{C}$-arrows $f\colon a \to b$ and $g\colon b \to c$.

Example 6 (Functor)**.** We define the category I whose objects are the integer numbers, whose arrows denote the "less than equal" relation $\leq_{\mathbb{Z}}$ over the integers (i.e., a unique arrow $f\colon x \to y$ exists if and only if $x \leq_{\mathbb{Z}} y$) and whose composition denotes the composition of this relation. Analogously we introduce the category $\mathcal{R}$ whose objects are the real numbers, whose arrows denote the "less than equal" relation $\leq_{\mathbb{R}}$ over the reals, and whose composition operation denotes relational composition.

Consider a functor $R\colon I \to \mathcal{R}$ that maps the integers into the reals such that for every I-object i (i.e., an integer number) $R(i)$ represents an $\mathcal{R}$-object, (i.e., a real number) and every I-arrow $i_1 \to i_2$ is mapped to an $\mathcal{R}$-arrow $R(i_1) \to R(i_2)$ (i.e., if $i_1 \leq_{\mathbb{I}} i_2$, then $R(i_1) \leq_{\mathbb{R}} R(i_2)$). Thus R may be any monotonous map from the integers to the reals. □

Definition 21 (Adjunction, Left and Right Adjoint)**.** Let $\mathcal{C}$ and $\mathcal{D}$ be categories with functors $F\colon \mathcal{C} \to \mathcal{D}$ and $G\colon \mathcal{D} \to \mathcal{C}$. Then we have $F \dashv G$ (read: $\langle F, G\rangle$ is an *adjunction*, F is a *left adjoint* of G, G is a *right adjoint* of F) if for every $\mathcal{C}$-object a and $\mathcal{D}$-object b the arrow classes $\mathcal{D}(F(a), b)$ and $\mathcal{C}(a, G(b))$ are isomorphic, i.e., there exists a bijection between them. This is equivalent[1] to saying that for every $\mathcal{C}$-object a and $\mathcal{D}$-object b there exist two surjective mappings $s_1\colon \mathcal{D}(F(a), b) \to \mathcal{C}(a, G(b))$ and $s_2\colon \mathcal{C}(a, G(b)) \to \mathcal{D}(F(a), b)$, i.e.,

- for every $\mathcal{D}$-arrow $g\colon F(a) \to b$ we have a $\mathcal{C}$-arrow $f\colon a \to G(b)$ with $s_2(f) = g$ and
- for every $\mathcal{C}$-arrow $f\colon a \to G(b)$ twe have a $\mathcal{D}$-arrow $g\colon F(a) \to b$ with $s_1(g) = f$.

Example 7. We introduce between the set $\mathbb{Z}$ of integer numbers respectively the set $\mathbb{R}$ of real numbers the functions

$$
\begin{aligned}
&\mathit{real}\colon \mathbb{Z} \to \mathbb{R},\ \mathit{real}(x) := x\\
&\mathit{ceil}\colon \mathbb{R} \to \mathbb{Z},\ \mathit{ceil}(x) := \min\{y \in \mathbb{Z} \mid x \leq_{\mathbb{R}} \mathit{real}(y)\}\\
&\mathit{floor}\colon \mathbb{R} \to \mathbb{Z},\ \mathit{floor}(x) := \max\{y \in \mathbb{Z} \mid \mathit{real}(y) \leq_{\mathbb{R}} x\}
\end{aligned}
$$

[1] This equivalence is a consequence of the Cantor–Schröder–Bernstein theorem which states that there exists a bijective function between sets A and B if there exist injective functions $f\colon A \to B$ and $g\colon B \to A$. This implies that such a bijective function also exists if there exist surjective functions $f'\colon A \to B$ and $g'\colon B \to A$ because from these we can define the injective functions $f(a) :=$ such b. $g'(b) = a$ and $g(b) :=$ such a. $f'(a) = b$. While the theorem has been formulated for sets, it can also be generalized to classes.

where $real(x)$ denotes the "embedding" of integer x to real, $ceil(x)$ denotes the least integer greater equal x, and $floor(x)$ denotes the greatest integer less equal x.

Now consider the categories I and $\mathcal{R}$ introduced in Example 6. Above functions induce corresponding functors over these categories, the "embedding functor" $R\colon I \to \mathcal{R}$, the "ceiling functor" $C\colon \mathcal{R} \to I$, and the "floor functor" $F\colon \mathcal{R} \to I$, where for every $\mathcal{R}$-object x we have $R(x) = real(x)$ and for every I-object x we have $C(x) = ceil(x)$ and $F(x) = floor(x)$. Since function *real* is monotonic, R maps I-arrows, i.e., inequalities on $\mathbb{Z}$, to $\mathcal{R}$-arrows, i.e., inequalities on $\mathbb{R}$; thus R is indeed a functor. Likewise, since also functions *ceil* and *floor* are monotonic, also C and I are indeed functors.

Above functions satisfy the following properties (we will prove this claim below):

$$\forall x \in \mathbb{R}, y \in \mathbb{Z}.\ ceil(x) \leq_{\mathbb{Z}} y \Leftrightarrow x \leq_{\mathbb{R}} real(y) \tag{1}$$

$$\forall x \in \mathbb{Z}, y \in \mathbb{R}.\ x \leq_{\mathbb{Z}} floor(y) \Leftrightarrow real(x) \leq_{\mathbb{R}} y \tag{2}$$

As a consequence of property (1), we have for every I-arrow $f\colon C(x) \to y$ (with $\mathcal{R}$-object x and I-object y) exactly one $\mathcal{R}$-arrow $g\colon x \to R(y)$ and vice versa, i.e., the arrow classes $I(C(x), y)$ and $\mathcal{R}(x, R(y))$ are isomorphic. As a consequence of property (2), we have for every I-arrow $f\colon x \to F(y)$ (with I-object x and $\mathcal{R}$-object y) exactly one $\mathcal{R}$-arrow $g\colon R(x) \to y$ and vice versa, i.e., the arrow classes $\mathcal{R}(R(x), y)$ and $I(x, F(x))$ are isomorphic. Thus we have

$$C \dashv R \wedge R \dashv F$$

i.e., the ceiling functor is a left adjoint of the embedding functor and the floor functor is a right adjoint, respectively.

It remains to prove (1) and (2). The definitions of functions *ceil* and *floor* immediately give us the following properties:

$$ceil(x) = y \Leftrightarrow \left(x \leq_{\mathbb{R}} real(y) \wedge \forall y' \in \mathbb{Z} : x \leq_{\mathbb{R}} real(y') \Rightarrow y \leq_{\mathbb{Z}} y'\right) \tag{3}$$

$$floor(x) = y \Leftrightarrow \left(real(y) \leq_{\mathbb{R}} x \wedge \forall y' \in \mathbb{Z} : real(y') \leq_{\mathbb{R}} x \Rightarrow y' \leq_{\mathbb{Z}} y\right) \tag{4}$$

Furthermore we clearly have the following properties:

$$\forall x_1, x_2, x_3 \in \mathbb{R}.\ x_1 \leq_{\mathbb{R}} x_2 \wedge x_2 \leq_{\mathbb{R}} x_3 \Rightarrow x_1 \leq_{\mathbb{R}} x_3 \tag{5}$$

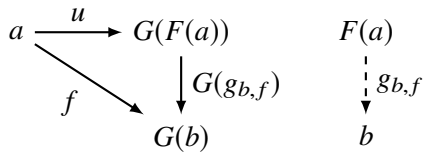
$$\forall x, x' \in \mathbb{Z}.\ x \leq_{\mathbb{Z}} x' \Rightarrow real(x) \leq_{\mathbb{R}} real(x') \tag{6}$$

To prove (1), we take arbitrary $x \in \mathbb{R}$ and $y \in \mathbb{Z}$ and show $ceil(x) \leq_{\mathbb{Z}} y \Leftrightarrow x \leq_{\mathbb{R}} real(y)$. First we assume $ceil(x) \leq_{\mathbb{Z}} y$ (7) and show $x \leq_{\mathbb{R}} real(y)$. From (3) and (7), we know $x \leq_{\mathbb{R}} real(ceil(x))$; from (6) and (7), we know $real(ceil(x)) \leq_{\mathbb{R}} real(y)$; from (5) we thus have the goal. Next we assume $x \leq_{\mathbb{R}} real(y)$ (7) and show $ceil(x) \leq_{\mathbb{Z}} y$ which follows directly from (3) and (7).

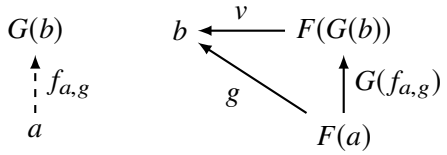
To prove (2), we take arbitrary $x \in \mathbb{R}$ and $y \in \mathbb{Z}$. and show $x \leq_{\mathbb{Z}} floor(y) \Leftrightarrow real(x) \leq_{\mathbb{R}} y$. First we assume $x \leq_{\mathbb{Z}} floor(y)$ (7) and show $real(x) \leq_{\mathbb{R}} y$. From (6) and (7), we know $real(x) \leq_{\mathbb{R}} real(floor(y))$; from (4), we know $real(floor(y)) \leq_{\mathbb{R}} y$; from (5), we have the goal. Next we assume $real(x) \leq_{\mathbb{R}} y$ (7) and show $x \leq_{\mathbb{Z}} floor(y)$ which follows from (4) and (7). □

Proposition 7 (Equivalence of Adjunctions and Universals)**.** *Let $\mathcal{C}$ and $\mathcal{D}$ be categories with functors $F\colon \mathcal{C} \to \mathcal{D}$ and $G\colon \mathcal{D} \to \mathcal{C}$. Then the condition $F \dashv G$ is equivalent to each of the following two conditions:*

1. *For every $\mathcal{C}$-object a there is a $\mathcal{C}$-arrow $u\colon a \to G(F(a))$, the "universal arrow", such that for every $\mathcal{D}$-object b and $\mathcal{C}$-arrow $f\colon a \to G(b)$ there exists a $\mathcal{D}$-arrow $g_{b,f}\colon F(a) \to b$:*



2. *For every $\mathcal{D}$-object b there is a $\mathcal{C}$-arrow $v\colon F(G(b)) \to b$, the "couniversal arrow", such that for every $\mathcal{C}$-object a and $\mathcal{D}$-arrow $g\colon F(a) \to b$ there is a $\mathcal{C}$-arrow $f_{a,g}\colon a \to G(b)$:*



Proof. See the proof of Propositions 6 and 7 in [2]. □

Example 8. Consider the categories $\mathcal{R}$ and I with functors $C\colon \mathcal{R} \to I$, $R\colon I \to \mathcal{R}$, and $F\colon \mathcal{R} \to I$ introduced in Example 7. For these we have $\mathcal{C} \dashv \mathcal{R}$ and $\mathcal{R} \dashv \mathcal{F}$.

We consider $\mathcal{C} \dashv \mathcal{R}$ and apply the first part of Proposition 7:

- First, for every $\mathcal{R}$-object a, i.e. $a \in \mathbb{R}$, we have $u\colon a \to R(C(a))$, i.e., $a \leq_{\mathbb{R}} \mathit{real}(\mathit{ceil}(a))$.
- Second, for every I-object b and $\mathcal{R}$-arrow $f\colon a \to R(b)$, i.e., $b \in \mathbb{Z}$ with $a \leq_{\mathbb{Z}} \mathit{real}(b)$, we have an I-arrow $g_{b,f}\colon C(a) \to b$, i.e., $\mathit{ceil}(a) \leq_{\mathbb{Z}} b$.

In other words, the following proposition holds:

$$a \leq_{\mathbb{R}} \mathit{real}(\mathit{ceil}(a)) \wedge \forall b \in \mathbb{Z}.\ a \leq_{\mathbb{Z}} \mathit{real}(b) \Rightarrow \mathit{ceil}(a) \leq_{\mathbb{Z}} b$$

which is equivalent to property (3) in Example 7 and can be stated more succinctly as

$$\mathit{ceil}(a) = \min\{b \in \mathbb{Z} \mid a \leq_{\mathbb{R}} \mathit{real}(b)\}$$

which is the defining equation of *ceil*.

Now we consider $\mathcal{R} \dashv \mathcal{F}$ and apply the second part of Proposition 7:

- First, for every $\mathcal{R}$-object b, i.e. $b \in \mathbb{R}$, we have $v\colon R(F(b)) \to b$, i.e., $\mathit{real}(\mathit{floor}(b)) \leq_{\mathbb{R}} b$.
- Second, for every I-object a and $\mathcal{R}$-arrow $g\colon R(a) \to b$, i.e., $a \in \mathbb{Z}$ with $\mathit{real}(a) \leq_{\mathbb{Z}} b$, we have an $\mathcal{R}$-arrow $f_{a,g}\colon b \to F(a)$, i.e., $b \leq_{\mathbb{Z}} \mathit{floor}(a)$.

In other words, the following proposition holds:

$$real(floor(b)) \leq_{\mathbb{R}} b \wedge \forall a \in \mathbb{Z}.\ real(a) \leq_{\mathbb{Z}} b \Rightarrow b \leq_{\mathbb{Z}} floor(a)$$

which is equivalent to property (4) in Example 7 and can be stated more succinctly as

$$floor(b) = \max\{a \in \mathbb{Z} \mid real(a) \leq_{\mathbb{R}} b\}$$

which is the defining equation of *floor*. □

As Example 8 demonstrates, Proposition 7 states in a nutshell that a left adjoint uniquely defines a function by taking as its result the "least" element for which the application of the right adjoint does not yield a value that is "less" than the argument; dually a right adjoint uniquely defines a function by taking as its result the "greatest" element for which the application of the left adjoint does not yield a value that is "greater" than the argument.

Object Constructions by Adjunction We conclude this section by demonstrating that the previously described object conjunctions can be also considered as applications of functors that are determined as left respectively right adjoints to certain basic functors.

Proposition 8 (Initial and Final Object by Adjunction)**.** *Let* $\mathbf{1}$ *be the "singleton" category with a single object* $*$ *(and consequently a single arrow* $id_*: * \to *$*); this category is uniquely defined up to isomorphism. Let* $\mathcal{C}$ *be a category with the constant functor* $C: \mathcal{C} \to \mathbf{1}$*; also this functor is uniquely defined up to isomorphism. Then the following holds:*

- *Let* $\mathcal{C}$*-object* 0 *be initial and the "initial object functor"* $I_0: \mathbf{1} \to \mathcal{C}$ *be defined by* $I_0(*) := 0$ *and* $I_0(id_*) := id_0$*. Then we have* $I_0 \dashv C$*, i.e., the initial object functor is a left adjoint of the constant functor.*
- *Let* $\mathcal{C}$*-object* 1 *be final and the "final object functor"* $F_1: \mathbf{1} \to \mathcal{C}$ *be defined by* $F_1(*) := 1$ *and* $F_1(id_*) := id_1$*. Then we have* $C \dashv F_1$*, i.e., the final object functor is a right adjoint of the constant functor.*

Proof. For showing the first statement, we take the initial object 0 with initial object functor I_0. We show $I_0 \dashv C$, i.e., that $\mathcal{C}(I_0(*), a)$ and $\mathbf{1}(*, C(a))$ are isomorphic, for arbitrary $\mathcal{C}$-object a. This follows from $\mathbf{1}(*, C(a)) = \mathbf{1}(*, *)$, $\mathcal{C}(I_0(*), a) = \mathcal{C}(0, a)$, and the fact that there exists exactly one $\mathbf{1}$-arrow $id_*: * \to *$ and, since 0 is initial, exactly one $\mathcal{C}$-arrow $f: 0 \to a$.

For showing the second statement, we take the final object 1 with final functor F_1. We show $C \dashv F_1$, i.e., that $\mathbf{1}(C(a), *)$ and $\mathcal{C}(a, F_1(*))$ are isomorphic, for arbitrary $\mathcal{C}$-object a. This follows from $\mathbf{1}(C(a), *) = \mathbf{1}(*, *)$, $\mathcal{C}(a, F_1(*)) = \mathcal{C}(a, 1)$, and the fact that there exists exactly one $\mathbf{1}$-arrow $id_*: * \to *$ and, since 1 is final, exactly one $\mathcal{C}$-arrow $f: a \to 1$. □

Proposition 9 (Product and Coproduct by Adjunction)**.** *Let* $\mathcal{C}$ *be a category. Let the "product category"* $\mathcal{C} \times \mathcal{C}$ *be the category whose objects* (a, b) *are pairs of* $\mathcal{C}$*-objects* a *and* b*, whose arrows* $(f, g) : (a, c) \to (b, d)$ *are pairs of* $\mathcal{C}$*-arrows* $f: a \to b$ *and* $g: c \to d$*, where the identity arrows are pairs of identity arrows, and where composition is component-wise composition. Let the "diagonal functor"* $\Delta: \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ *be defined by* $\Delta(a) = (a, a)$ *for every* $\mathcal{C}$*-object* a *and* $\Delta(f) = (f, f)$ *for every* $\mathcal{C}$*-arrow* $f: a \to b$*. Then the following holds:*

- *Assume that every pair of $\mathcal{C}$-objects a and b has a product $a \times b$ and let the "product functor" $P\colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ be defined by $P(a,b) := a \times b$. Then we have $\Delta \dashv P$, i.e., the product functor is a right adjoint of the diagonal functor.*

- *Assume that every pair of $\mathcal{C}$-objects a and b has a coproduct $a + b$ and let the "coproduct functor" $C\colon \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ be defined by $C(a,b) := a + b$. Then we have $C \dashv \Delta$, i.e., the coproduct functor is a left adjoint of the diagonal functor.*

Proof. For showing the first statement, we take arbitrary category $\mathcal{C}$ and functor P satisfying the stated assumption. We show $\Delta \dashv P$, i.e., that for arbitrary $\mathcal{C}$-objects p, a, b the arrow classes $(\mathcal{C} \times \mathcal{C})(\Delta(p), (a,b))$ and $\mathcal{C}(p, P(a,b))$ are isomorphic. Since $\Delta(p) = (p,p)$ and $P(a,b) = a \times b$, it suffices to find surjections $s_1\colon (\mathcal{C} \times \mathcal{C})((p,p),(a,b)) \to \mathcal{C}(p, a \times b)$ and $s_2\colon \mathcal{C}(p, a \times b) \to (\mathcal{C} \times \mathcal{C})((p,p),(a,b))$. First, we define $s_1(f,g) := \langle f, g\rangle$ where $\langle f, g\rangle\colon p \to a \times b$ is the unique $\mathcal{C}$-arrow given to us by Definition 17 with property $f = \pi_1 \circ \langle f, g\rangle$ and $g = \pi_2 \circ \langle f, g\rangle$. Now we show that for every $\mathcal{C}$-arrow $h\colon p \to a \times b$ there exist some $\mathcal{C}$-arrows $f\colon p \to a$ and $g\colon p \to b$ with $s_1(f,g) = h$. We take $f := \pi_1 \circ h$ and $g := \pi_2 \circ h$. Due to the uniqueness of $\langle f, g\rangle$, the equalities $f = \pi_1 \circ h$ and $g = \pi_2 \circ h$ imply $h = \langle f, g\rangle$ and thus $s_1(f,g) = h$. Second, we define $s_2(h) := (\pi_1 \circ h, \pi_2 \circ h)$. Now we show that for every $(\mathcal{C} \times \mathcal{C})$-arrow $(f,g)\colon (p,p) \to (a,b)$, i.e., for all $\mathcal{C}$-arrows $f\colon p \to a$ and $g\colon p \to b$, there exists some $\mathcal{C}$-arrow $h\colon p \to a \times b$ with $s_2(h) = (f,g)$, i.e., $\pi_1 \circ h = f$ and $\pi_2 \circ h = g$. Definition 17 gives us this h.

For showing the second statement, we take arbitrary category $\mathcal{C}$ and functor C satisfying the stated assumption. We show $C \dashv \Delta$, i.e., that for arbitrary $\mathcal{C}$-objects a, b, c the arrow classes $\mathcal{C}(C(a,b), c)$ and $(\mathcal{C} \times \mathcal{C})((a,b), \Delta(c))$ are isomorphic. Since $C(a,b) = a + b$ and $\Delta(c) = (c,c)$, it suffices to find surjections $s_1\colon \mathcal{C}(a+b, c) \to (\mathcal{C}\times\mathcal{C})((a,b),(c,c))$ and $s_2\colon (\mathcal{C}\times\mathcal{C})((a,b),(c,c)) \to \mathcal{C}(a+b, c)$. First, we define $s_1(h) := (h \circ \iota_1, h \circ \iota_2)$. Now we show that for every $(\mathcal{C} \times \mathcal{C})$-arrow $(f,g)\colon (a,b) \to (c,c)$, i.e., for all $\mathcal{C}$-arrows $f\colon a \to c$ and $g\colon b \to c$, there exists some $\mathcal{C}$-arrow $h\colon a + b \to c$ with $s_1(h) = (f,g)$, i.e., $h \circ \iota_1 = f$ and $h \circ \iota_2 = g$. Definition 17 gives us this h. Second, we define $s_2(f,g) := [f,g]$ where $[f,g]\colon a + b \to c$ is the unique $\mathcal{C}$-arrow given to us by Definition 17 with property $f = [f,g] \circ \iota_1$ and $g = [f,g] \circ \iota_2$. Now we show that for every $\mathcal{C}$-arrow $h\colon a + b \to c$ there exist some $\mathcal{C}$-arrows $f\colon a \to c$ and $g\colon b \to c$ with $s_2(f,g) = h$. We take $f := h \circ \iota_1$ and $g := h \circ \iota_2$. Due to the uniqueness of $[f,g]$, the equalities $f = h \circ \iota_1$ and $g = h \circ \iota_2$ imply $h = [f,g]$ and thus $s_2(f,g) = h$. □

Proposition 10 (Exponential by Adjunction)**.** *Let $\mathcal{C}$ be a category in which for every pair of $\mathcal{C}$-objects a and b there exists a product object $b \times a$ and an exponential object b^a. For every $\mathcal{C}$-object a, let the "(unary) product functor" $P_a\colon \mathcal{C} \to \mathcal{C}$ be defined by $P_a(b) := b \times a$ and the "(unary) exponential functor" $E_a\colon \mathcal{C} \to \mathcal{C}$ be defined by $E_a(b) := b^a$. Then we have $P_a \dashv E_a$, i.e., the exponential functor is a right adjoint of the product functor.*

Proof. We take arbitrary category $\mathcal{C}$, $\mathcal{C}$-object a, and functors P_a and E_a satisfying the assumption. We show $P_a \dashv E_a$, i.e., that for arbitrary $\mathcal{C}$-objects b, c the arrow classes $\mathcal{C}(P_a(c), b)$ and $\mathcal{C}(c, E_a(b))$ are isomorphic. Since $P_a(b) = b \times a$ and $E_a(b) = b^a$ it suffices to find surjections $s_1\colon \mathcal{C}(c \times a, b) \to \mathcal{C}(c, b^a)$ and $s_2\colon \mathcal{C}(c, b^a) \to \mathcal{C}(c \times a, b)$. First, we define $s_1(f) := curry_f$. Now we show that for every $\mathcal{C}$-arrow $g\colon c \to b^a$ there exists some $\mathcal{C}$-arrow $f\colon c \times a \to b$ with $s_1(f) = g$, i.e., $curry_f = g$. We define $f := eval_{a,b} \circ (g \times id_a)$ and show $curry_f = g$. From the

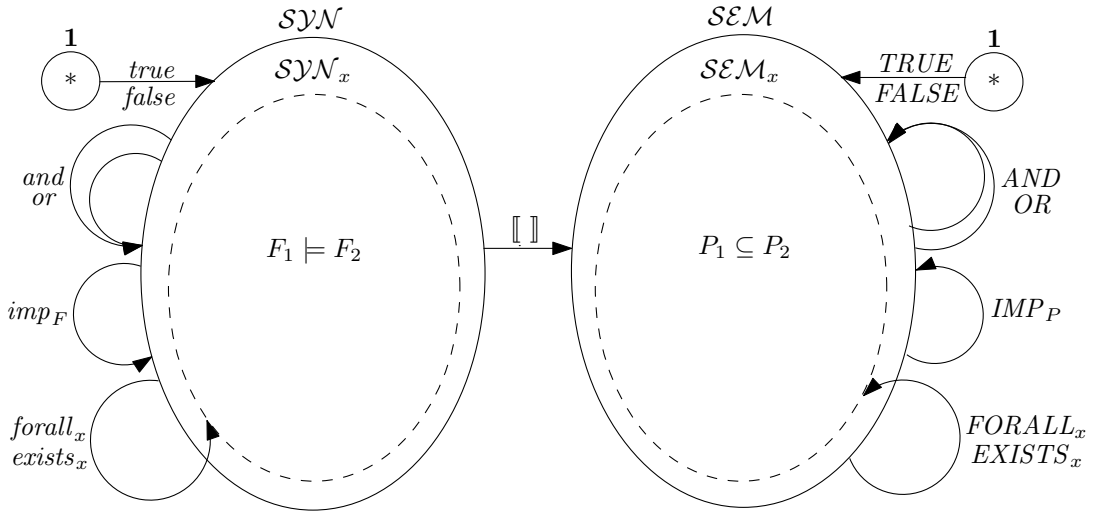


Figure 1: A Categorical Semantics of Relational First-Order Logic

definition of f, we know that the $\mathcal{C}$-arrow $g\colon c \to b^a$ satisfies the equality $f = eval_{a,b} \circ (g \times id_a)$. However, Definition 19 implies that the only such $\mathcal{C}$-arrow is $curry_f$; thus $curry_f = g$. Second, we define $s_2(g) := eval_{a,b} \circ (g \times id_a)$. Now we show that for every $\mathcal{C}$-arrow $f\colon c \times a \to b$ there exists some $\mathcal{C}$-arrow $g\colon c \to b^a$ with $s_2(g) = f$, i.e., $eval_{a,b} \circ (g \times id_a) = f$. We define $g := curry_f$ from which Definition 19 proves the goal. □

We are now ready to discuss the central aspects of categorical logic.

4 A Categorical Semantics

Based on the concepts introduced in the previous sections, this section elaborates a categorical semantics of our relational version of first order logic. We advise the reader to consult Figure 1 to grasp the overall framework and the relationship between its various categories and functors.

Syntactic Category and Formula Functors We start by introducing the "syntactic category" $\mathcal{SYN} = \langle For, A, \circ \rangle$ as follows:

- The objects of this category are the formulas in the set *For* which was introduced in Definition 6.
- The arrow class A consists of all pairs $\langle F_1, F_2 \rangle$ of formulas F_1, F_2 for which $F_1 \models F_2$ holds, i.e., for which F_2 is a logical consequence of F_1, as described in Definition 8. The source object of such an arrow is F_1, its target object is F_2. The existence of an arrow $f\colon F_1 \to F_2$ thus indicates $F_1 \models F_2$. The identity $id_F\colon F \to F$ indicates the fact $F \models F$.
- The composition $\circ$ denotes relational composition: for all arrows $f\colon F_1 \to F_2$ and $g\colon F_2 \to F_3$, the existence of the arrow $(g \circ f)\colon F_1 \to F_3$ indicates the transitivity of the relation $\models$.

For every variable x, $\mathcal{SYN}_x$ is that subcategory of $\mathcal{SYN}$ whose objects are formulas whose semantics is independent of x (see Definition 4).

For reasons explained below, we will exclude from the syntactic category negations and equivalences, i.e., formulas of form $(\neg F)$ and $(F_1 \leftrightarrow F_2)$. We may do so by considering them as the following syntactic shortcuts:

$$(\neg F) \equiv (F \to \bot)$$
$$(F_1 \leftrightarrow F_2) \equiv (F_1 \to F_2) \wedge (F_2 \to F_1)$$

The validity of these shortcuts can be easily shown by proving the corresponding logical equivalences. Consequently negations and conjunctions need subsequently not be considered any more and their semantics need not be explicitly defined.

For the other kinds of formulas, we introduce the following (families of) "formula functors" where $\mathbf{1}$ is the "singleton" category with a single object $*$ (see Proposition 8):

$$\begin{aligned}
\mathit{true} &: \mathbf{1} \to \mathcal{SYN} \\
\mathit{false} &: \mathbf{1} \to \mathcal{SYN} \\
\mathit{and} &: \mathcal{SYN} \times \mathcal{SYN} \to \mathcal{SYN} \\
\mathit{or} &: \mathcal{SYN} \times \mathcal{SYN} \to \mathcal{SYN} \\
\mathit{imp}_{F \in For} &: \mathcal{SYN} \to \mathcal{SYN} \\
\mathit{forall}_{x \in Var} &: \mathcal{SYN} \to \mathcal{SYN}_x \\
\mathit{exists}_{x \in Var} &: \mathcal{SYN} \to \mathcal{SYN}_x
\end{aligned}$$

These functors map formulas to formulas, and logical consequences to logical consequences. The formula mappings are naturally defined as follows:

$$\begin{aligned}
\mathit{true}(*) &:= \top \\
\mathit{false}(*) &:= \bot \\
\mathit{and}(F_1, F_2) &:= F_1 \wedge F_2 \\
\mathit{or}(F_1, F_2) &:= F_1 \vee F_2 \\
\mathit{imp}_{F_1}(F_2) &:= F_1 \to F_2 \\
\mathit{forall}_x(F) &:= \forall x.\ F \\
\mathit{exists}_x(F) &:= \exists x.\ F
\end{aligned}$$

As for the mapping of consequences, we notice that all functors are *covariant* in their $\mathcal{SYN}$ arguments[2], i.e., we have for all formulas F, F_1, F_2, G, G_1, G_2 and every variable x the following

[2]It is exactly for this reason that negation and equivalence (which do not enjoy covariance in their arguments) are not modeled as formula functors and that implication (which is only covariant in its second argument) is not modeled by a binary functor but by a family of unary functors.

(easy to prove) properties:

$$(F_1 \models G_1) \wedge (F_2 \models G_2) \Rightarrow and(F_1, F_2) \models and(G_1, G_2)$$
$$(F_1 \models G_1) \wedge (F_2 \models G_2) \Rightarrow or(F_1, F_2) \models or(G_1, G_2)$$
$$(F_2 \models G_2) \Rightarrow imp_{F_1}(F_2) \models imp_{F_1}(G_2)$$
$$(F \models G) \Rightarrow forall_x(F) \models forall_x(G)$$
$$(F \models G) \Rightarrow exists_x(F) \models exists_x(G)$$

Therefore the object maps of these functors naturally induce the necessary logical consequences.

Semantic Category and Predicate Functors Next we introduce the "semantic category" $\mathcal{SEM} = \langle Pred, B, \circ \rangle$ as follows:

- The objects of this category are the predicates in the set *Pred* which was introduced in Definition 5 (thus $\mathcal{SEM}$-objects are relations, i.e., sets).
- The arrow class B consists of all pairs $\langle P_1, P_2 \rangle$ of predicates P_1, P_2 for which $P_1 \subseteq P_2$ holds, i.e., for which P_1 is a subset of P_2. The source object of such an arrow is P_1, its target object is P_2. The existence of an arrow $f : P_1 \to P_2$ thus indicates $P_1 \subseteq P_2$. The identity $id_P : P \to P$ indicates the fact $P \subseteq P$.
- The composition $\circ$ denotes relational composition: for all arrows $f : P_1 \to P_2$ and $g : P_2 \to P_3$, the existence of the arrow $(g \circ f) : P_1 \to P_3$ indicates the transitivity of the relation $\subseteq$.

For every variable x, $\mathcal{SEM}_x$ is that subcategory of $\mathcal{SEM}$ whose objects are predicates that are independent of x (see Definition 4).

Corresponding to the various kinds of formula constructions, we will have the following "predicate functors" (respectively families of functors):

$$\begin{aligned}
TRUE &: \mathbf{1} \to \mathcal{SEM} \\
FALSE &: \mathbf{1} \to \mathcal{SEM} \\
AND &: \mathcal{SEM} \times \mathcal{SEM} \to \mathcal{SEM} \\
OR &: \mathcal{SEM} \times \mathcal{SEM} \to \mathcal{SEM} \\
IMP_{P \in Pred} &: \mathcal{SEM} \to \mathcal{SEM} \\
FORALL_{x \in Var} &: \mathcal{SEM} \to \mathcal{SEM}_x \\
EXISTS_{x \in Var} &: \mathcal{SEM} \to \mathcal{SEM}_x
\end{aligned}$$

These functors map predicates to predicates and subset relations to subset relations (their detailed definitions will be given later). As we will see, these functors are covariant in their $\mathcal{SEM}$-arguments, i.e., we have for all predicates P, P_1, P_2, Q, Q_1, Q_2 and every variable x the following

properties:

$$(P_1 \subseteq Q_1) \wedge (P_2 \subseteq Q_2) \Rightarrow AND(P_1, P_2) \subseteq AND(Q_1, Q_2)$$
$$(P_1 \subseteq Q_1) \wedge (P_2 \subseteq Q_2) \Rightarrow OR(P_1, P_2) \subseteq OR(Q_1, Q_2)$$
$$(P_2 \subseteq Q_2) \Rightarrow IMP_{P_1}(P_2) \subseteq IMP_{P_1}(Q_2)$$
$$(P \subseteq Q) \Rightarrow FORALL_x(P) \subseteq FORALL_x(Q)$$
$$(P \subseteq Q) \Rightarrow EXISTS_x(P) \subseteq EXISTS_x(Q)$$

Therefore the object maps of these functors (defined by the respective predicate operations) naturally induce appropriate arrow maps (the corresponding subset relations).

The Semantic Functor Now we introduce the "semantic functor" $[\![\ ]\!] \colon \mathcal{SYN} \to \mathcal{SEM}$ defined as follows:

- For every $\mathcal{SYN}$-object F, i.e., formula F, $[\![F]\!]$ denotes the semantics of F as defined in Definition 7, which according to Proposition 3 is a predicate, i.e., indeed a $\mathcal{SEM}$-object.
- For every $\mathcal{SYN}$-arrow $f \colon F_1 \to F_2$, i.e., every pair of formulas F_1 and F_2 with $F_1 \models F_2$, we have the $\mathcal{SEM}$-arrow $[\![f]\!] \colon [\![F_1]\!] \to [\![F_2]\!]$, i.e., the fact $[\![F_1]\!] \subseteq [\![F_2]\!]$, which is a direct consequence of Definition 10 which introduces the $\models$ relation.

This semantic functor establishes the relationship between the previously introduced formula functors and predicate functors by the following identities on $\mathcal{SEM}$-objects, i.e., predicate identities, that will hold for all formulas F, F_1, F_2 and every variable x:

$$[\![true(*)]\!] = TRUE(*)$$
$$[\![false(*)]\!] = FALSE(*)$$
$$[\![and(F_1, F_2)]\!] = AND([\![F_1]\!], [\![F_2]\!])$$
$$[\![or(F_1, F_2)]\!] = OR([\![F_1]\!], [\![F_2]\!])$$
$$[\![imp_{F_1}(F_2)]\!] = IMP_{[\![F_1]\!]}([\![F_2]\!])$$
$$[\![forall_x(F)]\!] = FORALL_x([\![F]\!])$$
$$[\![exists_x(F)]\!] = EXISTS_x([\![F]\!])$$

We are now going to elaborate in detail the semantic functors from which all of above can be shown; this elaboration is inspired from and indeed directly derived from the well-known logical inference rules of first order logic. The resulting definitions are based on the categorical notions introduced in Section 3, i.e., final and initial objects, products and coproducts, exponentials, and left and right adjoints, respectively. This gives us for every logical operation a "universal" definition of its semantics. Nevertheless this semantics is also "constructive" in the sense that it is explicitly defined from well-known set-theoretic operations.

Logical Constants The role of the logical constants in reasoning is exhibited by the following two "rules" which follow directly from Definition 8 (these rules are propositions that are valid for every formula F; they mimic the corresponding inference rules of first order logic):

$$F \models \top \qquad \bot \models F$$

In other words, $\top$ is a logical consequence of every formula F, i.e., $\top$ is the "weakest" formula. Dually, every formula F is a logical consequence of $\bot$, i.e., $\bot$ is the "strongest" formula. This implies that $\mathit{true}(*) = \top$ is the final object of category $\mathcal{SYN}$ and $\mathit{false}(*) = \bot$ is its initial one (see Definition 16). Then Proposition 8 implies $C_{\mathcal{SYN}} \dashv \mathit{true}$ and $\mathit{false} \dashv C_{\mathcal{SYN}}$, i.e., functor true is the right adjoint of the constant functor $C_{\mathcal{SYN}} \colon \mathcal{SYN} \to \mathbf{1}$ while functor false is its left one.

Correspondingly $\mathit{TRUE}(*)$ is the final object of category $\mathcal{SEM}$ (the "weakest" predicate, i.e., the predicate which is a superset of every predicate) and $\mathit{FALSE}(*)$ is its initial object (the "strongest" predicate, i.e., the predicate which is a subset of every predicate). By Proposition 8 we then have $C_{\mathcal{SEM}} \dashv \mathit{TRUE}$ and $\mathit{FALSE} \dashv C_{\mathcal{SEM}}$, i.e., functor TRUE is the right adjoint of the constant functor $C_{\mathcal{SEM}} \colon \mathcal{SEM} \to \mathbf{1}$ while functor FALSE is its left one.

Therefore, corresponding to above rules for formulas, we have the following rules for every predicate P:

$$P \subseteq \mathit{TRUE}(*) \qquad \mathit{FALSE}(*) \subseteq P$$

Since final and initial objects are unique, these rules actually represent implicit but unique definitions of $\mathit{TRUE}(*)$ and $\mathit{FALSE}(*)$ which can be explicitly written as

$$\mathit{TRUE}(*) := \bigcup \{P \mid P \in \mathit{Pred}\} \qquad \mathit{FALSE}(*) := \bigcap \{P \mid P \in \mathit{Pred}\}$$

i.e., $\mathit{TRUE}(*)$ is the union of all predicates and $\mathit{FALSE}(*)$ is their intersection. Thus we have derived alternative characterizations $[\![\top]\!] = \mathit{TRUE}(*)$ and $[\![\bot]\!] = \mathit{FALSE}(*)$ that are both constructive and universal (Property 5 gives us $[\![\top]\!] = \mathit{Ass}$ and $[\![\bot]\!] = \emptyset$ from which it is easy to verify these equalities).

Conjunction and Disjunction The role of conjunction in reasoning is exhibited by the following rules for arbitrary formulas F_1, F_2, F (the first two ones mimic the logical inference rules of "elimination", the last one mimics the inference rule of "introduction"):

$$F_1 \wedge F_2 \models F_1$$
$$F_1 \wedge F_2 \models F_2$$
$$(F \models F_1) \wedge (F \models F_2) \Rightarrow (F \models F_1 \wedge F_2)$$

Dually we have the following rules for disjunction:

$$F_1 \models F_1 \vee F_2$$
$$F_2 \models F_1 \vee F_2$$
$$(F_1 \models F) \wedge (F_2 \models F) \Rightarrow (F_1 \vee F_2 \models F)$$

These rules (whose soundness can be established with the help of Definition 8) state that $(F_1 \wedge F_2)$ is the "weakest" formula F for which both $(F \models F_1)$ and $(F \models F_2)$ hold and that $(F_1 \vee F_2)$ is the

"strongest" formula F for which both $(F_1 \models F)$ and $(F_2 \models F)$ hold. Thus $and(F_1, F_2) = (F_1 \wedge F_2)$ is the *product* of the $\mathcal{SYN}$-objects F_1 and F_2 and $or(F_1, F_2) = (F_1 \vee F_2)$ is their *coproduct* (see Definition 17). Furthermore, by Proposition 9, we have $\Delta_{\mathcal{SYN}} \dashv and$ and $or \dashv \Delta_{\mathcal{SYN}}$ i.e., functor *and* is the right adjoint of the diagonal functor $\Delta_{\mathcal{SYN}} \colon \mathcal{SYN} \to \mathcal{SYN} \times \mathcal{SYN}$ while functor *or* is its left one.

Correspondingly $AND(P_1, P_2)$ is the product of the $\mathcal{SEM}$-objects P_1 and P_2 (the "weakest" predicate P for which both $(P \subseteq P_1)$ and $(P \subseteq P_2)$ hold) and $OR(P_1, P_2)$ is their coproduct (the "strongest" predicate P for which $(P_1 \subseteq P)$ and $(P_2 \subseteq P)$ hold). By Proposition 9, we then have $\Delta_{\mathcal{SEM}} \dashv AND$ and $OR \dashv \Delta_{\mathcal{SEM}}$ i.e., functor *AND* is the right adjoint of the diagonal functor $\Delta_{\mathcal{SEM}} \colon \mathcal{SEM} \to \mathcal{SEM} \times \mathcal{SEM}$ while functor *OR* is its left one.

Thus we have, corresponding to the rules for formulas, the following rules for all predicates P_1, P_2, P:

$$AND(P_1, P_2) \subseteq P_1$$
$$AND(P_1, P_2) \subseteq P_2$$
$$(P \subseteq P_1) \wedge (P \subseteq P_2) \Rightarrow (P \subseteq AND(P_1, P_2))$$

Dually we have

$$P_1 \subseteq OR(P_1, P_2)$$
$$P_2 \subseteq OR(P_1, P_2)$$
$$(P_1 \subseteq P) \wedge (P_2 \subseteq P) \Rightarrow (OR(P_1, P_2) \subseteq P)$$

Since products and coproducts are uniquely defined, these rules actually represent implicit but unique definitions of $AND(P_1, P_2)$ and $OR(P_1, P_2)$ which can be explicitly written as follows:

$$AND(P_1, P_2) := \bigcup \{P \in Pred \mid P \subseteq P_1 \wedge P \subseteq P_2\}$$
$$OR(P_1, P_2) := \bigcap \{P \in Pred \mid P_1 \subseteq P \wedge P_2 \subseteq P\}$$

This gives us alternative characterizations $[\![F_1 \wedge F_2]\!] = AND([\![F_1]\!], [\![F_2]\!])$ and $[\![F_1 \vee F_2]\!] = [\![F_1]\!] \cup [\![F_2]\!] = OR([\![F_1]\!], [\![F_2]\!])$ that are both constructive and universal (Proposition 5 implies $[\![F_1 \wedge F_2]\!] = [\![F_1]\!] \cap [\![F_2]\!]$ and $[\![F_1 \vee F_2]\!] = [\![F_1]\!] \cup [\![F_2]\!]$ from which it is not difficult to verify these equalities).

Implication The role of implication in reasoning is exhibited by the following rules for arbitrary formulas F_1, F_2, F (the first rule mimics the logical inference rules of "implication elimination" or "modus ponens", the last one mimics the inference rule of "implication introduction"):

$$(F_1 \to F_2) \wedge F_1 \models F_2$$
$$(F \wedge F_1 \models F_2) \Rightarrow (F \models F_1 \to F_2)$$

These rules (whose soundness can be established with the help of Definition 8) state that $(F_1 \to F_2)$ is the "weakest" formula F for which $(F \wedge F_1 \models F_2)$ holds. Thus $imp_{F_1}(F_2) = (F_1 \to F_2)$ is the *exponential* of the $\mathcal{SYN}$-objects F_1 and F_2 (see Definition 19). Proposition 10 then gives us $and_{F_1} : imp_{F_1}$, i.e., functor imp_{F_1} is the right adjoint of the unary conjunction functor $and_{F_1} \colon \mathcal{SYN} \to \mathcal{SYN} \times \mathcal{SYN}$ with object map $and_{F_1}(F_2) := and(F_1, F_2) = F_1 \wedge F_2$.

Correspondingly $IMP_{P_1}(P_2)$ is the product of the $\mathcal{SEM}$-objects P_1 and P_2 (the "weakest" predicate P for which $(P \cap P_1 \subseteq P_2)$ holds; Proposition 10 then gives us $AND_{P_1} \dashv IMP_{P_1}$, i.e.,

functor IMP_{P_1} is the right adjoint of the unary functor $AND_{P_1} : \mathcal{SEM} \to \mathcal{SEM} \times \mathcal{SEM}$ with object map $AND_{P_1}(P_2) := AND(P_1, P_2) = P_1 \cup P_2$.

Thus, corresponding to above rules for formulas, we have the following rules for all predicates P_1, P_2, P:

$$IMP_{P_1}(P_2) \cap P_1 \subseteq P_2$$
$$(P \cap P_1 \subseteq P_2) \Rightarrow (P \subseteq IMP_{P_1}(P_2))$$

Since exponentials are uniquely defined, these rules represent an implicit but unique definition of $IMP_{P_1}(P_2)$ which can be explicitly written as follows:

$$IMP_{P_1}(P_2) := \bigcup \{P \in Pred \mid P \cap P_1 \subseteq P_2\}$$

This gives us an alternative characterization $[\![F_1 \to F_2]\!] = IMP_{[\![F_1]\!]}([\![F_2]\!])$ that is both constructive and universal (Proposition 5 implies $[\![F_1 \wedge F_2]\!] = \overline{[\![F_1]\!]} \cup [\![F_2]\!]$ from which it is possible to verify this equality).

Universal and Existential Quantification The role of universal quantification in reasoning is exhibited by the following rules for arbitrary formulas F, G provided that the semantics $[\![G]\!]$ of G *does not depend* on x (see Definition 4):

$$\forall x.\ F \models F$$
$$(G \models F) \Rightarrow (G \models \forall x.\ F)$$

The first rule mimics the logical inference rule of "universal elimination", the second one mimics the inference rule of "universal introduction" (except that our version of first-order logic does not involve terms and variables and thus copes without variable substitutions). This pair of rules in a nutshell says that $(\forall x.\ F)$ is the "weakest" formula G from which F is a logical consequence and whose semantics *does not depend* on x. Dually we have for existential quantification the following pair of rules:

$$F \models \exists x.\ F$$
$$(F \models G) \Rightarrow (\exists x.\ F \models G)$$

These rules state that $(\exists x.\ F)$ is the "strongest" formula G that is a logical consequence of F and whose semantics $[\![G]\!]$ *does not depend on* x.

We are now going to derive appropriate categorical characterizations of the corresponding functors $forall_{x \in Var} : \mathcal{SYN} \to \mathcal{SYN}_x$ and $exists_{x \in Var} : \mathcal{SYN} \to \mathcal{SYN}_x$ from the category $\mathcal{SYN}$ of all formulas to the subcategory $\mathcal{SYN}_x$ of all those formulas whose semantics does not depend on x. For this we may notice that above rules the relations $(G \models F)$ and $(F \models G)$ involve two kinds of relations, a more general relation F that may depend on x and a more special relation G that is independent of x. In order to bring all relations to the "same level", we introduce a syntactic "injection" functor $I_x : \mathcal{SYN}_x \to \mathcal{SYN}$ whose maps are just identities, i.e., $I_x(G) = G$ and $I_x(f : F \to G) = f : F \to G$. This allows us to express above rules as

$$I_x(forall_x(F)) \models F$$
$$(I_x(G) \models F) \Rightarrow (G \models forall_x(F))$$

and dually

$$F \models I_x(exists_x(F))$$
$$(F \models I_x(G)) \Rightarrow (exists_x(F) \models G)$$

Now the first set of rules matches the assumptions of the second part of Proposition 7 for $F := I_x$ and $G := forall_x$ (considering that the satisfaction relation $\models$ denotes the existence of an arrow in categories $\mathcal{SYN}$ respectively $\mathcal{SYN}_x$); thus we have $I_x \dashv forall_x$. Likewise, the second set of rules matches the assumptions of the first part of that proposition for $F := exists_x$ and $G := I_x$; thus we have $exists_x \dashv I_x$. Summarizing, the universal functor $forall_x$ is the right adjoint of the injection functor I_x while the existential functor $exists_x$ is its left adjoint.

These considerations can be easily transferred to categorical characterizations of the corresponding functors $FORALL_{x \in Var} \colon \mathcal{SEM} \to \mathcal{SEM}_x$ and $EXISTS_{x \in Var} \colon \mathcal{SEM} \to \mathcal{SEM}_x$ from the category $\mathcal{SEM}$ of all predicates to the subcategory $\mathcal{SEM}_x$ of all those predicates that do not depend on x with the semantic "injection" functor $J_x \colon \mathcal{SEM}_x \to \mathcal{SEM}$ whose maps are just identities, i.e., $J_x(Q) = Q$ and $J_x(f \colon P \to Q) = f \colon P \to Q$. We then have

$$J_x(FORALL_x(P)) \subseteq P$$
$$(J_x(Q) \subseteq P) \Rightarrow (Q \subseteq FORALL_x(P))$$

and dually

$$P \subseteq J_x(EXISTS_x(P))$$
$$(P \subseteq J_x(Q)) \Rightarrow (EXISTS_x(P) \subseteq Q)$$

Now the first set of rules matches the assumptions of the second part of Proposition 7 for $F := J_x$ and $G := FORALL_x$ (considering that the subset relation $\subseteq$ denotes the existence of an arrow in categories $\mathcal{SEM}$ respectively $\mathcal{SEM}_x$); thus we have $J_x \dashv FORALL_x$. Likewise, the second set of rules matches the assumptions of the first part of that proposition for $F := EXISTS_x$ and $G := J_x$; thus we have $EXISTS_x \dashv J_x$. Summarizing, the universal functor $FORALL_x$ is the right adjoint of the injection functor J_x while the existential functor $EXISTS_x$ is its left adjoint.

Above rules say that $FORALL_x(P)$ is the weakest predicate Q that does not depend on x for which $(J_x(Q) \subseteq Q)$ holds while $EXISTS_x(P)$ is the strongest predicate Q that does not depend on x for which $(Q \subseteq J_x(Q))$ holds. Since left and right adjoints are uniquely defined, these rules represent implicit but unique definitions of $FORALL_x(P)$ and $EXISTS_x(P)$ which can be explicitly written as follows:

$$FORALL_x(P) := \bigcup \{Q \in Pred_x \mid J_x(Q) \subseteq P\}$$
$$EXISTS_x(P) := \bigcap \{Q \in Pred_x \mid P \subseteq J_x(Q)\}$$

From $J_x(Q) = Q$ and $Q \in Pred_x \Leftrightarrow Q \in Pred \wedge Q \perp\!\!\!\perp x$ (see Definition 5), this can be also written as follows:

$$FORALL_x(P) := \bigcup \{Q \in Pred \mid Q \perp\!\!\!\perp x \wedge Q \subseteq P\}$$
$$EXISTS_x(P) := \bigcap \{Q \in Pred \mid Q \perp\!\!\!\perp x \wedge P \subseteq Q\}$$

Thus we have derived alternative characterizations $[\![\forall x.\ F]\!] = FORALL_x([\![F]\!])$ and $[\![\exists x.\ F]\!] = EXISTS_x([\![F]\!])$ that are both constructive and universal. Actually, this is exactly the characterization whose correctness we have proved in Proposition 6.

Summary We summarize the categorical semantics of our relational version of first-order logic by giving for each logical operation (logical constant, connective, and quantifier)

- the original set-theoretic definition of its semantics,
- an equivalent formulation by the application of a predicate functor,
- the logical "rules" associated to the operation,
- the corresponding rules for the predicate functor,
- the categorical semantics of the functor derived from the rule, and
- the characterization of the functor as a left or right adjoint of one of the following functors:

$$\begin{aligned} C_{\mathcal{SEM}} &: \mathcal{SEM} \to \mathbf{1}, & C_{\mathcal{SEM}}(P) &:= * \\ \Delta_{\mathcal{SEM}} &: \mathcal{SEM} \to \mathcal{SEM} \times \mathcal{SEM}, & \Delta_{\mathcal{SEM}}(P) &:= (P, P) \\ AND_{P_1 \in \mathcal{P}red} &: \mathcal{SEM} \to \mathcal{SEM}, & AND_{P_1}(P_2) &:= P_1 \cap P_2 \\ J_{x \in Var} &: \mathcal{SEM}_x \to \mathcal{SEM}, & J_x(F) &:= F \end{aligned}$$

$[\![\top]\!] = \boldsymbol{Ass} = \boldsymbol{TRUE}(*)$

$$F \models \top$$
$$P \subseteq TRUE(*)$$
$$TRUE(*) = \bigcup \{P \mid P \in Pred\}$$
$$C_{\mathcal{SEM}} \dashv TRUE$$

$[\![\top]\!] = \emptyset = \boldsymbol{FALSE}(*)$

$$\bot \models F$$
$$FALSE(*) \subseteq P$$
$$FALSE(*) = \bigcap \{P \mid P \in Pred\}$$
$$FALSE \dashv C_{\mathcal{SEM}}$$

$[\![\boldsymbol{F_1} \wedge \boldsymbol{F_2}]\!] = [\![\boldsymbol{F_1}]\!] \cap [\![\boldsymbol{F_2}]\!] = \boldsymbol{AND}([\![\boldsymbol{F_1}]\!], [\![\boldsymbol{F_2}]\!])$

$$F_1 \wedge F_2 \models F_1$$
$$F_1 \wedge F_2 \models F_2$$
$$(F \models F_1) \wedge (F \models F_2) \Rightarrow (F \models F_1 \wedge F_2)$$

$$AND(P_1, P_2) \subseteq P_1$$
$$AND(P_1, P_2) \subseteq P_2$$
$$(P \subseteq P_1) \wedge (P \subseteq P_2) \Rightarrow (P \subseteq AND(P_1, P_2))$$
$$AND(P_1, P_2) = \bigcup \{P \in Pred \mid P \subseteq P_1 \wedge P \subseteq P_2\}$$
$$\Delta_{\mathcal{SEM}} \dashv AND$$

$[\![\boldsymbol{F_1} \vee \boldsymbol{F_2}]\!] = [\![\boldsymbol{F_1}]\!] \cup [\![\boldsymbol{F_2}]\!] = \boldsymbol{OR}([\![\boldsymbol{F_1}]\!], [\![\boldsymbol{F_2}]\!])$

$$F_1 \models F_1 \vee F_2$$
$$F_2 \models F_1 \vee F_2$$
$$(F_1 \models F) \wedge (F_2 \models F) \Rightarrow (F_1 \vee F_2 \models F)$$

$$P_1 \subseteq OR(P_1, P_2)$$
$$P_2 \subseteq OR(P_1, P_2)$$
$$(P_1 \subseteq P) \wedge (P_2 \subseteq P) \Rightarrow (OR(P_1, P_2) \subseteq P)$$

$$OR(P_1, P_2) = \bigcap \{P \in Pred \mid P_1 \subseteq P \wedge P_2 \subseteq P\}$$

$$OR \dashv \Delta_{\mathcal{SEM}}$$

$[\![\boldsymbol{F_1} \rightarrow \boldsymbol{F_2}]\!] = \overline{[\![\boldsymbol{F_1}]\!]} \cup [\![\boldsymbol{F_2}]\!] = \boldsymbol{IMP}_{[\![\boldsymbol{F_1}]\!]}([\![\boldsymbol{F_2}]\!])$

$$(F_1 \rightarrow F_2) \wedge F_1 \models F_2$$
$$(F \wedge F_1 \models F_2) \Rightarrow (F \models F_1 \rightarrow F_2)$$

$$IMP_{P_1}(P_2) \cap P_1 \subseteq P_2$$
$$(P \cap P_1 \subseteq P_2) \Rightarrow (P \subseteq IMP_{P_1}(P_2))$$

$$IMP_{P_1}(P_2) = \bigcup \{P \in Pred \mid P \cap P_1 \subseteq P_2\}$$

$$AND_{P_1} \dashv IMP_{P_1}$$

$[\![\forall \boldsymbol{x}.\, \boldsymbol{F}]\!] = \{\boldsymbol{a} \in \boldsymbol{Ass} \mid \boldsymbol{a}[\boldsymbol{x} \mapsto \boldsymbol{v}] \in [\![\boldsymbol{F}]\!], \textbf{ for all } \boldsymbol{v} \in \boldsymbol{Val}\} = \boldsymbol{FORALL_x}([\![\boldsymbol{F}]\!])$

$$\forall x.\, F \models F$$
$$(G \models F) \Rightarrow (G \models \forall x.\, F)$$

$$FORALL_x(P) \subseteq P$$
$$(Q \subseteq P) \Rightarrow (Q \subseteq FORALL_x(P))$$

$$FORALL_x(P) = \bigcup \{Q \in Pred \mid Q \perp\!\!\!\perp x \wedge Q \subseteq P\}$$

$$J_x \dashv FORALL_x$$

$[\![\exists \boldsymbol{x}.\, \boldsymbol{F}]\!] = \{\boldsymbol{a} \in \boldsymbol{Ass} \mid \boldsymbol{a}[\boldsymbol{x} \mapsto \boldsymbol{v}] \in [\![\boldsymbol{F}]\!], \textbf{ for some } \boldsymbol{v} \in \boldsymbol{Val}\} = \boldsymbol{EXISTS_x}([\![\boldsymbol{F}]\!])$

$$F \models \exists x.\, F$$
$$(F \models G) \Rightarrow (\exists x.\, F \models G)$$

$$P \subseteq EXISTS_x(P)$$
$$(P \subseteq Q) \Rightarrow (EXISTS_x(P) \subseteq Q)$$

$$EXISTS_x(P) = \bigcap \{Q \in Pred \mid Q \perp\!\!\!\perp x \wedge P \subseteq Q\}$$

$$EXISTS_x \dashv J_x$$

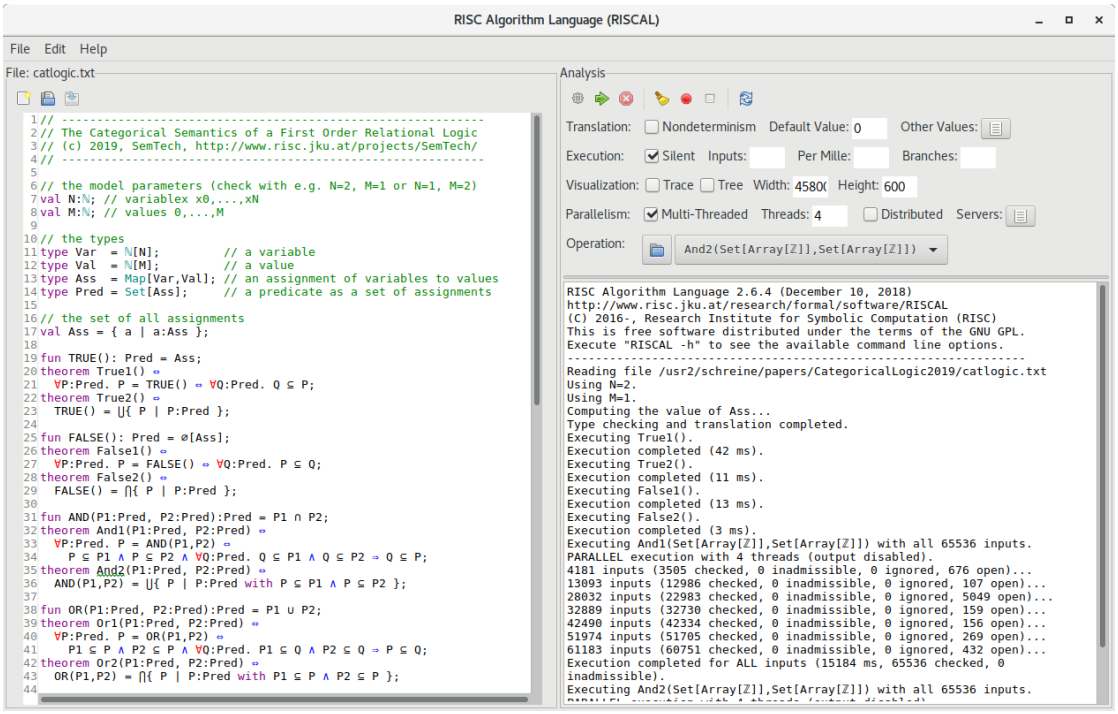


Figure 2: The RISCAL Software

5 An Implementation of the Categorical Semantics

In this section we describe how the constructions that we have theoretically modeled in Section 2 can be actually implemented. For this purpose we use RISCAL, the RISC Algorithm Language [10, 8], a language and associated software system for modeling mathematical theories and algorithms in a specification language based on first order logic and set theory. Since the domains of RISCAL models have (parameterized but) finite size, the validity of all theorems and the correctness of all algorithms can be fully automatically checked; the system has been mainly employed in educational scenarios [12, 11]. Figure 2 gives a screenshot of the software with the RISCAL model that is going to be discussed below.

Figures 3 and 4 list a RISCAL model of the categorical semantics over a domain of $N+1$ variables (identified with the natural numbers $0,\ldots,N$) with $M+1$ values, for arbitrary model parameters $N, M \in \mathbb{N}$; all theorems over these domains are decidable and can be checked by RISCAL. The RISCAL definition of domains, functions, predicates closely correspond to those given in this paper; in particular we have a domain `Pred` of predicates (since the number of variables is finite, by definition all relations are predicates) and predicate functions `TRUE`, `FALSE`, `AND`, `OR`, `IMP`, `FORALL`, `EXISTS`. Different from the categorical formulation `IMP` is a binary function, not a family of unary functions; likewise `FORALL` and `EXISTS` are binary functions whose first argument is a variable. Furthermore, we introduce functions `NOT` and `EQUIV` for the semantics of negation and conjunction and show by theorems `Not` and `Equiv` that they can be reduced to the other functions.

```
// -------------------------------------------------------------
// The Categorical Semantics of a First Order Relational Logic
// (c) 2019, SemTech, http://www.risc.jku.at/projects/SemTech/
// -------------------------------------------------------------

// the model parameters (check with e.g. N=2, M=1 or N=1, M=2)
val N:ℕ; // variablex x0,...,xN
val M:ℕ; // values 0,...,M

// the types
type Var  = ℕ[N];         // a variable
type Val  = ℕ[M];         // a value
type Ass  = Map[Var,Val]; // an assignment of variables to values
type Pred = Set[Ass];     // a predicate as a set of assignments

// the set of all assignments
val Ass = { a | a:Ass };

fun TRUE(): Pred = Ass;
theorem True1() ⇔
  ∀P:Pred. P = TRUE() ⇔ ∀Q:Pred. Q ⊆ P;
theorem True2() ⇔
  TRUE() = ⋃{ P | P:Pred };

fun FALSE(): Pred = ∅[Ass];
theorem False1() ⇔
  ∀P:Pred. P = FALSE() ⇔ ∀Q:Pred. P ⊆ Q;
theorem False2() ⇔
  FALSE() = ⋂{ P | P:Pred };

fun AND(P1:Pred, P2:Pred):Pred = P1 ∩ P2;
theorem And1(P1:Pred, P2:Pred) ⇔
  ∀P:Pred. P = AND(P1,P2) ⇔
    P ⊆ P1 ∧ P ⊆ P2 ∧ ∀Q:Pred. Q ⊆ P1 ∧ Q ⊆ P2 ⇒ Q ⊆ P;
theorem And2(P1:Pred, P2:Pred) ⇔
  AND(P1,P2) = ⋃{ P | P:Pred with P ⊆ P1 ∧ P ⊆ P2 };

fun OR(P1:Pred, P2:Pred):Pred = P1 ∪ P2;
theorem Or1(P1:Pred, P2:Pred) ⇔
  ∀P:Pred. P = OR(P1,P2) ⇔
    P1 ⊆ P ∧ P2 ⊆ P ∧ ∀Q:Pred. P1 ⊆ Q ∧ P2 ⊆ Q ⇒ P ⊆ Q;
theorem Or2(P1:Pred, P2:Pred) ⇔
  OR(P1,P2) = ⋂{ P | P:Pred with P1 ⊆ P ∧ P2 ⊆ P };

fun IMP(P1:Pred, P2:Pred):Pred = (Ass\P1) ∪ P2;
theorem Imp1(P1:Pred, P2:Pred) ⇔
  ∀P:Pred. P = IMP(P1,P2) ⇔
    P ∩ P1 ⊆ P2 ∧ ∀Q:Pred. Q ∩ P1 ⊆ P2 ⇒ Q ⊆ P;
theorem Imp2(P1:Pred, P2:Pred) ⇔
  IMP(P1,P2) = ⋃{ P | P:Pred with P ∩ P1 ⊆ P2 };
...
```

Figure 3: A RISCAL Model of the Categorical Semantics (Part 1)

```
...

fun NOT(P:Pred): Pred = Ass\P;
theorem Not(P:Pred) ⇔ NOT(P) = IMP(P,FALSE());

fun EQUIV(P1:Pred, P2:Pred): Pred =
  ((Ass\P1)∪P2) ∩ ((Ass\P2)∪P1);
theorem Equiv(P1:Pred, P2:Pred) ⇔
  EQUIV(P1,P2) = AND(IMP(P1,P2),IMP(P2,P1));

pred independent(P:Pred, x:Var) ⇔
  ∀a:Ass, v1:Val, v2:Val.
    (a with [x] = v1) ∈ P ⇔ (a with [x] = v2) ∈ P;

fun FORALL(x:Var, P:Pred):Pred =
  { a | a:Ass with ∀v:Val. (a with [x] = v) ∈ P } ;
theorem Forall1(x:Var, P:Pred) ⇔
  ∀Q:Pred with independent(Q,x). Q = FORALL(x,P) ⇔
    Q ⊆ P ∧ ∀Q0:Pred with independent(Q0,x). Q0 ⊆ P ⇒ Q0 ⊆ Q;
theorem Forall2(x:Var, P:Pred) ⇔
  FORALL(x,P) = ⋃{ Q | Q:Pred with independent(Q,x) ∧ Q ⊆ P };

fun EXISTS(x:Var, P:Pred):Pred =
  { a | a:Ass with ∃v:Val. (a with [x] = v) ∈ P } ;
theorem Exists1(x:Var, P:Pred) ⇔
  ∀Q:Pred with independent(Q,x). Q = EXISTS(x,P) ⇔
    P ⊆ Q ∧ ∀Q0:Pred with independent(Q0,x). P ⊆ Q0 ⇒ Q ⊆ Q0;
theorem Exists2(x:Var, P:Pred) ⇔
  EXISTS(x,P) = ⋂{ Q | Q:Pred with independent(Q,x) ∧ P ⊆ Q };

// ---------------------------------------------------------
// end of file
// ---------------------------------------------------------
```

Figure 4: A RISCAL Model of the Categorical Semantics (Part 2)

All other logical operations are first defined in their usual set theoretic form. Subsequently we describe their categorical semantics by a pair of theorems: the first theorem claims that the set theoretic semantics is equivalent to an implicit definition of the categorical semantics while the second theorem claims equivalence to the corresponding constructive definition. Choosing small parameter values $N = 2$ and $M = 1$ (i.e., relations with variables x_0, x_1, x_2 and values $0, 1$), RISCAL can easily check the validity of all claims, as demonstrated by the following output:

```
RISC Algorithm Language 2.6.4 (December 10, 2018)
http://www.risc.jku.at/research/formal/software/RISCAL
(C) 2016-, Research Institute for Symbolic Computation (RISC)
This is free software distributed under the terms of the GNU GPL.
Execute "RISCAL -h" to see the available command line options.
---------------------------------------------------------------
Reading file /usr2/schreine/papers/CategoricalLogic2019/catlogic.txt
Using N=2.
Using M=1.
Computing the value of Ass...
Computing the value of TRUE...
Computing the value of FALSE...
Type checking and translation completed.
Executing True1().
Execution completed (3 ms).
Executing True2().
Execution completed (1 ms).
Executing False1().
Execution completed (0 ms).
Executing False2().
Execution completed (1 ms).
Executing And1(Set[Array[ℤ]],Set[Array[ℤ]]) with all 65536 inputs.
PARALLEL execution with 4 threads (output disabled).
...
Execution completed for ALL inputs (18373 ms, 65536 checked, 0 inadmissible).
Executing And2(Set[Array[ℤ]],Set[Array[ℤ]]) with all 65536 inputs.
PARALLEL execution with 4 threads (output disabled).
46273 inputs (36446 checked, 0 inadmissible, 0 ignored, 9827 open)...
Execution completed for ALL inputs (3576 ms, 65536 checked, 0 inadmissible).
Executing Or1(Set[Array[ℤ]],Set[Array[ℤ]]) with all 65536 inputs.
PARALLEL execution with 4 threads (output disabled).
...
Execution completed for ALL inputs (26889 ms, 65536 checked, 0 inadmissible).
Executing Or2(Set[Array[ℤ]],Set[Array[ℤ]]) with all 65536 inputs.
PARALLEL execution with 4 threads (output disabled).
42676 inputs (32887 checked, 0 inadmissible, 0 ignored, 9789 open)...
Execution completed for ALL inputs (3907 ms, 65536 checked, 0 inadmissible).
Executing Imp1(Set[Array[ℤ]],Set[Array[ℤ]]) with all 65536 inputs.
PARALLEL execution with 4 threads (output disabled).
...
Execution completed for ALL inputs (48592 ms, 65536 checked, 0 inadmissible).
Executing Imp2(Set[Array[ℤ]],Set[Array[ℤ]]) with all 65536 inputs.
PARALLEL execution with 4 threads (output disabled).
...
Execution completed for ALL inputs (9462 ms, 65536 checked, 0 inadmissible).
Executing Not(Set[Array[ℤ]]) with all 256 inputs.
PARALLEL execution with 4 threads (output disabled).
Execution completed for ALL inputs (28 ms, 256 checked, 0 inadmissible).
Executing Equiv(Set[Array[ℤ]],Set[Array[ℤ]]) with all 65536 inputs.
PARALLEL execution with 4 threads (output disabled).
Execution completed for ALL inputs (354 ms, 65536 checked, 0 inadmissible).
Executing Forall1(ℤ,Set[Array[ℤ]]) with all 768 inputs.
PARALLEL execution with 4 threads (output disabled).
```

```
Execution completed for ALL inputs (1315 ms, 768 checked, 0 inadmissible).
Executing Forall2(ℤ,Set[Array[ℤ]]) with all 768 inputs.
PARALLEL execution with 4 threads (output disabled).
Execution completed for ALL inputs (512 ms, 768 checked, 0 inadmissible).
Executing Exists1(ℤ,Set[Array[ℤ]]) with all 768 inputs.
PARALLEL execution with 4 threads (output disabled).
Execution completed for ALL inputs (1299 ms, 768 checked, 0 inadmissible).
Executing Exists2(ℤ,Set[Array[ℤ]]) with all 768 inputs.
PARALLEL execution with 4 threads (output disabled).
Execution completed for ALL inputs (461 ms, 768 checked, 0 inadmissible).
```

These values are, however, the largest ones with which model checking is realistically feasible; choosing for example $N = 3$ and $M = 2$ gives for checking theorem `And1` about $4 \cdot 10^9$ possible inputs whose checking on a single processor core would take RISCAL more than two decades.

6 Conclusions

We hope that this paper, by its self-contained nature and by focusing on the core principles of categorical logic rather than attempting an exhaustive treatment, contributes to a more widespread dissemination of categorical ideas to students and researchers of logic, its applications, and its automation; in particular it may provide an alternative view on the semantics of first-order logic by complementing the classical formulation and thus help to gain deeper insights.

It remains to be shown, however, whether and how this view can be indeed helpful and illuminating in educational scenarios, i.e., in courses on logic and its applications. In previous work [16, 18, 17, 13], we have strived to improve the understanding of the formal semantics of programming languages by developing corresponding tools with appropriate visualization techniques, partially also based on categorical principles. Other work of ours [14] has extended this work towards the visualization of the semantics of first-order formulas by pruned evaluation trees, however, based on the classical formulation. Future work of us will investigate how the categorical principles outlined on this paper can be transferred to corresponding novel tools and visualization techniques for education in semantics and logic.

References

[1] Samson Abramsky. "Logic and Categories As Tools For Building Theories". In: *Journal of Indian Council of Philosophical Research, Issue on Logic and Philosophy Today* 27.1 (2010), pp. 277–304. URL: https://arxiv.org/abs/1201.5342.

[2] Samson Abramsky and Nikos Tzevelekos. "Introduction to Categories and Categorical Logic". In: *New Structures for Physics*. Ed. by Bob Coecke. Vol. 813. Lecture Notes in Physics. Berlin, Germany: Springer, 2010, pp. 3–94. DOI: 10.1007/978-3-642-12821-9_1.

[3] Steve Awodey. *Category Theory*. Second Edition. Okford, UK: Oxford University Press, 2010.

[4] Bart Jacobs. *Categorical Logic and Type Theory*. Vol. 141. Studies in Logic and the Foundations of Mathematics. Amsterdam, The Netherlands: North Holland, Elsevier, 1999.

[5] F. William Lawvere. "Adjointness in Foundations". In: *Dialectica* 23.3/4 (1969). Reprint available from http://www.tac.mta.ca/tac/reprints/articles/16/tr16.pdf. URL: https://www.jstor.org/stable/42969800.

[6] Benjamin C. Pierce. *Basic Category for Computer Scientists*. Cambridge, MA, USA: MIT Press, 1991.

[7] Axel Poigné. "Category Theory and Logic". In: *Category Theory and Computer Programming: Tutorial and Workshop*. Vol. 240. Lecture Notes in Computer Science. Guildford, UK, September 16-–20: Springer, Berlin, Germany, 1985, pp. 103–142. DOI: 10.1007/3-540-17162-2_119.

[8] RISCAL. *The RISC Algorithm Language (RISCAL)*. Jan. 2018. URL: https://www.risc.jku.at/research/formal/software/RISCAL.

[9] David A. Schmidt. *Denotational Semantics — A Methodology for Language Development*. Boston, MA, USA: Allyn and Bacon, 1986. URL: http://people.cis.ksu.edu/~schmidt/text/densem.html.

[10] Wolfgang Schreiner. *The RISC Algorithm Language (RISCAL) — Tutorial and Reference Manual (Version 1.0)*. Technical Report. Available at [8]. Johannes Kepler University, Linz, Austria: RISC, Mar. 2017.

[11] Wolfgang Schreiner. "Validating Mathematical Theories and Algorithms with RISCAL". In: *CICM 2018, 11th Conference on Intelligent Computer Mathematics, Hagenberg, Austria, August 13–17*. Ed. by F. Rabe, W. Farmer, G. Passmore, and A. Youssef. Vol. 11006. Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence. Springer, Berlin, 2018, pp. 248–254. DOI: 10.1007/978-3-319-96812-4_21.

[12] Wolfgang Schreiner, Alexander Brunhuemer, and Christoph Fürst. "Teaching the Formalization of Mathematical Theories and Algorithms via the Automatic Checking of Finite Models". In: *Post-Proceedings ThEdu'17, Theorem proving components for Educational software, Gothenburg, Sweden, August 6, 2017*. Ed. by Pedro Quaresma and Walther Neuper. Vol. 267. EPTCS. 2018, pp. 120–139. DOI: 10.4204/EPTCS.267.8.

[13] Wolfgang Schreiner and William Steingartner. *Visualizing Execution Traces in RISCAL*. Technical Report. Available at [8]. Johannes Kepler University, Linz, Austria: RISC, Mar. 2018.

[14] Wolfgang Schreiner and William Steingartner. *Visualizing Logic Formula Evaluation in RISCAL*. Technical Report. Available at [8]. Johannes Kepler University, Linz, Austria: RISC, July 2018.

[15] David I. Spiwak. *Category Theory for the Sciences*. Cambridge, MA, USA: MIT Press, 2014.

[16] William Steingartner, Mohamed Ali M. Eldojali, Davorka Radaković, and Jiří Dostál. "Software support for course in Semantics of programming languages". In: *IEEE 14th International Scientific Conference on Informatics*. Poprad, Slovakia, November 14–16, 2017, pp. 359–364. URL: https://www.researchgate.net/publication/321341571_Software_support_for_course_in_Semantics_of_programming_languages.

[17] William Steingartner and Valerie Novitzká. "Categorical Semantics of Programming Langages". In: *Selected Topics in Contemporary Mathematical Modeling*. Ed. by Andrzej Z. Grzybowski. Vol. 331. Monographs. Czestochowa University of Technology. Chap. 11, pp. 167–192. URL: https://doi.org/10.17512/STiCMM2017.11.

[18] William Steingartner and Valerie Novitzká. "Learning tools in course on semantics of programming languages". In: *MMFT 2017 — Mathematical Modelling in Physics and Engineering*. Czestochowa, Poland, September 18–21, 2017, pp. 137–142. URL: http://im.pcz.pl/konferencja/get.php?doc=MMFT2017_streszczenia_wykladow.pdf.

[19] Alfred Tarski. "The Semantic Conception of Truth: and the Foundations of Semantics". In: *Philosophy and Phenomenological Research* 4.3 (1944), pp. 341–376. URL: https://doi.org/10.2307/2102968.