**Abstract**

RelAPS is an interactive system assisting in proving relation-algebraic theorems. The aim of the system is to provide an environment where a user can perform a relation-algebraic proof similar to doing it using pencil and paper. The previous version of RelAPS accepts only Horn-formulas. To extend the system to first order logic, we have defined and implemented a new language based on theory of allegories as well as a new calculus. The language has two different kinds of terms; object terms and relational terms, where object terms are built from object constant symbols and object variables, and relational terms from typed relational constant symbols, typed relational variables, typed operation symbols and the regular operations available in any allegory. The calculus is a mixture of natural deduction and the sequent calculus. It is formulated in a sequent style but with exactly one formula on the right-hand side. We have shown soundness and completeness of this new logic which verifies that the underlying proof system of RelAPS is working correctly.

# Contents

**1 Introduction** **1**

**2 Background** **4**
2.1 Categories . . . . . 4
2.2 Allegories . . . . . 5

**3 Formal Language of Relational Categories** **7**
3.1 Syntax . . . . . 7
3.2 Semantics . . . . . 11

**4 Calculus of Relational Categories** **23**
4.1 Inference Rules . . . . . 23
4.2 Soundness Proof of Calculus . . . . . 27
4.3 Completeness Proof of Calculus . . . . . 30

**5 RelAPS** **48**
5.1 Overview of RelAPS . . . . . 48
5.2 Extending RelAPS . . . . . 53
5.2.1 A Manual for the System . . . . . 53
5.2.2 Implementation . . . . . 60

**6 Conclusions** **61**
6.1 Summary and Related Works . . . . . 61
6.2 Future Work . . . . . 63

# List of Figures


4.1 Axiom rules . . . . . 24
4.2 Structural rules . . . . . 25
4.3 Logical rules . . . . . 26
4.4 Example Derivation . . . . . 26
4.5 Derivation Tree for $T_n, \psi'_1, \ldots, \psi_{m-1} \vdash \bot$ . . . . . 45

5.1 Example of Valid and Invalid Formulas . . . . . 49
5.2 After Automatic Rules Applied . . . . . 49
5.3 Splitting Equality to two Inclusions . . . . . 50
5.4 Creating two Inclusions . . . . . 50
5.5 Selecting a Term . . . . . 51
5.6 Derivation Steps . . . . . 51
5.7 Completed Proof . . . . . 52
5.8 View of the New Version of RelAPS . . . . . 53
5.9 Derivation Tree for Example . . . . . 55
5.10 Creating a New Derivation . . . . . 56
5.11 Using Left Hand Rule Buttons . . . . . 57
5.12 a View of System After proving the Left Sub Tree . . . . . 58
5.13 Replacing Relational Variables by New Terms . . . . . 58
5.14 Using Working Area Window for Applying (=L) Rule . . . . . 59
5.15 Complete Tree of Proof . . . . . 59

# Chapter 1

# Introduction

Relation algebras and allegories are used in various areas of computer science. Among several formal approaches, relation algebra has been used as a basis for analyzing, modeling or resolving several computer science problems such as program specification, program fault tolerance, data abstraction and information coding, and spatial reasoning. Relations are well suited for describing certain types of problems and also contribute to the corresponding proof theory. Hence, any attempt to streamline the process of relational reasoning would be of benefit especially to those interested in program semantics and correctness [14, 15].

RelAPS is an interactive system assisting in proving relation-algebraic theorems. The aim of the system is to provide an environment where a user can perform a relation-algebraic proof similar to doing it using pencil and paper.

The previous version of RelAPS is mainly focused on equational reasoning, i.e., stepwise manipulating expressions using general equations similar to regular algebra. Therefore, the system only accepts Horn-formulas as potential theorems [7]. Horn-formulas are formulas of the form $(\forall x_1) \ldots (\forall x_m) e_1 \wedge \ldots \wedge e_n \rightarrow e$ where $\forall$ denotes universal quantification, $x_1, ..., x_m$ are variables, $\wedge$ denotes logical conjunction, $\rightarrow$ denotes implication, and $e_1, ..., e_n, e$ are atomic propositions, i.e., they do not contain any further logical symbols. For example, the property of a binary relation $\leq$ of being transitive can be formulized in a Horn-formula by $(\forall x)(\forall y)(\forall z) x \leq y \wedge y \leq z \rightarrow x \leq z$. On the other hand, if $x|y$ denotes the fact that $x$ divides $y$, then the obvious formulation of $y$ being prime $(\forall x) x|y \wedge \neg(x = 1) \rightarrow x = y$ is not a Horn-formula because of the application of the negation operator $\neg$ on the left hand side of the implication. In this thesis we performed several individual steps in order to extend RelAPS to full first order logic.

The first step is to define a proper language [3]. The language needs two different

types of terms. Obviously, one needs terms to denote relations themselves. In addition, relations are typed since they may act between different kinds of elements, in general. For example, the relation 'is_owned_by' is a binary relation between cars and humans. Therefore, terms denoting objects, the categorical notion of types, are also needed. Object terms are built from object constant symbols and object variables. Relational terms need to be typed because relations are between two different types of objects. Relational terms are built from typed relational constant symbols, typed relational variables, typed operation symbols and the regular operations available in any allegory.

The next step is to provide a suitable interpretation of the entities of the language [3]. The main part of this step is finding a proper environment and a model and then defining terms value and formulas validity. This is similar to regular first order logic but has to be done for formulas and both kind of terms considering the typing of relational terms.

The main requirement of the system is to define and implement a formal calculus for reasoning about allegories. Our calculus is a mixture of natural deduction and the sequent calculus [5, 6]. Natural deduction is a system that mimics human reasoning very well. However, it is not very well suited for computer applications since derivations are trees and individual steps may affect the whole tree. On the other hand, the sequent calculus expressions of the form $\Gamma \vdash \Delta$, where $\Gamma$ and $\Delta$ are (possibly empty) sequences of logical formulas, are modified in each individual step, i.e., those steps are local. Therefore, this calculus can easily be used in computer applications. The overall derivation does not necessarily reflect the 'natural' way of reasoning. In particular, the fact that the right hand side of a sequent might contain multiple or no formula is sometimes not very intuitive. In order to combine the advantages of both calculi our calculus is formulated in a sequent style but with exactly one formula on the right hand side.

For every logical calculus two properties are of particular interest, soundness and completeness. Soundness is the property that every formula that can be derived is also true. Completeness is the opposite statement formulating that every true formula can also be derived in the calculus. As usual, soundness of calculus has been proven by induction on the structure of the calculus rules. Our completeness proof is based on Henkins' famous completeness proof [9, 10]. His proof started by claiming that the completeness of first order logic is equivalent to showing that every consistent theory $T$ has a model. In order to construct a model for a consistent theory, Henkin chose the syntactic material itself. To this end, he enriched the language with enough new individual constants. However, Henkins' proof had to be modified extensively since our language contains two different kinds of terms and,

more importantly, it is typed.

# Chapter 2

# Background

In this chapter we want to introduce the basic mathematical notion used in the thesis. We start by introducing categories which are the underlying structure of allegories. Afterwards we define allegories which constitute a suitable abstract theory for binary relations.

## 2.1 Categories

A category is an algebraic structure consisting of a collection of *objects*, linked together by a collection of *morphisms* that have two basic properties; for each object exists an identity morphism and the morphisms can be composed associatively [1].

**Definition 1** *A category $\mathcal{C}$ consists of*

1. *A class of objects $Obj_{\mathcal{C}}$,*
2. *For every pair of objects $A$ and $B$ a class of morphisms $\mathcal{C}[A, B]$,*
3. *An associative binary operation ; mapping each pair of morphisms $f$ in $\mathcal{C}[A, B]$ and $g$ in $\mathcal{C}[B, C]$ to a morphism $f; g$ in $\mathcal{C}[A, C]$,*
4. *For every object $A$ a morphism $\mathbb{I}_A$ such that for all $f$ in $\mathcal{C}[A, B]$ and $g$ in $\mathcal{C}[C, A]$ we have $\mathbb{I}_A; f = f$ and $g; \mathbb{I}_A = g$.*

One of the common categories is **Set**. The objects of this category are sets and for every object $A$ and $B$, $Set[A, B]$ is the set of all functions from $A$ to $B$. The identity morphisms are the identity functions and the composition is the usual composition of functions. **Top** with topological spaces and continuous functions, **Vec** with vector

spaces and linear mappings, and **PO** with posets and monotone functions are other examples of categories.

## 2.2 Allegories

An allegory is a category that has some of the structure of the category of sets and binary relations [4]. In this sense, allegories are also a generalization of relation algebras [16] introduced by A. Tarski since a relation in an allegory can be between different sorts.

**Definition 2** *An allegory $\mathcal{R}$ is a category satisfying the following:*

1. *For all objects $A$ and $B$ the class $\mathcal{R}[A, B]$ is a lower semilattice. Meet and the induced ordering are denoted by $\sqcap, \sqsubseteq$ respectively. The elements in $\mathcal{R}[A, B]$ are called relations.*

2. *There is a monotone operation $^\smile$ called the converse operation such that for all relations $R : A \to B$ and $Q : B \to C$ the following holds:*

$$(R; Q)^\smile = Q^\smile; R^\smile \qquad \text{and} \qquad (R^\smile)^\smile = R.$$

3. *For all relations $R : A \to B$, and $Q, S : B \to C$ we have $R; (Q \sqcap S) \sqsubseteq R; Q \sqcap R; S$.*

4. *For all relations $R : A \to B$, $Q : B \to C$ and $S : A \to C$ the modular law $R; Q \sqcap S \sqsubseteq R; (Q \sqcap R^\smile; S)$ holds.*

A first example of an allegory is *Rel*. The objects of this allegory are sets, and the morphisms in $Rel[A, B]$ are binary relations between $A$ and $B$, i.e., subsets of the Cartesian product $A \times B$ of $A$ and $B$. Composition of morphisms is composition of relations, converse of morphisms is converse of relations and intersection of morphisms is intersection of relations.

For example, suppose $A = \{0, 1\}$ and $B = \{a, b, c\}$. Then $Rel[A, B] = \mathbb{P}(A \times B)$ and $Rel[B, A] = \mathbb{P}(B \times A)$. If $Q, S, P \in Rel[A, B]$, $T \in Rel[B, A]$ and $Q = \{(0, a), (1, c), (1, b), (0, b)\}$, $S = \{(0, a), (1, c), (1, b)\}$, $P = \{(0, a), (1, c)\}$ and $T = \{(a, 1), (c, 0)\}$ then we have:

$$S^\smile = \{(a, 0), (c, 1), (b, 1)\},$$

$$S \sqcap P = Q \sqcap P = P,$$

$$S;T = \{(0,1),(1,0)\} \text{ and } T;S = \{(a,c),(a,b),(c,a)\}.$$

As an other example, every modular lattice with a smallest element is an one-object allegory. Composition is given by the join and meet by the meet in the lattice. The identity is the smallest element and converse of an element is the element itself. The modular law above (4.) is then equivalent to the regular modular property.

In the following chapters, allegories will become the models of our logic. It is usually assumed that models are not empty.

**Definition 3** *A non-empty allegory $\mathcal{R}$ is an allegory which has at least one object and $\mathcal{R}[A,B]$ is not empty for every pair of objects $A$ and $B$.*

The smallest non-empty allegory is an allegory with only one object $A = \{1\}$ and one relation $\mathbb{I}_A = \{(1,1)\}$.

# Chapter 3

# Formal Language of Relational Categories

The main purpose of a proof system is verifying the validity of a formula. This requires a formal definition of a language, a suitable notion of an interpretation, and a precise definition of the meaning of each sentence of the language [3]. In this chapter, first we introduce the syntax of a formal language of allegories, and then we define the semantics of this language [17, 18].

## 3.1 Syntax

In order to provide a proper language for allegories, we require a set of object variables $V_{obj}$ and a set of object constant symbols $C_{obj}$. The two sets $V_{obj}$ and $C_{obj}$ as well as similar sets introduced later are supposed to be disjoint, i.e., $V_{obj} \cap C_{obj} = \emptyset$.

**Definition 4** *The set of object terms consists of object variables and object constant symbols.*

We also require the following components:

- $V_{rel}$ is a countable set of relational variables. Each variable $r$ has a type $t_1 \to t_2$ where $t_1$ and $t_2$ are object terms. To indicate that the variable $r$ has type $t_1 \to t_2$ we write $r : t_1 \to t_2$,
- $C_{rel}$ is a countable set of relational constant symbols. Each constant symbol $c$ has a type $t_1 \to t_2$ where $t_1$ and $t_2$ are object terms. To indicate that the constant symbol $c$ has type $t_1 \to t_2$ we write $c : t_1 \to t_2$,

- $F$ is a countable set of typed function symbols. Each function symbol $f$ has a type $\{(t_1 \to s_1), ..., (t_n \to s_n)\} \to (t \to s)$ where $t_1, s_1, ..., t_n, s_n, t, s$ are object terms. To indicate that the variable $f$ has type $\{(t_1 \to s_1), ..., (t_n \to s_n)\} \to (t \to s)$ we write $f : \{(t_1 \to s_1), ..., (t_n \to s_n)\} \to (t \to s)$.

**Definition 5** *The set of relational terms of type $s_1 \to s_2$, where $s_1$ and $s_2$ are object terms is defined recursively as follows:*

1. *If $r : s_1 \to s_2$ is a relational variable, then $r$ is a relational term of type $s_1 \to s_2$.*
2. *If $c : s_1 \to s_2$ is a relational constant symbol, then $c$ is a relational term of type $s_1 \to s_2$.*
3. *If $s$ is an object term, then $\mathbb{I}_s$ is a relational term of type $s \to s$.*
4. *If $t$ is a relational term of type $s_1 \to s_2$, then $t^\smile$ is a relational term of type $s_2 \to s_1$.*
5. *If $t_1$ and $t_2$ are relational terms of type $s_1 \to s_2$, then $t_1 \sqcap t_2$ is a relational term of type $s_1 \to s_2$.*
6. *If $t_1$ and $t_2$ are relational terms of type $s_1 \to s_2$ resp. $s_2 \to s_3$, then $t_1; t_2$ is a relational term of type $s_1 \to s_3$.*
7. *If $t_1$,...,$t_n$ are relational terms of type $s_1 \to s'_1$,..., $s_n \to s'_n$ and $f$ is a n-ary function symbol with type $f : \{(s_1 \to s'_1), ..., (s_n \to s'_n)\} \to (s \to s')$, then $f(t_1, ..., t_n)$ is a relational term of type $s \to s'$.*

$r^\smile$ and $\mathbb{I}_{s_1}; (q \sqcap u)$ are examples of relational terms where $r, q, u : r : s_1 \to s_2$ are relational variables.

In order to define formulas we need an additional component, a countable set $P$ of typed predicate symbols. Each predicate symbol $p$ has a type $\{(t_1 \to s_1), ..., (t_n \to s_n)\}$ where $t_1, s_1, ..., t_n, s_n$, are object terms. To indicate that the predicate symbol $p$ has type $\{(t_1 \to s_1), ..., (t_n \to s_n)\}$ we write $p : \{(t_1 \to s_1), ..., (t_n \to s_n)\}$. Finally we can define the set of formulas.

**Definition 6** *The set of formulas is defined recursively as follows:*

1. *$\perp$ is a formula.*
2. *If $t_1$ and $t_2$ are relational terms of type $s_1 \to s_2$, then $t_1 = t_2$ is a formula.*

3. *If $t_1, \ldots, t_n$ are relational terms of type $s_1 \to s'_1$,..., $s_n \to s'_n$ and $p$ is a n-ary predicate symbol with type $\{(s_1 \to s'_1), ..., (s_n \to s'_n)\}$, then $p(t_1, \ldots, t_n)$ is a formula.*

4. *If $\varphi_1$ and $\varphi_2$ are formulas, then $\varphi_1 \wedge \varphi_2$ is a formula.*

5. *If $\varphi_1$ and $\varphi_2$ are formulas, then $\varphi_1 \vee \varphi_2$ is a formula.*

6. *If $\varphi_1$ and $\varphi_2$ are formulas, then $\varphi_1 \to \varphi_2$ is a formula.*

7. *If $\varphi$ is a formula, then $\neg\varphi$ is a formula.*

8. *If $\varphi$ is a formula and $r : s_1 \to s_2$ is a relation variable, then $(\forall r : s_1 \to s_2)\varphi$ is a formula.*

9. *If $\varphi$ is a formula and $a$ is an object variable, then $(\forall a)\varphi$ is a formula.*

10. *If $\varphi$ is a formula and $r : s_1 \to s_2$ is a relation variable, then $(\exists r : s_1 \to s_2)\varphi$ is a formula.*

11. *If $\varphi$ is a formula and $a$ is an object variable, then $(\exists a)\varphi$ is a formula.*

Let $\sqsubseteq$ be a predicate symbol with type $\{(s_1 \to s_2), (s_1 \to s_2)\}$ for any object terms $s_1, s_2$ and $r, q : a \to b$ be two relational variables. So both $r \sqsubseteq q$ and $r = q$ are valid formulas as well as $(\forall r : a \to b)(r \sqsubseteq q \to r = q)$.

We adopt certain precedence rules of the logical symbols. $\neg$ has higher precedence than $\wedge$, $\wedge$ higher than $\vee$, $\vee$ higher than $\to$, and $\to$ higher than $=$. The precedence of $\forall$ and $\exists$ is the same as $\neg$.

In the next step, we want to introduce the concept of free variables in a formula. In order to do so we have to define the set of object and relational variables in a term first.

**Definition 7** *The set of object variables $OV(s)$ of an object term $s$ is defined recursively as follows:*

1. *$OV(a) = \{a\}$ for every object variable $a$,*

2. *$OV(c) = \emptyset$ for every object constant symbol $c$ in $C_{obj}$.*

**Definition 8** *The set of object variables $OV(t)$ and the set of relational variables $RV(t)$ of a relational term $t$ is defined recursively as follows:*

1. *$OV(r) = OV(s_1) \cup OV(s_2)$ for a relational variable $r : s_1 \to s_2$,*

2. $OV(c) = OV(s_1) \cup OV(s_2)$ *for a relational constant symbol* $c : s_1 \to s_2$ *in* $C_{rel}$,
3. $OV(t^\smile) = OV(t)$,
4. $OV(t_1; t_2) = OV(t_1 \sqcap t_2) = OV(t_1) \cup OV(t_2)$,
5. $OV(f(t_1, ..., t_n)) = OV(t_1) \cup ... \cup OV(t_n)$ *for every function symbol* $f : \{(s_1 \to s_1'), ..., (s_n \to s_n')\} \to (s \to s')$.
6. $RV(r) = \{r\}$ *for every relational variable* $r$,
7. $RV(c) = \emptyset$ *for every relational constant symbol* $c$ *in* $C_{rel}$,
8. $RV(t^\smile) = RV(t)$,
9. $RV(t_1; t_2) = RV(t_1 \sqcap t_2) = RV(t_1) \cup RV(t_2)$,
10. $RV(f(t_1, ..., t_n)) = RV(t_1) \cup ... \cup RV(t_n)$ *for every function symbol* $f$.

Now we can define free object and relational variables in a formula.

**Definition 9** *The set of free object variables* $OFV(\varphi)$ *and the set of free relational variables* $RFV(\varphi)$ *of a formula* $\varphi$ *is defined as follows:*

1. $OFV(\bot) = \emptyset$,
2. $OFV(t_1 = t_2) = OV(t_1) \cup OV(t_2)$,
3. $OFV(p(t_1, ..., t_n)) = OV(t_1) \cup ... \cup OV(t_n)$,
4. $OFV(\varphi_1 \otimes \varphi_2) = OFV(\varphi_1) \cup OFV(\varphi_2)$ *where* $\otimes \in \{\wedge, \vee, \to\}$,
5. $OFV(\neg\varphi) = OFV(\varphi)$,
6. $OFV((Qa)\varphi) = OFV(\varphi) \setminus \{a\}$ *where* $Q \in \{\forall, \exists\}$,
7. $OFV((Qr : s_1 \to s_2)\varphi) = OFV(\varphi) \cup OV(r)$ *where* $Q \in \{\forall, \exists\}$,
8. $RFV(\bot) = \emptyset$,
9. $RFV(t_1 = t_2) = RV(t_1) \cup RV(t_2)$,
10. $RFV(p(t_1, ..., t_n)) = RV(t_1) \cup ... \cup RV(t_n)$,
11. $RFV(\varphi_1 \otimes \varphi_2) = RFV(\varphi_1) \cup RFV(\varphi_2)$ *where* $\otimes \in \{\wedge, \vee, \to\}$,

12. $RFV(\neg\varphi) = RFV(\varphi)$,

13. $RFV((Qa)\varphi) = RFV(\varphi)$ *where* $Q \in \{\forall, \exists\}$,

14. $RFV((Qr : s_1 \to s_2)\varphi) = RFV(\varphi) \setminus \{r\}$ *where* $Q \in \{\forall, \exists\}$,

For example, for the formula $(\forall r : a \to a)r = r$ we have:

$$\begin{aligned} RFV((\forall r : a \to a)r = r) &= RFV(r = r) \setminus \{r\} \\ &= RV(r) \cup RV(r) \setminus \{r\} \\ &= \{r\} \setminus \{r\} = \emptyset \end{aligned}$$

$$\begin{aligned} OFV((\forall r : a \to a)r = r) &= OFV(r = r) \cup OV(r) \\ &= OV(r) \cup OV(r) \cup OV(r) \\ &= OV(a) = \{a\} \end{aligned}$$

.

**Definition 10** *Relational variable $r : s_1 \to s_2$ resp. object variable $a$ in a formula $\psi$ is called* free *iff $r \in RFV(\psi)$ resp. $a \in OFV(\psi)$. It is called* bounded *iff it is not free. A formula $\psi$ that does not contain any free object or relational variable, i.e., $RFV(\psi) = \emptyset$ and $OFV(\psi) = \emptyset$, is called closed.*

So according to the previous definition the formula $(\forall r : a \to a)r = r$ is not closed because it contains the free object variable $a$. On the other hand, the formula $(\exists a)(\forall r : a \to a)r = r$ is closed.

## 3.2 Semantics

We want to define what it means for a formula to be valid. Therefore, we first need a universe where all syntactic entities can be interpreted by suitable values.

**Definition 11** *A pre-model $\mathcal{P}$ consists of the following data:*

1. $|\mathcal{P}|$ *a non-empty allegory,*

2. *For each constant symbol $c \in C_{obj}$ a constant $c^{\mathcal{P}} \in \mathrm{Obj}_{|\mathcal{P}|}$.*

In order to define the semantics of terms and formulas we have to replace the free variables of the formula by actual values. Those values are stored in so called environments.

**Definition 12** *An object environment $\sigma_o$ over a pre-model $\mathcal{P}$ is a function from the set of object variables to the objects of $|\mathcal{P}|$.*

We are now ready to define the value of an object term in a pre-model.

**Definition 13** *The value $\mathcal{V}_\mathcal{P}$ of object terms under the environment $\sigma_o$ is defined by:*

- $\mathcal{V}_\mathcal{P}(a)(\sigma_o) = \sigma_o(a)$ *for every object variable* $a$,
- $\mathcal{V}_\mathcal{P}(c)(\sigma_o) = c^\mathcal{P}$ *for every constant symbol* $c \in C_{obj}$.

In the next definition we define an environment for both relational and object variables.

**Definition 14** *An environment $\sigma = (\sigma_o, \sigma_r)$ over a pre-model $\mathcal{P}$ is a pair of functions so that $\sigma_o$ is an object environment over $\mathcal{P}$ and $\sigma_r$ maps each relational variable $r : s_1 \to s_2$ to a relation $\sigma_r(r) : \mathcal{V}_\mathcal{P}(s_1)(\sigma_o) \to \mathcal{V}_\mathcal{P}(s_2)(\sigma_o)$.*

In the following $\sigma_o$ and $\sigma_r$ will always refer to the object and relational part of an environment $\sigma$, respectively. Similarly, we will write $\sigma(a)$ instead of $\sigma_o(a)$ for object variables $a$, and $\sigma(r : s_1 \to s_2)$ instead of $\sigma_r(r : s_1 \to s_2)$ for relational variables $r : s_1 \to s_2$. Storing a new value for a variable in an environment is called update. Such an update of an environment yields again an environment and is defined as follows:

**Definition 15** *The update $\sigma[A/a]$ resp. $\sigma[R/r : s_1 \to s_2]$ of $\sigma$ at the object variable $a$ resp. at the relation variable $r : s_1 \to s_2$ with the object $A$ resp. with the relation $R : \sigma(s_1) \to \sigma(s_2)$ is defined by:*

$$\sigma[A/a](b) = \begin{cases} \sigma(b) & \text{iff } a \neq b, \\ A & \text{iff } a = b, \end{cases}$$

$$\sigma[A/a](r : s_1 \to s_2) = \begin{cases} \sigma(r : s_1 \to s_2) & \text{iff } s_1 \neq a \text{ and } s_2 \neq a, \\ R : \sigma(s_1) \to \sigma(s_2) & \text{iff } s_1 = a \text{ or } s_2 = a \end{cases}$$

*For an arbitrary relation $R : \sigma(s_1) \to \sigma(s_2)$,*

$$\sigma[R/r : s_1 \to s_2](a) = \sigma(a),$$

$$\sigma[R/r : s_1 \to s_2](q : s_1 \to s_2) = \begin{cases} \sigma(q : s_1 \to s_2) & \text{iff } r : s_1' \to s_2' \neq s_1 \to s_2, \\ R & \text{iff } r : s_1' \to s_2' = q : s_1 \to s_2. \end{cases}$$

To ascribe meaning to all formulas, we need, besides a non empty allegory, an appropriate interpretation of each of the constant, function and predicate symbols.

**Definition 16** *A relational model $\mathcal{M}$ is a pre-model with the following data:*

1. *For each $c : s_1 \to s_2$ in $C_{rel}$ and environment $\sigma$ a constant $c_\sigma^{\mathcal{M}} : \sigma(s_1) \to \sigma(s_2)$ so that $\sigma(s_1) = \sigma'(s_1)$ and $\sigma(s_2) = \sigma'(s_2)$ implies $c_\sigma^{\mathcal{M}} = c_{\sigma'}^{\mathcal{M}}$,*

2. *For each function symbol $f : \{(t_1 \to s_1), ..., (t_n \to s_n)\} \to (t \to s)$ in $F$ and environment $\sigma$, a n-ary function $f_\sigma^{\mathcal{M}}$ which is mapping $|\mathcal{M}|[\sigma(t_1), \sigma(s_1)] \times ... \times |\mathcal{M}|[\sigma(t_n), \sigma(s_n)]$ to $|\mathcal{M}|[\sigma(t), \sigma(s)]$ so that $\sigma(t_1) = \sigma'(t_1), ..., \sigma(t_n) = \sigma'(t_n), \sigma(t) = \sigma'(t)$ and $\sigma(s_1) = \sigma'(s_1), ..., \sigma(s_n) = \sigma'(s_n), \sigma(s) = \sigma'(s)$ implies $f_\sigma^{\mathcal{M}} = f_{\sigma'}^{\mathcal{M}}$,*

3. *For each predicate symbol $p$ in $P$ with type $(t_1 \to s_1), ..., (t_n \to s_n)$ and environment $\sigma$, a subset $p_\sigma^{\mathcal{M}} \subseteq \{|\mathcal{M}|[\sigma(t_1), \sigma(s_1)] \times ... \times |\mathcal{M}|[\sigma(t_n), \sigma(s_n)]\}$ so that $\sigma(t_1) = \sigma'(t_1), ..., \sigma(t_n) = \sigma'(t_n), \sigma(t) = \sigma'(t)$ and $\sigma(s_1) = \sigma'(s_1), ..., \sigma(s_n) = \sigma'(s_n), \sigma(s) = \sigma'(s)$ implies $p_\sigma^{\mathcal{M}} = p_{\sigma'}^{\mathcal{M}}$.*

In the previous definition, an object environment would be sufficient because it is just needed to get the value of an object term. However, we defined it this way. Note that the value of an object term in a model $\mathcal{M}$ is the same as that in the pre-model $\mathcal{P}$ it contains, i.e., $\mathcal{V}_{\mathcal{M}}(s)(\sigma) = \mathcal{V}_{\mathcal{P}}(s)(\sigma_o)$.

Now we are ready to define the value of relational terms and the validity of formulas. Both definitions are done inductively on the structure of the language.

**Definition 17** *Let $\mathcal{M}$ be a relational model and $\sigma$ be an environment. The value $\mathcal{V}_{\mathcal{M}}$ of terms under the environment $\sigma$ is defined by:*

1. $\mathcal{V}_{\mathcal{M}}(r : s_1 \to s_2)(\sigma) = \sigma(r : s_1 \to s_2)$ *for every relational variable* $r : s_1 \to s_2$,
2. $\mathcal{V}_{\mathcal{M}}(c : s_1 \to s_2)(\sigma) = c_\sigma^{\mathcal{M}}$ *for every constant* $c : s_1 \to s_2$ *in* $C_{rel}$,
3. $\mathcal{V}_{\mathcal{M}}(f(t_1, ..., t_n))(\sigma) = f_\sigma^{\mathcal{M}}(\mathcal{V}_{\mathcal{M}}(t_1)(\sigma), ..., \mathcal{V}_{\mathcal{M}}(t_n)(\sigma))$
4. $\mathcal{V}_{\mathcal{M}}(\mathbb{I}_a)(\sigma) = \mathbb{I}_{\sigma(a)}$,
5. $\mathcal{V}_{\mathcal{M}}(t^\smile)(\sigma) = (\mathcal{V}_{\mathcal{M}}(t)(\sigma))^\smile$,
6. $\mathcal{V}_{\mathcal{M}}(t_1 \sqcap t_2)(\sigma) = \mathcal{V}_{\mathcal{M}}(t_1)(\sigma) \sqcap \mathcal{V}_{\mathcal{M}}(t_2)(\sigma)$,
7. $\mathcal{V}_{\mathcal{M}}(t_1; t_2)(\sigma) = \mathcal{V}_{\mathcal{M}}(t_1)(\sigma); \mathcal{V}_{\mathcal{M}}(t_2)(\sigma)$,

The next step is to define the validity of formulas.

**Definition 18** *Let $\mathcal{M}$ be a relational model, and $\sigma$ be an environment. The validity of a formula in $\mathcal{M}$ under $\sigma$ is defined inductively as follows:*

1. $\mathcal{M} \models_\sigma t_1 = t_2$ *iff* $\mathcal{V}_\mathcal{M}(t_1)(\sigma) = \mathcal{V}_\mathcal{M}(t_2)(\sigma)$,
2. $\mathcal{M} \models_\sigma p(t_1, \ldots, t_n)$ *iff* $(\mathcal{V}_\mathcal{M}(t_1)(\sigma), \ldots, \mathcal{V}_\mathcal{M}(t_n)(\sigma)) \in p_\sigma^\mathcal{M}$,
3. $\mathcal{M} \models_\sigma \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M} \models_\sigma \varphi_1$ *and* $\mathcal{M} \models_\sigma \varphi_2$,
4. $\mathcal{M} \models_\sigma \varphi_1 \vee \varphi_2$ *iff* $\mathcal{M} \models_\sigma \varphi_1$ *or* $\mathcal{M} \models_\sigma \varphi_2$,
5. $\mathcal{M} \models_\sigma \varphi_1 \to \varphi_2$ *iff* $\mathcal{M} \models_\sigma \neg\varphi_1$ *or* $\mathcal{M} \models_\sigma \varphi_2$,
6. $\mathcal{M} \models_\sigma \neg\varphi$ *iff* $\mathcal{M} \not\models_\sigma \varphi$
7. $\mathcal{M} \models_\sigma (\forall r : s_1 \to s_2)\varphi$ *iff* $\mathcal{M} \models_{\sigma[R/r:s_1 \to s_2]} \varphi$ *for all relations* $R : \sigma(s_1) \to \sigma(s_2)$,
8. $\mathcal{M} \models_\sigma (\forall a)\varphi$ *iff* $\mathcal{M} \models_{\sigma[A/a]} \varphi$ *for all objects* $A$,
9. $\mathcal{M} \models_\sigma (\exists r : s_1 \to s_2)\varphi$ *iff* $\mathcal{M} \models_{\sigma[R/r:s_1 \to s_2]} \varphi$ *for some relation* $R : \sigma(s_1) \to \sigma(s_2)$,
10. $\mathcal{M} \models_\sigma (\exists a)\varphi$ *iff* $\mathcal{M} \models_{\sigma[A/a]} \varphi$ *for some object* $A$.

Based on the previous definition we now introduce the validity of formulas in general.

**Definition 19** *Let $\mathcal{M}$ be a relational model, and $\varphi$ be a formula. Then:*

1. *$\varphi$ is called valid in the relational model $\mathcal{M}$, $\mathcal{M} \models \varphi$, iff $\mathcal{M} \models_\sigma \varphi$ for all environments $\sigma$.*
2. *$\varphi$ is called valid in the allegory $\mathcal{R}$, $\mathcal{R} \models \varphi$, iff $\mathcal{M} \models \varphi$ for all models $\mathcal{M}$ with $|\mathcal{M}| = \mathcal{R}$.*
3. *$\varphi$ is called valid, $\models \varphi$, iff $\mathcal{R} \models \varphi$ for all allegories $\mathcal{R}$.*
4. *$\varphi$ is said to follow from a sequence of formulas $\Gamma$, $\Gamma \models \varphi$, iff whenever $\mathcal{M} \models \psi$ for all $\psi \in \Gamma$, then $\mathcal{M} \models \varphi$ for all relational models $\mathcal{M}$ .*

If a formula $\varphi$ or a sequence of formulas $\Gamma$ is valid in a relational model $\mathcal{M}$, we will call $\mathcal{M}$ a model of $\varphi$ or $\Gamma$ respectively.

In the next lemma, we show that in order to check the validity of a formula we are just interested in variables which occur free.

**Lemma 20** *(Coincidence Lemma) Let $s$ be an object term, $t$ a relational term and $\psi$ a formula in the language of relational categories, $\mathcal{M}$ be a relational model and $\sigma_1$ and $\sigma_2$ environments over $\mathcal{M}$ so that $\sigma_1(a) = \sigma_2(a)$ for all free object variables in $s, t$ or $\psi$ and $\sigma_1(r : s_1 \to s_2) = \sigma_2(r : s_1 \to s_2)$ for all free relational variables $r$ in $t$ or $\psi$, respectively. Then:*

1. $\mathcal{V}_{\mathcal{M}}(s)(\sigma_1) = \mathcal{V}_{\mathcal{M}}(s)(\sigma_2)$.
2. $\mathcal{V}_{\mathcal{M}}(t)(\sigma_1) = \mathcal{V}_{\mathcal{M}}(t)(\sigma_2)$.
3. $\mathcal{M} \models_{\sigma_1} \psi$ *iff* $\mathcal{M} \models_{\sigma_2} \psi$.

**Proof.** All proofs except part 1 are shown by induction.

1. If $s = c$, i.e., $s$ is an object constant symbol

$$\mathcal{V}_{\mathcal{M}}(c)(\sigma_1) = c^{\mathcal{M}} = \mathcal{V}_{\mathcal{M}}(c)(\sigma_2).$$

   If $s = a$, i.e., $s$ is an object variable, we get

$$\mathcal{V}_{\mathcal{M}}(a)(\sigma_1) = \sigma_1(a) = \sigma_2(a) = \mathcal{V}_{\mathcal{M}}(a)(\sigma_2).$$

2. If $t = c$ and $c : s_1 \to s_2$ is a relational constant symbol

$$\begin{aligned} \mathcal{V}_{\mathcal{M}}(c : s_1 \to s_2)(\sigma_1) &= c^{\mathcal{M}}_{\sigma_{o_1}} \\ &= c^{\mathcal{M}}_{\sigma_{o_2}} && \text{by 16(3) since for all object} \\ & && \text{terms } \sigma_1(s) = \sigma_2(s) \text{ by 1} \\ &= \mathcal{V}_{\mathcal{M}}(c : s_1 \to s_2)(\sigma_2). \end{aligned}$$

   If $t = r$ and $r : s_1 \to s_2$ be a relational variable, we get

$$\begin{aligned} \mathcal{V}_{\mathcal{M}}(r : s_1 \to s_2)(\sigma_1) &= \sigma_{r_1}(r : s_1 \to s_2) \\ &= \sigma_{r_2}(r : s_1 \to s_2) = \mathcal{V}_{\mathcal{M}}(r : s_1 \to s_2)(\sigma_2). \end{aligned}$$

If $t = t_1^\smile$, we get

$$\begin{aligned}\mathcal{V}_\mathcal{M}(t_1^\smile)(\sigma_1) &= (\mathcal{V}_\mathcal{M}(t_1)(\sigma_1))^\smile \\ &= (\mathcal{V}_\mathcal{M}(t_1)(\sigma_2))^\smile && \text{by the I.H.} \\ &= \mathcal{V}_\mathcal{M}(t_1^\smile)(\sigma_2).\end{aligned}$$

The remaining cases are similar to the last one.

3. If $\psi = (t_1 = t_2)$, then we get

$$\begin{aligned}\mathcal{M} \models_{\sigma_1} t_1 = t_2 &\Leftrightarrow \mathcal{V}_\mathcal{M}(t_1)(\sigma_1) = \mathcal{V}_\mathcal{M}(t_2)(\sigma_1) \\ &\Leftrightarrow \mathcal{V}_\mathcal{M}(t_1)(\sigma_2) = \mathcal{V}_\mathcal{M}(t_2)(\sigma_2) && \text{by 2} \\ &\Leftrightarrow \mathcal{M} \models_{\sigma_2} t_1 = t_2.\end{aligned}$$

If $\psi = p(t_1, ..., t_n)$, then we get

$$\begin{aligned}\mathcal{M} \models_{\sigma_1} p(t_1, ..., t_n) &\Leftrightarrow (\mathcal{V}_\mathcal{M}(t_1)(\sigma_1), ..., \mathcal{V}_\mathcal{R}(t_n)(\sigma_1)) \in p^\mathcal{M} \\ &\Leftrightarrow (\mathcal{V}_\mathcal{M}(t_1)(\sigma_2), ..., \mathcal{V}_\mathcal{R}(t_n)(\sigma_2)) \in p^\mathcal{M} && \text{by 2} \\ &\Leftrightarrow \mathcal{M} \models_{\sigma_2} p(t_1, ..., t_n),\end{aligned}$$

If $\psi = \varphi_1 \wedge \varphi_2$, then we get

$$\begin{aligned}\mathcal{M} \models_{\sigma_1} \varphi_1 \wedge \varphi_2 &\Leftrightarrow \mathcal{M} \models_{\sigma_1} \varphi_1 \text{ and } \mathcal{M} \models_{\sigma_1} \varphi_2 \\ &\Leftrightarrow \mathcal{M} \models_{\sigma_2} \varphi_1 \text{ and } \mathcal{M} \models_{\sigma_2} \varphi_2 && \text{by I.H.} \\ &\Leftrightarrow \mathcal{M} \models_{\sigma_2} \varphi_1 \wedge \varphi_2.\end{aligned}$$

The cases in which $\psi$ is one of the formulas $\bot$, $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, or $\varphi_1 \to \varphi_2$ are similar to the previous case.

Assume $\psi = (Qr : s_1 \to s_2)\varphi$ with $Q \in \{\forall, \exists\}$. The free variables of $\varphi$ are the free variables of $\psi$ and the variables $s_1$, $s_2$ and $r$. Consequently, the environments $\sigma_1[R/r]$ and $\sigma_2[R/r]$ for an arbitrary relation $R : \sigma_1(s_1) \to \sigma_1(s_2) = \sigma_2(s_1) \to \sigma_2(s_2)$ coincide on all free variables in $\varphi$. We conclude

$$\begin{aligned}\mathcal{M} \models_{\sigma_1} \psi &\Leftrightarrow \mathcal{M} \models_{\sigma_1[R/r]} \varphi && \text{for all/some } R : \sigma_1(s_1) \to \sigma_1(s_2) \\ &\Leftrightarrow \mathcal{M} \models_{\sigma_2[R/r]} \varphi && \text{for all/some } R : \sigma_2(s_1) \to \sigma_2(s_2) \\ &\Leftrightarrow \mathcal{M} \models_{\sigma_2} \psi\end{aligned}$$

where the second equivalence is an application of the induction hypothesis.

The case $\psi = (Qa)\varphi$ with $Q \in \{\forall, \exists\}$ is similar to the previous case. $\square$

In the rest of this chapter we introduce the substitution of variables and the restrictions of substitution. Then we prove the Substitution Lemma.

**Definition 21** *Let $r : s_1 \to s_2$ be a relational variable resp. $a$ be an object variable, and $\varphi$ be a formula. A relational term $t : s_1 \to s_2$ resp. object term $s$ is called free for $r$ resp. $a$ in $\varphi$ iff no free occurrence of $r$ resp. $a$ is in a subformula $(Qb)\varphi'$ or $(Qq : s_1 \to s_2)\varphi'$, $Q \in \{\forall, \exists\}$, of $\varphi$ for an object variable $b$ or a relational variable $q$ occurring in $t$ resp. for an object variable $b$ occurring in $s$.*

Now we are ready to define the notion of substitution.

**Definition 22** *Let $a$ be an object variable, $r : s_1 \to s_2$ be a relational variable, $s$ be an object term, $t, t' : s_1 \to s_2$ be relational terms, and $\varphi$ be a formula.*

1. *By $t'[t/r]$ and $t'[s/a]$ we denote the result of replacing all occurrences of $t$ in $t'$ by $t$ resp. replacing all occurrences of $a$ in $t'$ by $s$.*

2. *If $t$ is free for $r$ in $\varphi$, then we denote by $\varphi[t/r]$ the result of replacing any free occurrence of $r$ in $\varphi$ by $t$.*

3. *If $s$ is free for $a$ in $\varphi$, then we denote by $\varphi[s/a]$ the result of replacing any free occurrence of $a$ in $\varphi$ by $s$.*

If we write $\varphi[t/r]$ we always assume that $t$ is free for $r$. It can always be achieved by renaming bounded variables.

The next lemma relates the notions of substitution and updating an environment.

**Lemma 23** *(Substitution Lemma) Let $a$ be an object variable, $r : s_1 \to s_2$ be a relational variable, $s, s'$ be object terms, $t, t' : s_1 \to s_2$, be relational terms, $\varphi$ be a formula, and $\mathcal{M}$ be a relational model.*

1. $\mathcal{V}_{\mathcal{M}}(s'[s/a])(\sigma) = \mathcal{V}_{\mathcal{M}}(s')(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a])$.

2. $\mathcal{V}_{\mathcal{M}}(t'[t/r])(\sigma) = \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])$.

3. $\mathcal{V}_{\mathcal{M}}(t'[s/a])(\sigma) = \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a])$.

4. $\mathcal{M} \models_\sigma \varphi[t/r]$ *iff* $\mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]} \varphi$.

5. $\mathcal{M} \models_\sigma \varphi[d/a]$ *iff* $\mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a]} \varphi$.

**Proof.** All assertions except 1 are shown by induction.

1. If $s' = c$ and $c$ is an object constant symbol, we get

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(c[s/a])(\sigma) &= \mathcal{V}_{\mathcal{M}}(c)(\sigma)\\ &= c^{\mathcal{M}}\\ &= \mathcal{V}_{\mathcal{M}}(c)(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a])\\ &= \mathcal{V}_{\mathcal{M}}(s')(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a]).\end{aligned}$$

   If $s' = a'$ and $a'$ is an object variable, we distinguish two cases. If $a = a'$, we get

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(s'[s/a])(\sigma) &= \mathcal{V}_{\mathcal{M}}(s)(\sigma)\\ &= \mathcal{V}_{\mathcal{M}}(a)(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a])\\ &= \mathcal{V}_{\mathcal{M}}(s')(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a]).\end{aligned}$$

   If $a \neq a'$, the environments $\sigma$ and $\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a]$ coincide on all variables in $s'$. We use Lemma 20(2) and conclude

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(s'[s/a])(\sigma) &= \mathcal{V}_{\mathcal{M}}(a')(\sigma)\\ &= \mathcal{V}_{\mathcal{M}}(a')(\sigma[\mathcal{V}_{\mathcal{M}}(s)/a])\\ &= \mathcal{V}_{\mathcal{M}}(s')(\sigma[\mathcal{V}_{\mathcal{M}}(s)/a]).\end{aligned}$$

2. If $t' = c$ and $c$ is a relational constant symbol, we get

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(c[t/r])(\sigma) &= \mathcal{V}_{\mathcal{M}}(c)(\sigma)\\ &= c_\sigma^{\mathcal{M}}\\ &= c_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]}^{\mathcal{M}}\\ &= \mathcal{V}_{\mathcal{M}}(c)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])\\ &= \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]).\end{aligned}$$

If $t' = r'$ and $r'$ is a relational variable, we distinguish two cases. If $r = r'$, we get

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(t'[t/r])(\sigma) &= \mathcal{V}_{\mathcal{M}}(t)(\sigma)\\ &= \mathcal{V}_{\mathcal{M}}(r)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])\\ &= \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]).\end{aligned}$$

If $r \neq r'$, the environments $\sigma$ and $\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]$ coincide on all variables in $t'$. We use Lemma 20(2) and conclude

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(t'[t/r])(\sigma) &= \mathcal{V}_{\mathcal{M}}(r')(\sigma)\\ &= \mathcal{V}_{\mathcal{M}}(r')(\sigma[\mathcal{V}_{\mathcal{M}}(t)/r])\\ &= \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(t)/r]).\end{aligned}$$

If $t' = f(t_1, \ldots, t_n)$, we immediately get

$$\begin{aligned}&\mathcal{V}_{\mathcal{M}}(t'[t/r])(\sigma)\\ &= f^{\mathcal{M}}(\mathcal{V}_{\mathcal{M}}(t_1[t/r])(\sigma), \ldots, \mathcal{V}_{\mathcal{M}}(t_n[t/r])(\sigma))\\ &= f^{\mathcal{M}}(\mathcal{V}_{\mathcal{M}}(t_1)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]), \ldots, \mathcal{V}_{\mathcal{M}}(t_2)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])) \text{ by I.H.}\\ &= \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])\end{aligned}$$

If $t' = t_1^{\smile}$, we get

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(t_1^{\smile}[t/r])(\sigma) &= (\mathcal{V}_{\mathcal{M}}(t_1[t/r])(\sigma))^{\smile}\\ &= (\mathcal{V}_{\mathcal{M}}(t_1)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]))^{\smile} && \text{by I.H.}\\ &= \mathcal{V}_{\mathcal{M}}(t_1^{\smile})(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])\\ &= \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])\end{aligned}$$

$t' = t_1; t_2$ and $t' = t_1 \sqcap t_2$ are similar to the previous case.

3. If $t' = r : a \to s_2$, we get

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}((r : a \to s_2)[s/a])(\sigma) &= \mathcal{V}_{\mathcal{M}}(r : s \to s_2)(\sigma)\\ &= \mathcal{V}_{\mathcal{M}}(r : a \to s_2)(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a])\\ &= \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a]).\end{aligned}$$

If $t' = r : s_2 \to a$, then the proof is similar to the previous case.

If $t' = r : s_1 \to s_2$, the environments $\sigma$ and $\sigma[\mathcal{V}_{\mathcal{M}}(s)(\sigma)/a]$ coincide on all variables in $t'$. We use Lemma 20(2) and conclude

$$\begin{aligned}\mathcal{V}_{\mathcal{M}}(t'[s/a])(\sigma) &= \mathcal{V}_{\mathcal{M}}(r)(\sigma)\\ &= \mathcal{V}_{\mathcal{M}}(r)(\sigma[\mathcal{V}_{\mathcal{M}}(s)/a])\\ &= \mathcal{V}_{\mathcal{M}}(t')(\sigma[\mathcal{V}_{\mathcal{M}}(s)/a]).\end{aligned}$$

The remaining cases are similar to (1),

4. For this part we distinguish several cases:

   (a) If $\varphi = (t_1 = t_2)$, then

$$\begin{aligned}&\mathcal{M} \models_\sigma \varphi[t/r]\\ &\Leftrightarrow \mathcal{M} \models_\sigma t_1[t/r] = t_2[t/r]\\ &\Leftrightarrow \mathcal{V}_{\mathcal{M}}(t_1[t/r])(\sigma) = \mathcal{V}_{\mathcal{M}}(t_2[t/r])(\sigma)\\ &\Leftrightarrow \mathcal{V}_{\mathcal{M}}(t_1)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]) = \mathcal{V}_{\mathcal{M}}(t_2)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]) \qquad \text{by 2}\\ &\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]} t_1 = t_2\\ &\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]} \varphi.\end{aligned}$$

   (b) If $\varphi = p(t_1, \ldots, t_n)$, then

$$\begin{aligned}&\mathcal{M} \models_\sigma \varphi[t/r]\\ &\Leftrightarrow \mathcal{M} \models_\sigma p(t_1[t/r], \ldots, t_n[t/r])\\ &\Leftrightarrow (\mathcal{V}_{\mathcal{M}}(t_1[t/r])(\sigma), ..., \mathcal{V}_{\mathcal{M}}(t_n[t/r])(\sigma)) \in p^{\mathcal{M}}\\ &\Leftrightarrow (\mathcal{V}_{\mathcal{M}}(t_1)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]), .., \mathcal{V}_{\mathcal{M}}(t_n)(\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r])) \in p^{\mathcal{M}} \quad \text{by 2}\\ &\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]} p(t_1, \ldots, t_n)\\ &\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]} \varphi.\end{aligned}$$

   (c) If $\varphi = \neg\varphi'$, then

$$\begin{aligned}&\mathcal{M} \models_\sigma \varphi[t/r]\\ &\Leftrightarrow \mathcal{M} \models_\sigma \neg\varphi'[t/r]\\ &\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]} \neg\varphi' \qquad \text{by I.H.}\\ &\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_{\mathcal{M}}(t)(\sigma)/r]} \varphi.\end{aligned}$$

(d) If $\varphi = \varphi_1 \wedge \varphi_2$, then

$$\begin{aligned}
&\mathcal{M} \models_\sigma \varphi[t/r] \\
&\Leftrightarrow \mathcal{M} \models_\sigma \varphi_1[t/r] \wedge \varphi_2[t/r] \\
&\Leftrightarrow \mathcal{M} \models_\sigma \varphi_1[t/r] \text{ and } \mathcal{M} \models_\sigma \varphi_2[t/r] \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_1 \text{ and } \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_2 \qquad \text{by I.H.} \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_1 \wedge \varphi_2 \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi.
\end{aligned}$$

(e) If $\varphi = \varphi_1 \vee \varphi_2$, then

$$\begin{aligned}
&\mathcal{M} \models_\sigma \varphi[t/r] \\
&\Leftrightarrow \mathcal{M} \models_\sigma \varphi_1[t/r] \vee \varphi_2[t/r] \\
&\Leftrightarrow \mathcal{M} \models_\sigma \varphi_1[t/r] \text{ or } \mathcal{M} \models_\sigma \varphi_2[t/r] \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_1 \text{ or } \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_2 \qquad \text{by I.H.} \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_1 \vee \varphi_2 \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi.
\end{aligned}$$

(f) If $\varphi = \varphi_1 \to \varphi_2$, then

$$\begin{aligned}
&\mathcal{M} \models_\sigma \varphi[t/r] \\
&\Leftrightarrow \mathcal{M} \models_\sigma \varphi_1[t/r] \to \varphi_2[t/r] \\
&\Leftrightarrow \mathcal{M} \models_\sigma \neg\varphi_1[t/r] \text{ or } \mathcal{M} \models_\sigma \varphi_2[t/r] \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \neg\varphi_1 \text{ or } \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_2 \qquad \text{by I.H.} \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi_1 \to \varphi_2 \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi.
\end{aligned}$$

(g) Assume $\varphi = Q(q : s_1 \to s_2)\varphi'$ with $Q \in \{\forall, \exists\}$. We distinguish two cases:

Case $r = q$:

$$\begin{aligned}
\mathcal{M} \models_\sigma \varphi[t/r] &\Leftrightarrow \mathcal{M} \models_\sigma \varphi && \text{since } r \text{ does not occur free in } \varphi \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi && \text{by Lemma 20(3) since } r \\
& && \text{does not occur free in } \varphi
\end{aligned}$$

Case $r \neq q$:

$$
\begin{aligned}
\mathcal{M} \models_\sigma \varphi[t/r] &\Leftrightarrow \mathcal{M} \models_\sigma Q(q : s_1 \to s_2)\varphi'[t/r] \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[R/q]} \varphi'[t/r] && \text{for all/some } R : \sigma(s_1) \to \sigma(s_2) \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[R/q][\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi' && \text{for all/some } R : \sigma(s_1) \to \sigma(s_2) \\
& && \text{by I.H.} \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r][R/q]} \varphi' && \text{for all/some } R : \sigma(s_1) \to \sigma(s_2) \\
& && \text{since } r \neq q \text{ and } q \text{ does not} \\
& && \text{occur in } t \text{ because } t \text{ is free} \\
& && \text{for } r \text{ in } \varphi \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{R}\mathcal{M}(t)(\sigma)/r]} Q(q : s_1 \to s_2)\varphi' \\
&\Leftrightarrow \mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{R}\mathcal{M}(t)(\sigma)/r]} \varphi.
\end{aligned}
$$

(h) Analogously to (3). □

# Chapter 4

# Calculus of Relational Categories

In this chapter we introduce the first-order logic calculus of relational categories. The calculus is formulated in a sequent style [5] but with exactly one formula on the right-hand side. In the first section we discus the inference rules of the calculus and then we show its soundness and completeness.

In general, a logical calculus uses proof rules to infer a conclusion from a finite set of premises. Suppose a sequence of formulas $\Gamma_1$ and a derivation $\Gamma_1 \vdash \psi_1$ is given. We start to apply a proof rule of the calculus to the derivation generating a new derivation $\Gamma_2 \vdash \psi_2$. In the next step, we apply a rule to the new derivation generating a new derivation $\Gamma_3 \vdash \psi_3$. Continuous application of the rules will finally end in the intended result $\Gamma \vdash \psi$, the conclusion. The derivation itself is actually a tree with the premises as leaves, applications of rules as nodes, and the conclusion as the root. The tree will be ended at axioms rules which don't need premises.

## 4.1 Inference Rules

Our calculus has three different types of rules; structural rules, which operate on the sequent of formula in a judgment, logical rules, which are concerned with the logical operations, and axioms rules, which represent the basic tautology of logic and the axioms of the theory of allegories.

**Definition 24** *Let $\Gamma$ be a sequence of formulas, then the rules in Figures 4.1, 4.2, and 4.3 constitute the formal calculus of allegories. We write $\Gamma \vdash \varphi$ to indicate that there is a derivation ending in that sequence.*

$$\frac{}{\varphi \vdash \varphi}\ \text{Axiom}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b)\mathbb{I}_a; r = r}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b) r; \mathbb{I}_b = r}$$

$$\frac{}{\vdash (\forall a_1)(\forall a_2)(\forall a_3)(\forall a_4)(\forall r : a_1 \to a_2)(\forall q : a_2 \to a_3)(\forall u : a_3 \to a_4)(r; q); u = r; (q; u)}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b) r \sqcap r = r}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b)(\forall q : a \to b)(\forall u : a \to b)(r \sqcap q) \sqcap u = r \sqcap (q \sqcap u)}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b)(r^\smile)^\smile = r}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b)(\forall q : a \to b)(r \sqcap q)^\smile = r^\smile \sqcap q^\smile}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b)(\forall q : b \to c)(r; q)^\smile = q^\smile; r^\smile}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b)(\forall q : b \to c)(\forall u : b \to c) r; (q \sqcap u) = r; (q \sqcap u) \sqcap r; q \sqcap r; u}$$

$$\frac{}{\vdash (\forall a)(\forall b)(\forall r : a \to b)(\forall q : b \to c)(\forall u : a \to c) r; q \sqcap u = r; (q \sqcap r^\smile; u) \sqcap r; q \sqcap u}$$

Figure 4.1: Axiom rules

Weakening rule $\dfrac{\Gamma \vdash \psi}{\Gamma, \varphi \vdash \psi}$ Weak

Contraction rule $\dfrac{\Gamma, \varphi, \varphi \vdash \psi}{\Gamma, \varphi \vdash \psi}$ Cont.

Permutation rule $\dfrac{\Gamma, \varphi_2, \varphi_1 \vdash \psi}{\Gamma, \varphi_1, \varphi_2 \vdash \psi}$ Perm.

Cut rule $\dfrac{\Gamma \vdash \varphi \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}$ Cut

Figure 4.2: Structural rules

| left logical rules | right logical rules |
|---|---|
| $\dfrac{\Gamma \vdash t_1 = t_2 \quad \Gamma \vdash \psi[t_1/r]}{\Gamma \vdash \psi[t_2/r]}$ =L | $\dfrac{}{\vdash t = t}$ =R |
| $\dfrac{\Gamma, \varphi_1, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \wedge \varphi_2 \vdash \psi}$ $\wedge$L | $\dfrac{\Gamma \vdash \varphi_1 \quad \Gamma \vdash \varphi_2}{\Gamma \vdash \varphi_1 \wedge \varphi_2}$ $\wedge$R |
| $\dfrac{\Gamma, \varphi_1 \vdash \psi \quad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \vee \varphi_2 \vdash \psi}$ $\vee$L | $\dfrac{\Gamma \vdash \varphi_1}{\Gamma \vdash \varphi_1 \vee \varphi_2}$ $\vee$R $\quad$ $\dfrac{\Gamma \vdash \varphi_2}{\Gamma \vdash \varphi_1 \vee \varphi_2}$ $\vee$R |
| $\dfrac{\Gamma \vdash \varphi_1 \quad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \to \varphi_2 \vdash \psi}$ $\to$L | $\dfrac{\Gamma, \varphi_1 \vdash \varphi_2}{\Gamma \vdash \varphi_1 \to \varphi_2}$ $\to$R |
| $\dfrac{\Gamma \vdash \varphi}{\Gamma, \neg\varphi \vdash \psi}$ $\neg$L | $\dfrac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi}$ $\neg$R |

$$\frac{\Gamma, \varphi[t/r] \vdash \psi}{\Gamma, (\forall r : s_1 \to s_2)\varphi \vdash \psi}\ \forall\text{L (rel)} \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash (\forall r : s_1 \to s_2)\varphi}\ \forall\text{R (rel)}$$

If $r$ does not occur free in any formula of $\Gamma$

$$\frac{\Gamma, \varphi[s/a] \vdash \psi}{\Gamma, (\forall a)\varphi \vdash \psi}\ \forall\text{L (obj)} \qquad \frac{\Gamma \vdash \varphi}{\Gamma \vdash (\forall a)\varphi}\ \forall\text{R (obj)}$$

If $a$ does not occur free in any formula of $\Gamma$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma, (\exists r : s_1 \to s_2)\varphi \vdash \psi}\ \exists\text{L (rel)}$$

If $r$ does not occur free in any formula of $\Gamma$ and in $\psi$

$$\frac{\Gamma \vdash \varphi[t/r]}{\Gamma \vdash (\exists r : s_1 \to s_2)\varphi}\ \exists\text{R (rel)}$$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma, (\exists a)\varphi \vdash \psi}\ \exists\text{L (obj)}$$

If $a$ does not occur free in any formula of $\Gamma$ and in $\psi$

$$\frac{\Gamma \vdash \varphi[s/a]}{\Gamma \vdash (\exists a)\varphi}\ \exists\text{R (obj)}$$

$$\frac{\Gamma, \neg\varphi \vdash \bot}{\Gamma \vdash \varphi}\ \text{PBC}$$

Figure 4.3: Logical rules

Figure 4.4 shows an simple example of a derivation tree.

$$\dfrac{\varphi \vdash \varphi \qquad \dfrac{\psi \vdash \psi}{\varphi, \psi \vdash \psi}\ \text{Weak}}{\varphi, \varphi \Rightarrow \psi \vdash \psi}\ {\to}\text{L}$$

Figure 4.4: Example Derivation

Note that for the =L rule, we have chosen this version, which does not really look like a left rule, because it seems more convenient to use. In order to have the equation

on the left hand side it could use just the right assumption and the conclusion with the equation added to gamma.

## 4.2 Soundness Proof of Calculus

Soundness of a deductive system is the property that whatever can be derived is also valid, i.e., that $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$. This property is the very least one would require from any logical system. In the next lemma, we show the soundness of our calculus.

**Theorem 25 (Soundness)** *Let $\Gamma$ be a sequence of relational formulas and $\psi$ be a relational formula. If $\Gamma \vdash \psi$ is valid, then $\Gamma \models \psi$ holds.*

**Proof.** The proof is done by induction on the derivation $\Gamma \vdash \psi$.

**Base cases:** If $\Gamma = \varphi$ and $\psi = \varphi$ then the proof is just a premise. We need to show for all relational models $\mathcal{M}$ and environment $\sigma$ if $\mathcal{M} \models_\sigma \varphi$ holds, then $\mathcal{M} \models_\sigma \psi$. Since $\psi = \varphi$ that is trivial.

If $\Gamma$ is empty and $\psi = (\forall a)(\forall b)(\forall r : a \to b)\mathbb{I}_a; r = r$ then we need to show for all relational models $\mathcal{M}$ and environment $\sigma$, $\mathcal{M} \models_\sigma \psi$ holds. Since $|\mathcal{M}|$ is an allegory then $\mathcal{M} \models_{\sigma[A/a,B/b,R/r:a\to b]} \mathbb{I}_A; R = R$ for all objects $A, B$ and relations $R : A \to B$. Using the definition of $\models$ we get $\mathcal{M} \models_\sigma (\forall a)(\forall b)(\forall r : a \to b)\mathbb{I}_a; r = r$.

The remaining axiom rules can be shown analogously.

**Weak** : In this case $\Gamma = \Gamma', \varphi$ for some sequence of formulas $\Gamma'$ and we have a derivation $\Gamma' \vdash \psi$. Now assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \varphi$ and $\mathcal{M} \models_\sigma \Gamma'$ holds. Based on the induction hypothesis $\mathcal{M} \models_\sigma \Gamma'$ implies $\mathcal{M} \models_\sigma \psi$.

**Cont.** : In this case $\Gamma = \Gamma', \varphi$ and we have a derivation $\Gamma', \varphi, \varphi \vdash \psi$. Now assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \varphi$ and $\mathcal{M} \models_\sigma \Gamma'$ holds. By the induction hypothesis we conclude $\mathcal{M} \models_\sigma \psi$, and, hence $\Gamma', \varphi \models_\sigma \psi$.

**Perm.** : In this case $\Gamma = \Gamma', \varphi_2, \varphi_1$ and we have a derivation $\Gamma', \varphi_1, \varphi_2 \vdash \psi$. Now assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \varphi_1$, $\mathcal{M} \models_\sigma \varphi_2$ and $\mathcal{M} \models_\sigma \Gamma'$ holds. By the induction hypothesis we conclude $\mathcal{M} \models_\sigma \psi$, and, hence $\Gamma', \varphi_2, \varphi_1 \models_\sigma \psi$.

**Cut** : In this case we have derivations $\Gamma \vdash \varphi$ and $\Gamma, \varphi \vdash \psi$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$ holds. We get from the induction hypothesis that $\mathcal{M} \models_\sigma \varphi$ and, again by using the induction hypothesis we conclude $\mathcal{M} \models_\sigma \psi$.

$=$**L** : In this case we have derivations $\Gamma \vdash t_1 = t_2$ and $\Gamma \vdash \psi[t_1/r]$. Now assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$ holds. By the induction hypothesis we get $\Gamma \models_\sigma \psi[t_1/r]$. Using Lemma 23(4) we have $\Gamma \models_{\sigma[\mathcal{V}_\mathcal{M}(t_1)(\sigma)/r]} \psi$. Again by the induction hypothesis we get $\Gamma \models_\sigma t_1 = t_2$ which implies that $\mathcal{V}_\mathcal{M}(t_1)(\sigma) = \mathcal{V}_\mathcal{M}(t_2)(\sigma)$. Hence, $\Gamma \models_{\sigma[\mathcal{V}_\mathcal{M}(t_2)(\sigma)/r]} \psi$ and by using Lemma 23(4) we conclude $\Gamma \models_\sigma \psi[t_2/r]$.

$\wedge$**L** : In this case $\Gamma = \Gamma', \varphi_1 \wedge \varphi_2$ and we have a derivation $\Gamma', \varphi_1, \varphi_2 \vdash \psi$. Now assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma'$ and $\mathcal{M} \models_\sigma \varphi_1 \wedge \varphi_2$ holds. From the definition of $\models$ we get $\mathcal{M} \models_\sigma \varphi_1 \wedge \varphi_2$ iff $\mathcal{M} \models_\sigma \varphi_1$ and $\mathcal{M} \models_\sigma \varphi_2$. Which implies $\mathcal{M} \models_\sigma \Gamma'$, $\mathcal{M} \models_\sigma \varphi_1$ and $\mathcal{M} \models_\sigma \varphi_2$ holds. By the induction hypothesis we conclude $\mathcal{M} \models_\sigma \psi$, and, hence $\Gamma', \varphi_1 \wedge \varphi_2 \models_\sigma \psi$.

$\vee$**L** : In this case $\Gamma = \Gamma', \varphi_1 \vee \varphi_2$ and we have derivations $\Gamma', \varphi_1 \vdash \psi$ and $\Gamma', \varphi_2 \vdash \psi$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma'$ and $\mathcal{M} \models_\sigma \varphi_1 \vee \varphi_2$ holds. By the definition of $\models$ we get $\mathcal{M} \models_\sigma \varphi_1$ or $\mathcal{M} \models_\sigma \varphi_2$. In the first case we conclude $\mathcal{M}$ satisfies $\Gamma', \varphi_1$. Using induction hypothesis we conclude $\mathcal{M} \models_\sigma \psi$. If $\mathcal{M} \models_\sigma \varphi_2$ we conclude $\mathcal{M} \models_\sigma \psi$ analogously.

$\rightarrow$**L** : In this case $\Gamma = \Gamma', \varphi_1 \rightarrow \varphi_2$ and we have derivations $\Gamma' \vdash \varphi_1$ and $\Gamma', \varphi_2 \vdash \psi$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma'$ and $\mathcal{M} \models_\sigma \varphi_1 \rightarrow \varphi_2$ holds. From the definition of $\models$ we get $\mathcal{M} \models_\sigma \varphi_1 \rightarrow \varphi_2$ iff $\mathcal{M} \models_\sigma \varphi_2$ or $\mathcal{M} \not\models_\sigma \varphi_1$. We distinguish two cases:

Case $\mathcal{M} \models_\sigma \varphi_2$ : We know that $\mathcal{M} \models_\sigma \Gamma'$, and by induction hypothesis we conclude $\mathcal{M} \models_\sigma \psi$.

Case $\mathcal{M} \not\models_\sigma \varphi_1$: We have $\mathcal{M} \models_\sigma \Gamma'$, which is implies (based on induction hypothesis) that $\mathcal{M} \models_\sigma \varphi_1$. Hence this case is impossible.

$\neg$**L** In this case $\Gamma = \Gamma', \neg\varphi$ and we have a derivation $\Gamma' \vdash \varphi$. Now assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma'$ and $\mathcal{M} \models_\sigma \neg\varphi$ holds and from the induction hypothesis we get $\mathcal{M} \models_\sigma \varphi$. This is a contradiction so that such a $\mathcal{M}$ does not exist. Hence, $\Gamma', \neg\varphi \models \psi$ is always true.

$\forall$**L (rel)** : In this case $\Gamma = \Gamma', (\forall r : s_1 \to s_2)\varphi$ where $r$ is a relational variable and $s_1$ and $s_2$ are object terms and $\Gamma', \varphi[t/r] \vdash \psi$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma'$ and $\mathcal{M} \models_\sigma (\forall r : s_1 \to s_2)\varphi$ holds. Using the definition of $\models$ we get $\mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi$. By Lemma 23(4) the latter is equivalent to $\mathcal{M} \models_\sigma \varphi[t/r]$. Hence, from the induction hypothesis we conclude $\mathcal{M} \models_\sigma \psi$.

$\forall$**L (obj)** : Analogously to $\forall$L (rel).

$\exists$**L (rel)** : In this case $\Gamma = \Gamma', (\exists r : s_1 \to s_2)\varphi$ and $\Gamma', \varphi \vdash \psi$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma'$ and $\mathcal{M} \models_\sigma (\exists r : s_1 \to s_2)\varphi$ holds. Using the definition we get $\mathcal{M} \models_{\sigma[T/r:s_1 \to s_2]} \varphi$ for some relation $T : \sigma(s_1) \to \sigma(s_2)$. According to Lemma 20(3), since $r$ does not occur free in any formula of $\Gamma'$ we have $\mathcal{M} \models_{\sigma[T/r:s_1 \to s_2]} \Gamma'$. By the induction hypothesis we get $\mathcal{M} \models_{\sigma[T/r:s_1 \to s_2]} \psi$. Again by Lemma 20(3) we conclude $\mathcal{M} \models_\sigma \psi$ since $r$ does not occur free in $\psi$.

$\exists$**L (obj)** : Analogously to $\exists$L (rel).

$=$**R** : It is trivial.

$\wedge$**R** : In this case $\psi = \varphi_1 \wedge \varphi_2$ and we have derivations $\Gamma \vdash \varphi_1$ and $\Gamma \vdash \varphi_2$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$ holds. By the induction hypothesis we conclude $\mathcal{M} \models_\sigma \varphi_1$ and $\mathcal{M} \models_\sigma \varphi_2$ which implies $\mathcal{M} \models_\sigma \varphi_1 \wedge \varphi_2$, and, hence $\mathcal{M} \models_\sigma \psi$.

$\vee$**R** : In this case $\psi = \varphi_1 \vee \varphi_2$ and we have a derivation $\Gamma \vdash \varphi_1$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$ holds. From the induction hypothesis we get $\mathcal{M} \models_\sigma \varphi_1$, and, by the definition of $\models$ we conclude $\mathcal{M} \models_\sigma \varphi_1 \vee \varphi_2$.

$\to$**R** : In this case $\psi = \varphi_1 \to \varphi_2$ and we have a derivation $\Gamma, \varphi_1 \vdash \varphi_2$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$ holds. If $\mathcal{M} \models_\sigma \varphi_1$ holds then from the induction hypothesis we get $\mathcal{M} \models_\sigma \varphi_2$. By the definition of $\models$ we conclude $\mathcal{M} \models_\sigma \varphi_1 \to \varphi_2$. If $\mathcal{M} \not\models_\sigma \varphi_1$ then by the definition of $\models$, $\mathcal{M} \models_\sigma \varphi_1 \to \varphi_2$ is always hold.

$\neg$**R** : In this case $\psi = \neg\varphi$ and we have a derivation $\Gamma, \varphi \vdash \bot$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$ holds. If $\mathcal{M}$ satisfies $\varphi$ we conclude $\mathcal{M} \models_\sigma \bot$ from the induction hypothesis. That is a contradiction, hence, we conclude $\mathcal{M} \models_\sigma \neg\varphi$.

$\forall$**R (rel)** : In this case $\psi = (\forall r : s_1 \to s_2)\varphi$ and we have $\Gamma \vdash \varphi$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$. Since $r$ does not occur free in any formula of $\Gamma$, by Lemma 20(3) we get $\mathcal{M} \models_{\sigma[T/r:s_1 \to s_2]} \Gamma$ for all relations $T : \sigma(s_1) \to \sigma(s_2)$. By the induction hypothesis we get $\mathcal{M} \models_{\sigma[T/r:s_1 \to s_2]} \varphi$ for all relations $T : \sigma(s_1) \to \sigma(s_2)$. Using the definition we conclude $\mathcal{M} \models_\sigma (\forall r : s_1 \to s_2)\varphi$.

$\forall$**R (obj)** : Analogously to $\forall$R (rel).

$\exists$**R (rel)** : In this case $\psi = \exists r : s_1 \to s_2 : \varphi$ and we have $\Gamma \vdash \varphi[t/r]$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$. By the induction hypothesis we get $\mathcal{M} \models_\sigma \varphi[t/r]$. By Lemma 23(4) the latter is equivalent to $\mathcal{M} \models_{\sigma[\mathcal{V}_\mathcal{M}(t)(\sigma)/r]} \varphi$. Hence, we get from the definition $\mathcal{M} \models_\sigma (\exists r : s_1 \to s_2)\varphi$.

$\exists$**R (obj)** : Analogously to $\exists$R (rel).

**PBC** : In this case we have a derivation $\Gamma, \neg\psi \vdash \bot$. Assume $\mathcal{M}$ is a relational model and $\sigma$ an environment so that $\mathcal{M} \models_\sigma \Gamma$ holds. If $\mathcal{M} \models_\sigma \neg\psi$ we conclude $\mathcal{M} \models_\sigma \bot$ from the induction hypothesis. That is a contradiction, hence, we conclude $\mathcal{M} \models_\sigma \psi$.

This completes the proof. □

## 4.3 Completeness Proof of Calculus

A calculus is complete if for every set of premises $\Gamma$, any formula which semantically follows from $\Gamma$ is derivable from $\Gamma$, i.e., if $\Gamma \models \psi$ holds then $\Gamma \vdash \psi$ is valid [10]. Our next goal is to show completeness of the calculus. First, we want to get rid of premises in a proof.

**Lemma 26** *Let $\varphi_1, \ldots, \varphi_n$ and $\psi$ be formulas. Then we have:*

1. $\varphi_1, \ldots, \varphi_n \models \psi$ *iff* $\models \varphi_1 \to (\varphi_2 \to (\varphi_3 \to (\ldots(\varphi_n \to \psi)\ldots)))$.
2. $\varphi_1, \ldots, \varphi_n \vdash \psi$ *iff* $\vdash \varphi_1 \to (\varphi_2 \to (\varphi_3 \to (\ldots(\varphi_n \to \psi)\ldots)))$.

**Proof.** In this proof we will denote the formula $\varphi_1 \to (\varphi_2 \to (\varphi_3 \to (\ldots(\varphi_n \to \psi)\ldots)))$ by $\chi$.

1. $\Rightarrow$: Assume $\mathcal{M}$ is a relational model so that $\mathcal{M} \models \varphi_1, \ldots, \mathcal{M} \models \varphi_n$. we conclude $\mathcal{M} \models \psi$, and, hence, $\mathcal{M} \models \chi$. If there is an $i$ with $\mathcal{M} \not\models \varphi_i$ we immediately conclude $\mathcal{M} \models \chi$.

   $\Leftarrow$: Assume $\mathcal{M}$ is a relational model so that $\mathcal{M} \models \varphi_i$ for all $i \in \{1, \ldots, n\}$. Since $\chi$ is valid, we have $\mathcal{M} \models \chi$. From the fact that $\chi$ is a chain of implications we conclude $\mathcal{M} \models \psi$, and, hence, $\varphi_1, \ldots, \varphi_n \models \psi$.

2. $\Rightarrow$: Assume there is a derivation $\varphi_1, \ldots, \varphi_n \vdash \psi$. By applying the rule $\rightarrow$R $n$ times we get a derivation $\vdash \chi$.

   $\Leftarrow$: Assume there is a derivation $\vdash \chi$ and $\chi = \varphi_1 \rightarrow \chi'$. We give following derivation:

$$\dfrac{\dfrac{\vdash \varphi_1 \rightarrow \chi'}{\varphi_1 \vdash \varphi_1 \rightarrow \chi'}\,\text{Weak} \quad \dfrac{\varphi_1 \vdash \varphi_1 \quad \dfrac{\chi' \vdash \chi'}{\varphi_1, \chi' \vdash \varphi_1}\,\text{Weak}}{\varphi_1, \varphi_1 \rightarrow \chi' \vdash \chi'}\,\rightarrow\text{L}}{\varphi_1 \vdash \chi'}\,\text{Cut}$$

   By applying similar derivations $n$-times with the formula $\varphi_i$ in the $i$th application we get a derivation $\varphi_1, \ldots, \varphi_n \vdash \psi$. $\square$

The next step is to take care of the free variables in a formula.

**Lemma 27** *Let $\varphi$ be a formula. Then*

1. $\mathcal{M} \models \varphi$ *iff* $\mathcal{M} \models (\forall r : s_1 \rightarrow s_2)\varphi$.
2. $\models \varphi$ *iff* $\models (\forall r : s_1 \rightarrow s_2)\varphi$.
3. $\vdash \varphi$ *iff* $\vdash (\forall r : s_1 \rightarrow s_2)\varphi$.
4. $\mathcal{M} \models \varphi$ *iff* $\mathcal{M} \models (\forall a)\varphi$.
5. $\models \varphi$ *iff* $\models (\forall a)\varphi$.
6. $\vdash \varphi$ *iff* $\vdash (\forall a)\varphi$.

**Proof.**

1. $\Rightarrow$: Let $\sigma$ be an arbitrary environment. We get $\mathcal{M} \models_{\sigma[R/r:s_1 \to s_2]} \varphi$ by the assumption for all $R : \sigma(s_1) \to \sigma(s_1)$, hence, $\mathcal{M} \models_\sigma (\forall r : s_1 \to s_2)\varphi$.

   $\Leftarrow$: Let $\sigma$ be an arbitrary environment. Then $\sigma = \sigma[\sigma(x)/x]$. By the assumption we have $\mathcal{M} \models (\forall r : s_1 \to s_2)\varphi[\sigma]$, and, hence, $\mathcal{M} \models_{\sigma[\sigma(r)/r]} \varphi$. We conclude $\mathcal{M} \models_\sigma \varphi$.

2. This follows immediately from (1).

3. $\Rightarrow$: To the derivation $\vdash \varphi$ we apply the rule $\forall$L (rel) to get a derivation $\vdash (\forall r : s_1 \to s_2)\varphi$. The variable condition is satisfied since the derivation has no premises.

   $\Leftarrow$: We give a derivation for this implication:

$$\dfrac{\vdash (\forall r : s_1 \to s_2)\varphi \quad \dfrac{\varphi \vdash \varphi}{(\forall r : s_1 \to s_2)\varphi \vdash \varphi}\ \forall\text{L (rel)}}{\vdash \varphi}\ \text{Cut}$$

   The cases (4),(5) and (6) are similar to (1),(2) and (3). □

Since every formula just contains finitely many free variables we may close a formula by adding universal quantifiers, e.g., if $\varphi$ is a formula and has free (relational or object) variables $x_1, \ldots, x_n$ then $(\forall x_1, \ldots, x_n)\varphi$ is a closed formula.

**Lemma 28** *The following statements are equivalent:*

1. *The calculus is complete, i.e., $\varphi_1, \ldots, \varphi_n \models \psi$ implies $\varphi_1, \ldots, \varphi_n \vdash \psi$ for all formulas $\varphi_1, \ldots, \varphi_n$ and $\psi$.*

2. *$\models \varphi$ implies $\vdash \varphi$ for all closed formulas $\varphi$.*

**Proof.**

1. $\Rightarrow$ 2.**:** This implication is trivial.

2. $\Rightarrow$ 1.**:** Assume $\varphi_1, \ldots, \varphi_n \models \psi$ and let $\varphi = \varphi_1 \to (\varphi_2 \to (\varphi_3 \to (\ldots(\varphi_n \to \psi)\ldots)))$. By Lemma 26(1) we have $\models \varphi$. Now, we apply Lemma 27(2) as often as we have free variables to conclude $\models \forall\varphi$. (2) implies $\vdash \forall\varphi$, and, hence $\vdash \varphi$ using Lemma 27(3). Finally, Lemma 26(2) shows $\varphi_1, \ldots, \varphi_n \vdash \psi$. □

In the followings, we are going to prove (2) instead of (1).

**Definition 29** *A set of closed formulas $T$ is called a theory. A theory is called consistent iff $T \nvdash \bot$. It is called inconsistent iff it is not consistent, i.e., if $T \vdash \bot$.*

In the next lemma, we want to relate derivations with the consistency of a theory.

**Lemma 30** *Let $T$ be a theory, and $\varphi$ be a closed formula. Then $T \vdash \varphi$ iff $T \cup \{\neg\varphi\}$ is inconsistent.*

**Proof.** Let $\Gamma$ be a sequence of all formulas in $T$, then we give following derivations:
$\Rightarrow$: We have derivation $\Gamma \vdash \varphi$, hence we can conclude:

$$\frac{\Gamma \vdash \varphi}{\Gamma, \neg\varphi \vdash \bot}\ \neg\mathrm{L}$$

$\Leftarrow$: This time we have a derivation $\Gamma, \neg\varphi \vdash \bot$, So we get:

$$\frac{\Gamma, \neg\varphi \vdash \bot}{\Gamma \vdash \varphi}\ \mathrm{PBC}$$

□

In the next lemma, we provide the version of the completeness theorem we are going to prove. Notice that (1) actually implies completeness.

**Lemma 31** *The following statements are equivalent:*

1. *$T \models \varphi$ implies $T \vdash \varphi$ for all closed formulas $\varphi$ and consistent theories $T$.*
2. *Every consistent theory has a relational model.*

**Proof.**

1. $\Rightarrow$ 2.**:** $T$ is consistent, i.e., $T \nvdash \bot$, and by (1) we have $T \not\models \bot$. This implies that there is a relational model $\mathcal{M}$ so that $\mathcal{M} \models \psi$ for all $\psi \in T$ and $\mathcal{M} \not\models \bot$. Consequently, $\mathcal{M}$ is a relational model for $T$.

2. $\Rightarrow$ 1.**:** Assume $T \nvdash \varphi$. Then we have to show that $T \not\models \varphi$. By Lemma 30 the theory $T \cup \{\neg\varphi\}$ is consistent. From (2) we conclude that $T \cup \{\neg\varphi\}$ has a relational model $\mathcal{M}$, i.e., $\mathcal{M} \models \psi$ for all $\psi \in T$ and $\mathcal{M} \models \neg\varphi$, i.e., $T \not\models \varphi$. □

In the following we are going to prove the statement in Lemma 31(2). We are facing the problem that we have to construct a model for a given theory. The key idea is to basically use the syntactic material itself, i.e., the universe is formed by the variable-free or closed terms.

**Lemma 32** *Let $T$ be a theory, and define the relation $\sim_{s_1,s_2}$ on closed terms $t_1$ and $t_2$ with the same type $s_1 \to s_2$ by $t_1 \sim_{s_1,s_2} t_2$ iff $T \vdash t_1 = t_2$. Then $\sim_{s_1,s_2}$ is an equivalence relation, and we have*

1. *If $t \sim_{s_1,s_2} t'$ then $t^\smile \sim_{s_2,s_1} t'^\smile$,*
2. *If $t_1 \sim_{s_1,s_2} t'_1$ and $t_2 \sim_{s_1,s_2} t'_2$ then $t_1 \sqcap t_2 \sim_{s_1,s_2} t'_1 \sqcap t'_2$,*
3. *If $t_1 \sim_{s_1,s_2} t'_1$ and $t_2 \sim_{s_2,s_3} t'_2$ then $t_1; t_2 \sim_{s_1,s_3} t'_1; t'_2$,*
4. *If $t_1 \sim_{s_1,s'_1} t'_1, \ldots, t_n \sim_{s_n,s'_n} t'_n$ then $f(t_1, \ldots, t_n) \sim_{s,s'} f(t'_1, \ldots, t'_n)$ for all $n$-ary function symbols of type $s \to s'$, and*
5. *$T \vdash p(t_1, \ldots, t_n)$ iff $T \vdash p(t'_1, \ldots, t'_n)$ for all $n$-ary predicate symbols $p$.*

**Proof.** The rule $\vdash t = t$ shows that $\sim_{s_1,s_2}$ is reflexive. Assume $t_1 \sim_{s_1,s_2} t_2$ and $t_2 \sim_{s_1,s_2} t_3$. Then there are derivations $T \vdash t_1 = t_2$ and $T \vdash t_2 = t_3$. We get

$$\frac{T \vdash t_2 = t_3 \quad T \vdash t_1 = t_2}{T \vdash t_1 = t_3} \,{=}\mathrm{L}$$

and, hence, $t_1 \sim_{s_1,s_2} t_3$, i.e., $\sim$ is transitive. Assume $t_1 \sim_{s_1,s_2} t_2$, i.e., there is a derivation $T \vdash t_1 = t_2$. We get

$$\frac{T \vdash t_1 = t_2 \quad \dfrac{}{\vdash t_1 = t_1}\,{=}\mathrm{R}}{T \vdash t_2 = t_1}\,{=}\mathrm{L}$$

and, hence, $t_2 \sim_{s_1,s_2} t_1$, i.e., $\sim_{s_1,s_2}$ is symmetric.

Now, assume $t_1 \sim_{s_1,s'_1} t'_1, \ldots, t_n \sim_{s_n,s'_n} t'_n$, i.e., there are derivations $T \vdash t_1 = t'_1, \ldots, T \vdash t_n = t'_n$. We get

$$\frac{t_1 = t'_1 \quad \dfrac{}{f(t_1, t_2, \ldots, t_n) = f(t_1, t_2, \ldots, t_n)}\,{=}\mathrm{R}}{f(t_1, t_2, \ldots, t_n) = f(t'_1, t_2, \ldots, t_n)}\,{=}\mathrm{L}$$

$n - 1$ additional applications of the rule $=$L shows $T \vdash f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n)$. Properties (1),(2),(3) and (5) are shown analogously. □

In order to construct a model for a theory we first need to construct the underlying allegory. Due to the previous lemma, the following structure is well-defined.

**Definition 33** *Let $T$ be a theory. Then $|\mathcal{H}_T|$ is defined as follows:*

1. *$Obj_{|\mathcal{H}_T|}$ is the class of closed object terms, i.e., object constant symbols,*
2. *For every pair of objects $c_1$ and $c_2$ a class of morphisms $|\mathcal{H}_T|[c_1, c_2]$ exists. The morphisms in $|\mathcal{H}_T|[c_1, c_2]$ are $[t]_{(c_1,c_2)}$ of type $c_1 \to c_2$ where $[t]_{(c_1,c_2)}$ denotes the equivalence class of the term $t$ with respect to $\sim_{c_1,c_2}$,*
3. $\mathbb{I}_c^{\mathcal{H}_T} = [\mathbb{I}_c]_{(c,c)}$,
4. $[t]^{\smile}_{(c_1,c_2)} = [t^{\smile}]_{(c_2,c_1)}$,
5. $[t_1]_{(c_1,c_2)}; [t_2]_{(c_2,c_3)} = [t_1; t_2]_{(c_1,c_3)}$,
6. $[t_1]_{(c_1,c_2)} \sqcap [t_2]_{(c_1,c_2)} = [t_1 \sqcap t_2]_{(c_1,c_2)}$.

The next step is to show that $|\mathcal{H}_T|$ is an allegory.

**Lemma 34** *Let $T$ be a theory. Then $|\mathcal{H}_T|$ is an allegory.*

**Proof.** We are required to show that the properties in Definition 2 are valid in $|\mathcal{H}_T|$.

From the definition we get:

$$\mathbb{I}_c^{\mathcal{H}_T}; [t]_{(c,c')} = [\mathbb{I}_c]_{(c,c)}; [t]_{(c,c')} = [\mathbb{I}_c; t]_{(c,c')}$$

By the axiom rules of calculus we have $\overline{\vdash (\forall a)(\forall b)(\forall r : a \to b)\mathbb{I}_a; r = r}$ , so we get the following derivations:

$$\dfrac{\dfrac{}{T \vdash (\forall a)(\forall b)(\forall r : a \to b)\mathbb{I}_a; r = r} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbb{I}_c; t = t \vdash \mathbb{I}_c; t = t}\,\text{Axiom}}{(\forall r : c \to c')\mathbb{I}_c; r = r \vdash \mathbb{I}_c; t = t}\,\forall\text{L (rel)}}{(\forall b)(\forall r : c \to b)\mathbb{I}_c; r = r \vdash \mathbb{I}_c; t = t}\,\forall\text{L (obj)}}{(\forall a)(\forall b)(\forall r : a \to b)\mathbb{I}_a; r = r \vdash \mathbb{I}_c; t = t}\,\forall\text{L (obj)}}{T, (\forall a)(\forall b)(\forall r : a \to b)\mathbb{I}_a; r = r \vdash \mathbb{I}_c; t = t}\,\text{Weak}}{T \vdash \mathbb{I}_c; t = t}\,\text{Cut}$$

This shows $(\mathbb{I}_c; t) \sim_{(c,c')} t$, and, hence,$[\mathbb{I}_c; t]_{(c,c')} = [t]_{(c,c')}$.

Proof for the remaining cases are similar to the previous proof. □

Now we can define a model for theory $T$.

**Definition 35** *Let $T$ be a theory. Then the Henkin-model [10] $\mathcal{H}_T$ of $T$ is defined by:*

1. $|\mathcal{H}_T|$ *is the allegory defined in Definition 33,*

2. $f^{\mathcal{H}_T}([t_1]_{(c_1,c'_1)},\ldots,[t_n]_{(c_n,c'_n)}) = [f(t_1,\ldots,t_n)]_{(c,c')}$,

3. $([t_1]_{(c_1,c'_1)},\ldots,[t_n]_{(c_n,c'_n)}) \in p^{\mathcal{H}_T}$ *iff* $T \vdash p(t_1,\ldots,t_n)$.

Notice that in the Henkin-model we have $\mathcal{V}_{\mathcal{H}}(t)(\sigma) = [t]$ for all closed terms $t$ independent of $\sigma$.

The model above is not necessarily a model of the theory. It might not even be a model because it is possible that the language does not have any closed terms, i.e., the underlying allegory of $\mathcal{H}_T$ is empty. But if the theory and the language is strong enough, then the Henkin-model is indeed a model of the theory.

**Definition 36** *A theory $T$ is called*

1. *complete iff $T \vdash \varphi$ or $T \vdash \neg\varphi$ for all closed formulas $\varphi$.*

2. *a Henkin-theory iff*

   (a) *for every closed formula $(\exists a)\varphi$ there is an object constant symbol $c'$ so that $T \vdash (\exists a)\varphi \to \varphi[c'/a]$,*

   (b) *for every closed formula $(\exists r : s_1 \to s_2)\varphi$ there is a relational constant symbol $c$ so that $T \vdash (\exists r : s_1 \to s_2)\varphi \to \varphi[c/r]$.*

As mentioned earlier we have the following lemma.

**Lemma 37** *If $T$ is a consistent and complete Henkin-theory, then $\mathcal{H}_T$ is a model of $T$.*

**Proof.** First of all, we want to show that the universe $|\mathcal{H}_T|$ is not empty. Since $T$ is a Henkin-theory we have $T \vdash (\exists a)\mathbb{I}_a = \mathbb{I}_a \to \mathbb{I}_{c_o} = \mathbb{I}_{c_o}$ for an object constant $c_o$ so that $|\mathcal{H}_T|$ has at least one object. If $c_1$ and $c_2$ are objects of $|\mathcal{H}_T|$, then we have $T \vdash (\exists r : c_1 \to c_2)r = r \to c_r = c_r$ because $T$ is a Henkin-theory. This implies that $[c_r]_{(c1,c2)}$ is a morphism between $c_1$ and $c_2$.

In order to show the property that $\varphi \in T$ implies $\mathcal{H}_T \models \varphi$ we are going to prove a stronger property. We are going to show

$$(*) \quad \mathcal{H}_T \models_{\sigma[[t_1]/r_1]\ldots[[t_n]/r_n]} \varphi \iff T \vdash \varphi[t_1/r_1]\ldots[t_n/r_n]$$

for all formulas $\varphi$ with free (object or relational) variables $r_1, \ldots, r_n$, closed (object or relational) terms $t_1, \ldots, t_n$ and environments $\sigma$. In the proof we are going to use the abbreviations

$$\begin{array}{rll} \overrightarrow{t} & \text{for} & (t_1, \ldots, t_n), \\ \overrightarrow{t/r} & \text{for} & [t_1/r_1]\ldots[t_n/r_n] \end{array}$$

and, similarly, $\overrightarrow{[t]}$, $\overrightarrow{[t]/r}$ and $\overrightarrow{\bar{\sigma}(t)/r}$. With those conventions (*) reads

$$\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi \iff T \vdash \varphi\overrightarrow{[t/r]}.$$

This is shown by induction.

$\varphi = p(t'_1, \ldots, t'_m)$**:** First of all, we have

$$\begin{aligned} \mathcal{V}_{\mathcal{H}_T}(t'_i)(\sigma\overrightarrow{[[t]/r]}) &= \mathcal{V}_{\mathcal{H}_T}(t'_i)(\sigma[\overrightarrow{\mathcal{V}_{\mathcal{H}_T}(t)(\sigma)/r}]) \\ &= \mathcal{V}_{\mathcal{H}_T}(t'_i\overrightarrow{[t/r]})(\sigma) && \text{Lemma 23(2)} \\ &= [t'_i\overrightarrow{[t/r]}] \end{aligned}$$

for $i \in \{1, \ldots, m\}$. We conclude

$$\begin{aligned} &\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} p(t'_1, \ldots, t'_m) \\ &\Leftrightarrow (\mathcal{V}_{\mathcal{H}_T}(t'_1)(\sigma\overrightarrow{[[t]/r]}), \ldots, \mathcal{V}_{\mathcal{H}_T}(t'_m)(\sigma\overrightarrow{[[t]/r]})) \in p^{\mathcal{H}_T} \\ &\Leftrightarrow ([t'_1\overrightarrow{[t/r]}], \ldots, [t'_m\overrightarrow{[t/r]}]) \in p^{\mathcal{H}_T} && \text{above} \\ &\Leftrightarrow T \vdash p(t'_1\overrightarrow{[t/r]}, \ldots, t'_m\overrightarrow{[t/r]}) \\ &\Leftrightarrow T \vdash p(t'_1, \ldots, t'_m)\overrightarrow{[t/r]}. \end{aligned}$$

$\varphi = (t_1 = t_2)$ Similar to previous case.

$\varphi = \bot$**:** In this case we have $\mathcal{H}_T \not\models \bot$ and $T \nvdash \bot$ since $T$ is consistent.

$\varphi = \neg\varphi'$**:** We immediately conclude

$$\begin{aligned}
\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \neg\varphi' &\Leftrightarrow \mathcal{H}_T \not\models_{\sigma\overrightarrow{[[t]/r]}} \varphi' \\
&\Leftrightarrow T \not\vdash \varphi'\overrightarrow{[t/r]} && \text{induction hypothesis} \\
&\Leftrightarrow T \vdash (\neg\varphi')\overrightarrow{[t/r]}. && T \text{ complete}
\end{aligned}$$

$\varphi = \varphi_1 \wedge \varphi_2$**:** In this case we have

$$\begin{aligned}
&\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_1 \wedge \varphi_2 \\
&\Leftrightarrow \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_1 \text{ and } \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_2 \\
&\Leftrightarrow T \vdash \varphi_1\overrightarrow{[t/r]} \text{ and } T \vdash \varphi_2\overrightarrow{[t/r]} && \text{induction hypothesis} \\
&\Rightarrow: \quad \frac{T \vdash \varphi_1\overrightarrow{[t/r]} \quad T \vdash \varphi_2\overrightarrow{[t/r]}}{T \vdash (\varphi_1 \wedge \varphi_2)\overrightarrow{[t/r]}}\ \wedge\mathrm{R} \\
&\Leftarrow: \quad \frac{T \vdash (\varphi_1 \wedge \varphi_2)\overrightarrow{[t/r]} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\varphi_1\overrightarrow{[t/r]} \vdash \varphi_1\overrightarrow{[t/r]}}\ \mathrm{Weak}}{\varphi_1\overrightarrow{[t/r]}, \varphi_2\overrightarrow{[t/r]} \vdash \varphi_1\overrightarrow{[t/r]}}\ \mathrm{Weak}}{T, \varphi_1\overrightarrow{[t/r]}, \varphi_2\overrightarrow{[t/r]} \vdash \varphi_1\overrightarrow{[t/r]}}\ \wedge\mathrm{L}}{T, (\varphi_1 \wedge \varphi_2)\overrightarrow{[t/r]} \vdash \varphi_1\overrightarrow{[t/r]}}\ \mathrm{Cut}}{T \vdash \varphi_1\overrightarrow{[t/r]}}
\end{aligned}$$

$T \vdash \varphi_2\overrightarrow{[t/r]}$ is shown analogously.

$\varphi = \varphi_1 \vee \varphi_2$**:** In this case we have

$$\begin{aligned}
&\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_1 \vee \varphi_2 \\
&\Leftrightarrow \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_1 \text{ or } \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_2 \\
&\Leftrightarrow T \vdash \varphi_1\overrightarrow{[t/r]} \text{ or } T \vdash \varphi_2\overrightarrow{[t/r]} && \text{induction hypothesis}
\end{aligned}$$

If $T \vdash \varphi_1\overrightarrow{[t/r]}$ then by using $\vee$R1 we conclude $T \vdash (\varphi_1 \vee \varphi_2)\overrightarrow{[t/r]}$. If $T \vdash \varphi_2\overrightarrow{[t/r]}$ then by using $\vee$R2 we conclude same assertion. For the converse implication we notice that $T$ is complete, which implies $T \vdash \varphi_1\overrightarrow{[t/r]}$ or $T \vdash \neg\varphi_1\overrightarrow{[t/r]}$. If $T \vdash \varphi_1\overrightarrow{[t/r]}$ then $T \vdash \varphi_1\overrightarrow{[t/r]}$ or $T \vdash \varphi_2\overrightarrow{[t/r]}$ holds. If $T \vdash \neg\varphi_1$ then we can show $T \vdash \varphi_2\overrightarrow{[t/r]}$ by the following derivation:

$$\dfrac{T \vdash (\varphi_1 \vee \varphi_2)\overrightarrow{[t/r]} \qquad \dfrac{\dfrac{T \vdash \neg\varphi_1\overrightarrow{[t/r]}}{T, \varphi_1\overrightarrow{[t/r]} \vdash \varphi_2\overrightarrow{[t/r]}}\,\neg\mathrm{L} \quad \dfrac{T \vdash \varphi_2\overrightarrow{[t/r]}}{T, \varphi_2\overrightarrow{[t/r]} \vdash \varphi_2\overrightarrow{[t/r]}}\,\mathrm{Weak}}{T, (\varphi_1 \vee \varphi_2)\overrightarrow{[t/r]} \vdash \varphi_2\overrightarrow{[t/r]}}\,\vee\mathrm{L}}{T \vdash \varphi_2\overrightarrow{[t/r]}}\,\mathrm{Cut}$$

$\varphi = \varphi_1 \to \varphi_2$**:** In this case we have

$$\begin{aligned}
&\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_1 \to \varphi_2 \\
&\Leftrightarrow \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \neg\varphi_1 \text{ or } \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} \varphi_2 \\
&\Leftrightarrow T \vdash \neg\varphi_1\overrightarrow{[t/r]} \text{ or } T \vdash \varphi_2\overrightarrow{[t/r]} \qquad \text{induction hypothesis}
\end{aligned}$$

$\Rightarrow$: We distinguish two cases:

Case $T \vdash \neg\varphi_1\overrightarrow{[t/r]}$:

$$\dfrac{\dfrac{T \vdash \neg\varphi_1\overrightarrow{[t/r]}}{T, \varphi_1\overrightarrow{[t/r]} \vdash \varphi_2\overrightarrow{[t/r]}}\,\neg\mathrm{L}}{T \vdash (\varphi_1 \to \varphi_2)\overrightarrow{[t/r]}}\,\to\mathrm{R}$$

Case $T \vdash \varphi_1\overrightarrow{[t/r]}$:

$$\dfrac{\dfrac{T \vdash \varphi_2\overrightarrow{[t/r]}}{T, \varphi_1\overrightarrow{[t/r]} \vdash \varphi_2\overrightarrow{[t/r]}}\,\mathrm{Weak}}{T \vdash (\varphi_1 \to \varphi_2)\overrightarrow{[t/r]}}\,\to\mathrm{R}$$

$\Leftarrow$: $T$ is complete, so $T \vdash \neg\varphi_1\overrightarrow{[t/r]}$ or $T \vdash \neg\varphi_1\overrightarrow{[t/r]}$. If $T \vdash \neg\varphi_1\overrightarrow{[t/r]}$ then $T \vdash \neg\varphi_1\overrightarrow{[t/r]}$ or $T \vdash \varphi_2\overrightarrow{[t/r]}$holds. If $T \vdash \varphi_1$ then we can show $T \vdash \varphi_2\overrightarrow{[t/r]}$ by the following derivation:

$$\dfrac{T \vdash (\varphi_1 \to \varphi_2)\overrightarrow{[t/r]} \qquad \dfrac{T \vdash \varphi_1\overrightarrow{[t/r]} \quad \dfrac{T \vdash \varphi_2\overrightarrow{[t/r]}}{T, \varphi_2\overrightarrow{[t/r]} \vdash \varphi_2\overrightarrow{[t/r]}}\,\mathrm{Weak}}{T, (\varphi_1 \to \varphi_2)\overrightarrow{[t/r]} \vdash \varphi_2\overrightarrow{[t/r]}}\,\to\mathrm{L}}{T \vdash \varphi_2\overrightarrow{[t/r]}}\,\mathrm{Cut}$$

$\varphi = (\exists q : s_1 \to s_2)\varphi'$**:** First of all, we have

$$\begin{aligned}
&\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} (\exists q : s_1 \to s_2)\varphi' \\
&\Leftrightarrow \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r][[t']/q]}} \varphi' && \text{for some } [t'] \in |\mathcal{H}_T| \\
&\Leftrightarrow T \vdash \varphi'\overrightarrow{[t/r]}[t'/q]]. && \text{for some closed term } t' \\
& && \text{by the induction hypothesis}
\end{aligned}$$

It remains to show that the last property is equivalent to $T \vdash ((\exists q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]}$. The implication $\Rightarrow$ follows by using the rule $\exists$R (rel). Conversely, assume $T \vdash ((\exists q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]}$. Since $T$ is a Henkin-theory there is a relational constant symbol $c$ with $T \vdash ((\exists q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]} \to \varphi'\overrightarrow{[t/r]}[c/q]$. Suppose $A = ((\exists q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]}$ and $B = \varphi'\overrightarrow{[t/r]}[c/q]$. We conclude:

$$\dfrac{T \vdash A \to B \qquad \dfrac{\dfrac{T \vdash A}{T, A \to B \vdash A}\,\text{Weak} \qquad \dfrac{\dfrac{A \vdash A \quad \dfrac{B \vdash B}{A, B \vdash B}\,\text{Weak}}{A, A \to B \vdash B}\,{\to}\text{L}}{T, A, A \to B \vdash B}\,\text{Weak}}{T, A \to B \vdash B}\,\text{Cut}}{T \vdash B}\,\text{Cut}$$

$\varphi = (\exists a)\varphi'$**:** Similar to the previous case.

$\varphi = (\forall q : s_1 \to s_2)\varphi'$**:** Similar to the previous case we get

$$\begin{aligned}
&\mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r]}} (\forall q : s_1 \to s_2)\varphi' \\
&\Leftrightarrow \mathcal{H}_T \models_{\sigma\overrightarrow{[[t]/r][[t']/q]}} \varphi' && \text{for all } [t'] \in |\mathcal{H}_T| \\
&\Leftrightarrow T \vdash \varphi'\overrightarrow{[t/r]}[t'/q]. && \text{for all closed term } t' \\
& && \text{by the induction hypothesis}
\end{aligned}$$

and it remains to be shown that the last property is equivalent to $T \vdash ((\forall q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]}$. $\Rightarrow$: Assume $T \vdash \varphi'\overrightarrow{[t/r]}[t'/q]$ for all closed terms $t'$. Since $T$ is a Henkin-theory we have $T \vdash ((\exists q : s_1 \to s_2)\neg\varphi')\overrightarrow{[t/r]} \to (\neg\varphi')\overrightarrow{[t/r]}[c/q]$ for a relational constant symbol $c$. Let $A = ((\exists q : s_1 \to s_2)\neg\varphi')\overrightarrow{[t/r]}$ and

$B = (\neg\varphi')\overrightarrow{[t/r]}[c/q]$. We get:

$$\dfrac{\dfrac{T \vdash A \to B}{T, A \vdash A \to B}\ \text{Weak} \quad \dfrac{\dfrac{A \vdash A \quad \dfrac{B \vdash B}{A, B \vdash B}\ \text{Weak}}{A, A \to B \vdash B}\ \to\text{L}}{T, A, A \to B \vdash B}\ \text{Weak}}{T, A \vdash B}\ \text{Cut}$$

Let $\psi$ denote $\varphi'\overrightarrow{[t/r]}$,and $q$ be a relation of type $s_1 \to s_2$. So from the above derivation we know $T, (\exists q)\neg\psi \vdash \neg\psi[c/q]$. We get:

$$\dfrac{\dfrac{\dfrac{T, (\exists q)\neg\psi \vdash \neg\psi[c/q] \quad \dfrac{\dfrac{T \vdash \psi[c/q]}{T, (\exists q)\neg\psi \vdash \psi[c/q]}\ \text{Weak}}{T, (\exists q)\neg\psi, \neg\psi[c/q] \vdash \bot}\ \neg\text{L}}{T, (\exists q)\neg\psi \vdash \bot}\ \text{Cut}}{T \vdash \neg((\exists q)\neg\psi)}\ \neg\text{R} \qquad \dfrac{\dfrac{\dfrac{\neg\psi \vdash \neg\psi}{\neg\psi \vdash (\exists q)\neg\psi}\ \exists\text{R (rel)}}{T, \neg\psi \vdash (\exists q)\neg\psi}\ \text{Weak}}{T, \neg\psi, \neg((\exists q)\neg\psi) \vdash \bot}\ \neg\text{L}}{T, \neg\psi \vdash \bot}\ \text{Cut}$$

$$\dfrac{T \vdash \bot}{T \vdash \psi}\ \text{PBC}$$

$$T \vdash (\forall q)\psi\ \forall\text{R (rel)}$$

$\Leftarrow$:

$$\dfrac{T \vdash ((\forall q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]} \quad \dfrac{\dfrac{\dfrac{\varphi'\overrightarrow{[t/r]}[c/q] \vdash \varphi'\overrightarrow{[t/r]}[c/q]}{((\forall q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]} \vdash \varphi'\overrightarrow{[t/r]}[c/q]}\ \forall\text{L (rel)}}{T, ((\forall q : s_1 \to s_2)\varphi')\overrightarrow{[t/r]} \vdash \varphi'\overrightarrow{[t/r]}[c/q]}\ \text{Weak}}{T \vdash \varphi'\overrightarrow{[t/r]}[c/q]}\ \text{Cut}$$

$\varphi = (\forall a)\varphi'$**:** Similar to the previous case. □

It remains to show that every consistent theory can be extended to a consistent and complete Henkin-theory. First, we want to show that just adding new constant symbols does not have any effect on the consistency of a theory.

In order to distinguish different languages we denote by $L(T)$ the language of $T$, i.e., $L(T) = (C_{obj}, C_{rel}, F, P)$ with $C_{obj}$ the set of object constants, $C_{rel}$ the set of relational constants, $F$ the set of function symbols and $P$ the set of predicate symbols.

We say that $\varphi$ is a formula in the language $L(T)$ iff $\varphi$ just contains symbols from $C_{obj}$, $C_{rel}$, $F$ and $P$.

**Lemma 38** *Let $T$ be a theory, $\varphi$ be a formula in the language $L(T)$, and $C'_{rel}$ and $C'_{obj}$ be a set of relational and object constant symbols with $C'_{obj}$, $C_{obj}$, $C'_{rel}$, $C_{rel}$ and $F$ be pairwise disjoint. Then we have:*

1. *for all $c_1, \ldots, c_n \in C'_{rel}$ and relational variables $r_1, \ldots, r_n$*

$$T \vdash \varphi \qquad \Longleftrightarrow \qquad T \vdash \varphi[c_1/r_1] \ldots [c_n/r_n].$$

2. *for all $c'_1, \ldots, c'_n \in C'_{obj}$ and object variables $a_1, \ldots, a_n$*

$$T \vdash \varphi \qquad \Longleftrightarrow \qquad T \vdash \varphi[c'_1/a_1] \ldots [c'_n/a_n].$$

**Proof.**

1. $\Rightarrow$: Assume we have $T \vdash \varphi$. By using Lemma 27(3) $n$-times we get a derivation $T \vdash (\forall r_1, \ldots r_n)\varphi$. We conclude:

$$\dfrac{T \vdash (\forall r_1, \ldots, r_n)\varphi \qquad \dfrac{\dfrac{\dfrac{\varphi[c_1/r_1] \ldots [c_n/r_n] \vdash \varphi[c_1/r_1] \ldots [c_n/r_n]}{T, \varphi[c_1/r_1] \ldots [c_n/r_n] \vdash \varphi[c_1/r_1] \ldots [c_n/r_n]}\,\text{Weak}}{T, (\forall r_1)\varphi[c_2/r_2] \ldots [c_n/r_n] \vdash \varphi[c_1/r_1] \ldots [c_n/r_n]}\,\forall\text{L (rel)}}{\begin{array}{c}\vdots\ \text{repeat } \forall\text{L (rel) n-2 times}\\ T, (\forall r_1, \ldots, r_{n-1})\varphi[c_n/r_n] \vdash \varphi[c_1/r_1] \ldots [c_n/r_n]\end{array}\,\forall\text{L (rel)}}}{T, (\forall r_1, \ldots, r_n)\varphi \vdash \varphi[c_1/r_1] \ldots [c_n/r_n]}\,\text{Cut}}{T \vdash \varphi[c_1/r_1] \ldots [c_n/r_n]}$$

   $\Leftarrow$: We are going to use a similar notion as in the proof of Lemma 37, and we prove the following more general property for all formulas $\psi_1, \ldots, \psi_m$ and $\varphi$: Let $c_1, \ldots, c_n$ be the new relational constant symbols that occur in a derivation $T \cup \{\psi_1\overrightarrow{[c/r]}, \ldots, \psi_m\overrightarrow{[c/r]}\} \vdash \varphi\overrightarrow{[c/r]}$. Then there is a derivation $T \cup \{\psi_1, \ldots, \psi_m\} \vdash \varphi$. We are going to prove this property by induction on the structure of the give derivation.

   If the derivation is just an assumption, the formula $\varphi\overrightarrow{[c/r]}$ is either in $T$ or equal to $\psi_i\overrightarrow{[c/r]}$ for an $i \in \{1, \ldots, m\}$. In the first case the formula does not contain any relational constant $c$ since $F \cap C_{rel} = \emptyset$ so that $\varphi\overrightarrow{[c/r]}$ is actually

equal to $\varphi$. In the latter case we conclude that $\varphi = \psi_i$ since both formulas do not contain any of the new constant symbols.

**Cut:** In this case we have derivations

$$T, \psi_1\overrightarrow{[c/r]}, \ldots, \psi_m\overrightarrow{[c/r]} \vdash \varphi',$$
$$T, \psi_1\overrightarrow{[c/r]}, \ldots, \psi_m\overrightarrow{[c/r]}, \varphi_1, \varphi' \vdash \varphi\overrightarrow{[c/r]}$$

for some formulas $\varphi'$. In the extended language, $\varphi'$ may contain elements from $C_{rel}$. Therefore, $\varphi = \varphi''\overrightarrow{[c/r]}$. By the assumption on the new constant occurring in the derivation and the induction hypothesis we get derivations

$$T, \psi_1, \ldots, \psi_m \vdash \varphi'',$$
$$T, \psi_1, \ldots, \psi_m, \varphi'' \vdash \varphi$$

By applying Cut rule we get $T, \psi_1, \ldots, \psi_m \vdash \varphi$

$\wedge$**L:** In this case we have the following derivation:

$$\frac{T, \psi_1\overrightarrow{[c/r]}, \ldots, \psi_m\overrightarrow{[c/r]}, \varphi_1\overrightarrow{[c/r]}, \varphi_2\overrightarrow{[c/r]} \vdash \varphi\overrightarrow{[c/r]}}{T, \psi_1\overrightarrow{[c/r]}, \ldots, \psi_m\overrightarrow{[c/r]}, \varphi_1\overrightarrow{[c/r]} \wedge \varphi_2\overrightarrow{[c/r]} \vdash \varphi\overrightarrow{[c/r]}} \wedge\mathrm{L}$$

By induction hypothesis we get that $T, \psi_1, \ldots, \psi_m, \varphi_1, \varphi_2 \vdash \varphi$ and by applying $\wedge$L to the last derivation we get $T, \psi_1, \ldots, \psi_m, \varphi_1 \wedge \varphi_2 \vdash \varphi$

The remaining cases are similar to the previous case.

2. can be shown analogously.

In particular, the previous lemma implies that $T$ is consistent iff $T$ is consistent with respect to a language enriched by new constant symbols.

**Definition 39** *Let $T$ be a theory in the language $L(T)$. We define the following languages and theories recursively:*

1. *$L_0 := L(T)$ and $T_0 := T$.*
2. *Let $C_{(rel)_{n+1}} := \{c_{(\exists r:s_1 \to s_2)\varphi} \mid (\exists r : s_1 \to s_2)\varphi$ a closed formula in $L_n\}$ and $C_{(obj)_{n+1}} := \{c'_{(\exists a)\varphi} \mid (\exists a)\varphi$ a closed formula in the language $L_n\}$ be two sets of new constant symbols, i.e., $C_{(rel)_{n+1}} \cap F_n = C_{(obj)_{n+1}} \cap F_n = \emptyset$. Then $L_{n+1} :=$*

$L_n \cup C_{(rel)_{n+1}} \cup C_{(obj)_{n+1}}$ *and* $T_{n+1} := T_n \cup \{(\exists r : s_1 \to s_2)\varphi \to \varphi[c_{(\exists r:s_1\to s_2)\varphi}/r] \mid c_{(\exists r:s_1\to s_2)\varphi} \in C_{(rel)_{n+1}}\} \cup \{(\exists a)\varphi \to \varphi[c'_{(\exists a)\varphi}/a] \mid c'_{(\exists a)\varphi} \in C_{(obj)_{n+1}}\}$.

3. $L_H := \bigcup_{i\geq 0} L_i$, *and* $T_H := \bigcup_{i\geq 0} T_i$.

**Lemma 40** *If $T$ is a consistent theory, then $T_H$ is a consistent Henkin-theory.*

**Proof.** First, we show by induction that every $T_n$ is consistent. For $n = 0$ this is trivial. Assume there is a derivation $T_{n+1} \vdash \bot$. Then there are $m$ formulas $\psi_i = (\exists r_i : a \to b)\varphi_i \to \varphi_i[c_i/r_i]$ or $\psi_i = (\exists a_i)\varphi_i \to \varphi_i[c'_i/a_i]$ with $i \in \{1, \ldots, m\}$ so that the derivation above is actually a derivation $T_n \cup \{\psi_1, \ldots, \psi_m\} \vdash \bot$. Both cases are exactly the same except when $\psi_i = (\exists a_i)\varphi_i \to \varphi_i[c'_i/a_i]$ we use $\exists$R (obj) and $\exists$L (obj) instead of $\exists$R (rel) and $\exists$L (rel), so for simplicity we assume $\psi_i$ is only of form $(\exists r_i : s_1 \to s_2)\varphi_i \to \varphi_i[c_i/r_i]$. We also assume that the relational variables $r_1, \ldots, r_m$ are different, otherwise we rename certain variables. By Lemma 26(2) we get a derivation $T_n \vdash \psi_1 \to (\psi_2 \to \ldots(\psi_m \to \bot)\ldots)$. Notice that $\psi_i$ is of the form $\psi'_i[c_i/r_i]$ with $\psi'_i = (\exists r_i)\varphi_i \to \varphi_i$ a formula in the language $L_n$ so that the previous statement can be written as $T_n \vdash \psi'_1 \to (\psi'_2 \to \ldots(\psi'_m \to \bot)\ldots)[c_1/r_1]\cdots[c_n/r_n]$. Lemma 38 implies that there is a derivation $T_n \vdash \psi'_1 \to (\psi'_2 \to \ldots(\psi'_m \to \bot)\ldots)$ in the language $L_n$, and, hence, $T_n, \psi'_1, \ldots, \psi'_m \vdash \bot$ using Lemma 26(2) again. The following steps are repeated $m$ times:

By Lemma 26(2) we get a derivation $T_n, \psi'_1, \ldots, \psi'_{m-1} \vdash \psi'_m \to \bot$, and by Lemma 27(3) $T_n, \psi'_1, \ldots, \psi'_{m-1} \vdash (\forall r_m : s_1 \to s_2)(\psi'_m \to \bot)$. By the following derivation we get $\vdash (\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\exists r_m)\varphi_m \vdash (\exists r_m)\varphi_m}{(\exists r_m)\varphi_m \vdash (\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m}\vee\mathrm{R}}{\neg((\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m), (\exists r_m)\varphi_m \vdash \bot}\neg\mathrm{L}}{\neg((\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m) \vdash \neg(\exists r_m)\varphi_m}\mathrm{PBC}}{\neg((\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m) \vdash (\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m}\vee\mathrm{R}}{\neg((\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m), \neg((\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m) \vdash \bot}\neg\mathrm{L}}{\neg((\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m) \vdash \bot}\mathrm{Cont.}}{\vdash (\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m}\mathrm{PBC}$$

Consider the derivation of $(\exists r_m : s_1 \to s_2)\psi'_m$, and the combination of that derivation and $T_n, \psi'_1, \ldots, \psi'_{m-1} \vdash (\forall r_m : s_1 \to s_2)(\psi'_m \to \bot)$ given in Figure 4.5. This shows that there is a derivation $T_n, \psi'_1, \ldots, \psi_{m-1} \vdash \bot$.

Let $\Gamma = T_n, \psi'_1, \ldots, \psi'_{m-1}$ and $r_m$ a relational variable of type $s_1 \to s_2$.

$$\dfrac{\vdash (\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\varphi_m \vdash \varphi_m}{\varphi_m, (\exists r_m)\varphi_m \vdash \varphi_m}\,\text{Weak}}{\varphi_m \vdash (\exists r_m)\varphi_m \to \varphi_m}\,{\to}\text{R}}{\varphi_m \vdash (\exists r_m)((\exists r_m)\varphi_m \to \varphi_m)}\,\exists\text{R (rel)}}{(\exists r_m)\varphi_m \vdash (\exists r_m)((\exists r_m)\varphi_m \to \varphi_m)}\,\exists\text{L (rel)} \qquad \dfrac{\dfrac{\dfrac{(\exists r_m)\varphi_m \vdash (\exists r_m)\varphi_m}{\neg(\exists r_m)\varphi_m, (\exists r_m)\varphi_m \vdash \varphi_m}\,\neg\text{L}}{\neg(\exists r_m)\varphi_m \vdash (\exists r_m)\varphi_m \to \varphi_m}\,{\to}\text{R}}{\neg(\exists r_m)\varphi_m \vdash (\exists r_m)((\exists r_m)\varphi_m \to \varphi_m)}\,\exists\text{R (rel)}}{(\exists r_m)\varphi_m \vee \neg(\exists r_m)\varphi_m \vdash (\exists r_m)((\exists r_m)\varphi_m \to \varphi_m)}\,\vee\text{L}}{\vdash (\exists r_m)((\exists r_m)\varphi_m \to \varphi_m)}\,\text{Cut}$$

$$\dfrac{\dfrac{\vdash (\exists r_m)\psi'_m}{\Gamma \vdash (\exists r_m)\psi'_m}\,\text{Weak} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash (\forall r_m)(\psi'_m \to \bot) \qquad \dfrac{\dfrac{\dfrac{\psi'_m \to \bot \vdash \psi'_m \to \bot}{(\forall r_m)(\psi'_m \to \bot) \vdash \psi'_m \to \bot}\,\forall\text{L (rel)}}{\Gamma, (\forall r_m)(\psi'_m \to \bot) \vdash \psi'_m \to \bot}\,\text{Weak}}{\Gamma \vdash \psi'_m \to \bot}\,\text{Cut}}{\Gamma, \psi'_m \vdash \psi'_m \to \bot}\,\text{Weak} \qquad \dfrac{\dfrac{\psi'_m \vdash \psi'_m \qquad \dfrac{\bot \vdash \bot}{\psi'_m, \bot \vdash \bot}\,\text{Weak}}{\psi'_m, \psi'_m \to \bot \vdash \bot}\,{\to}\text{L}}{\Gamma, \psi'_m, \psi'_m \to \bot \vdash \bot}\,\text{Weak}}{\Gamma, \psi'_m \vdash \bot}\,\text{Cut} \qquad \dfrac{\Gamma, \psi'_m \vdash \bot}{\Gamma, (\exists r_m)\psi'_m \vdash \bot}\,\exists\text{L (rel)}^*}{\Gamma \vdash \bot}\,\text{Cut}$$

Note that the rule condition for * is satisfied since the only free variable in $\psi_i$ is $r_i$ and variables $r_1, \ldots, r_m$ are different.

Figure 4.5: Derivation Tree for $T_n, \psi'_1, \ldots, \psi_{m-1} \vdash \bot$

After $m$ repetitions we end up with a derivation $T_n \vdash \bot$, a contradiction to the induction hypothesis that $T_n$ is consistent.

Now, assume $T_H$ is not consistent. Since every derivation just uses finitely many premises and every formula uses just finitely many symbols this derivation is a derivation $T_n \vdash \bot$ for some $n$, a contradiction.

It remains to show that $T_H$ is a Henkin-theory. Assume that $(\exists r : s_1 \to s_2)\varphi$ is a closed formula in $L_H$. Since the formula just contains finitely many symbols there is an $n$ so that $(\exists r : s_1 \to s_2)\varphi$ is a closed formula in the language $L_n$. The theory $T_{n+1}$ contains the formula $(\exists r : s_1 \to s_2)\varphi \to \varphi[c/r]$ for a relational constant symbol $c$ so that $T_H \vdash (\exists r : s_1 \to s_2)\varphi \to \varphi[c/r]$ follows immediately. □

For the next step we assume that the closed formulas of the language are enumerated, and we denote by $\varphi_n$ the n-th closed formula. This does not cause any problems since the sets of variables, constant symbols, and function symbols, and, hence, the set of formulas are countable.

**Definition 41** *Let $T$ be a theory, and define the following theories recursively:*

1. $T_0 := T$.
2. $T_{n+1} := \begin{cases} T_n \cup \{\varphi_n\} & \text{if } T_n \cup \{\varphi_n\} \text{ is consistent,} \\ T_n \cup \{\neg\varphi_n\} & \text{if } T_n \cup \{\varphi_n\} \text{ is inconsistent.} \end{cases}$
3. $T^c := \bigcup_{i \geq 0} T_i$.

**Lemma 42** *If $T$ is a consistent Henkin-theory, then $T^c$ is a consistent and complete Henkin-theory.*

**Proof.** First, we want to show that each $T_n$ is consistent. The case $n = 0$ is trivial. Assume that $T_{n+1}$ is inconsistent. Then by the construction of $T_{n+1}$ the theory $T_n$ is inconsistent, a contradiction to the induction hypothesis.

Now, assume $T^c$ is inconsistent. Then there is a derivation $T^c \vdash \bot$. Since every derivation use just finitely many premises this derivation is actually a derivation $T_n \vdash \bot$ for some $n$, a contradiction.

$T^c$ is Henkin-theory because $T = T_0$ is, and the language was not modified.

Finally, for every closed formula $\varphi_n$ we have $\varphi \in T_n$ or $\neg\varphi_n \in T_n$ so that $T^c$ is complete. □

Now we are finally ready to prove the main theorem of this section.

**Theorem 43** *Every consistent theory has a model.*

**Proof.** Let $T$ be a consistent theory. Then the theory $T_H^c$, precisely $(T_H)^c$, is a consistent and complete Henkin-theory. By Lemma 37 this theory has a model $\mathcal{H}_T$. Let ${\mathcal{H}'}_T$ denote the model derived from $\mathcal{H}_T$ by restricting $\mathcal{H}_T$ to the language of $T$, i.e., removing the interpretation of those symbols that are not in the language $L(T)$.

Let $\varphi \in T$. Then $\varphi \in T_H^c$, and, hence, we have $\mathcal{H}_T \models \varphi$. Since $\varphi$ is a formula in the language $L(T)$ we conclude ${\mathcal{H}'}_T \models \varphi$. □

# Chapter 5

# RelAPS

This chapter will discuss the extension of RelAPS to first order logic. We will start by giving a brief overview of the system before the extension. We will then discuss how the extended logic and its rules were implemented, and how to use the new version of the application.

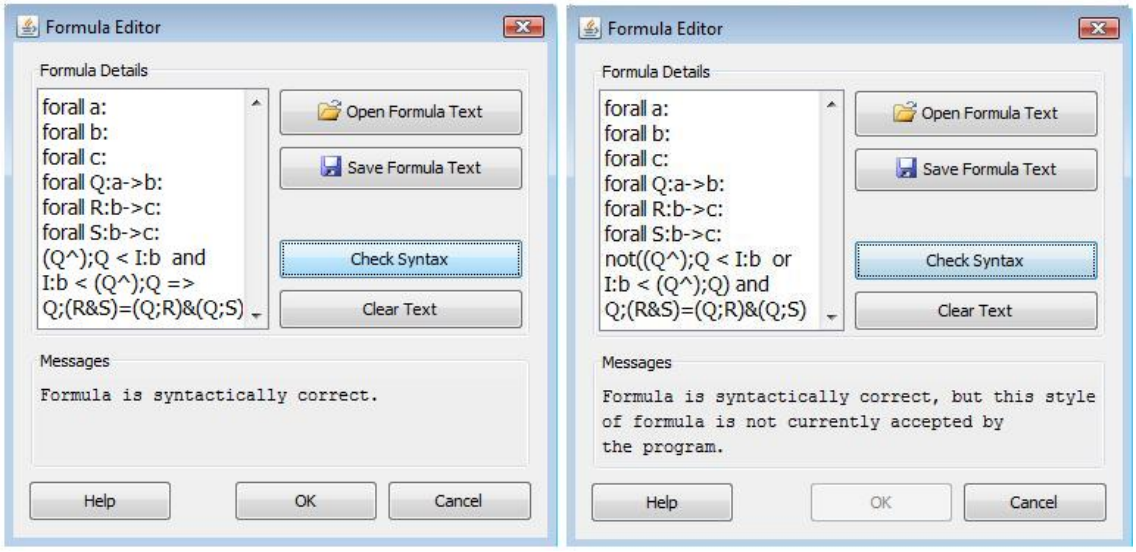
## 5.1 Overview of RelAPS

As mentioned before, the purpose of the RelAPS system is to provide an environment where a user can perform a relation-algebraic proof similar to doing it using pencil and paper. It is the responsibility of the user to complete the proof while the system ensures that each individual proof step is executed properly.

The previous version of RelAPS system has several aspects that are worth mentioning. A user may define custom operations. The nullary operations are actually the relational constant symbols we discussed in Chapters 2 and 3. The user defined operations may be combined with arbitrary axioms to produce new theories which represent the theories we introduced in Definition 29. The new theories can be saved and selected next time when the application starts. The base theory of the system is the theory of allegories which consists of the identity, converse, intersection and composition operations. For more detail about the previous version refer to [7, 8].

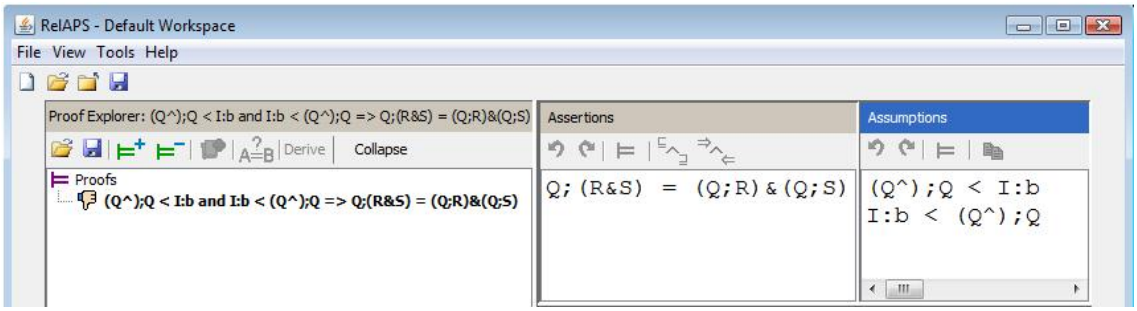
However the system only accepts Horn-formulas. These are formulas of the form $(\forall x_1) \ldots (\forall x_m) e_1 \wedge \ldots \wedge e_n \rightarrow e$ where $x_1, ..., x_m$ are relational or object variables and the $e_1, ..., e_n, e$ are atomic formulas. Figure 5.1 is an example of valid and invalid formulas.

Figure 5.1: Example of Valid and Invalid Formulas




When such a Horn-formula is entered, certain rules will be applied automatically, namely $\forall$R (rel) and $\wedge$L, until all formulas are split into atomic formulas. An atomic formula is a formula that contains no logical connectives. Figure 5.2 shows the previous example after the system removes its identifiers.

Figure 5.2: After Automatic Rules Applied



Suppose we want to prove the example above using the system. First we need to split the equality formula into two inclusions $Q;(R \wedge S) < Q;R \wedge Q;S$ and $Q;R \wedge Q;S < Q;(R \wedge S)$. This could be done by selecting the entire formula in the 'Assertion' window and pressing the fourth button as shown in Figure 5.3. Figure 5.4 shows the system after splitting.

Figure 5.3: Splitting Equality to two Inclusions

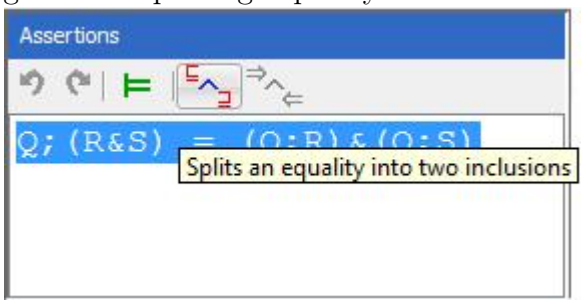
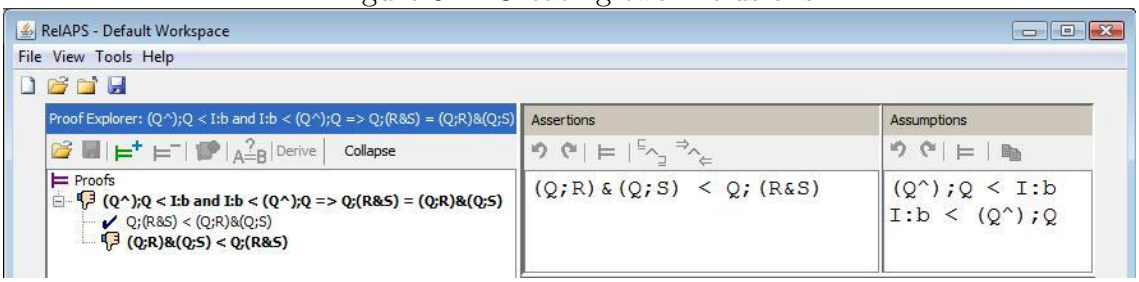



Figure 5.4: Creating two Inclusions




The first inclusion is trivial since it is a property of allegories so we just need to prove the second formula. By selecting the entire left hand side of the second inclusion and pressing the 'Derive' button, the selection will be moved to the 'Working Area'. The 'Working Area' window is where the main part of the derivation is performed.

When a term is selected, a menu immediately pops up which displays the axioms, assumptions, and theorems that may be applied to the current selection. Figure 5.5 is a screen shot of this process.

Figure 5.5: Selecting a Term

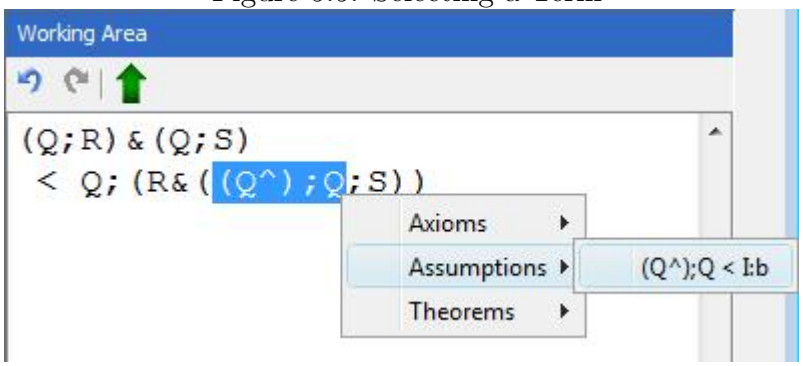



Figure 5.6 and Figure 5.7 show the derivation steps and the state of the system after applying the derivation.

For more detail about how to use the previous version of RelAPS refer to [8].

Figure 5.6: Derivation Steps

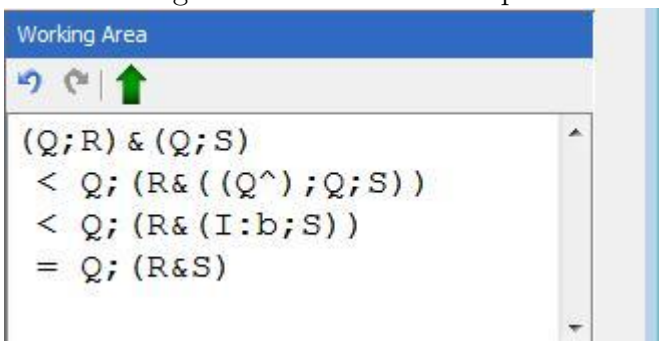
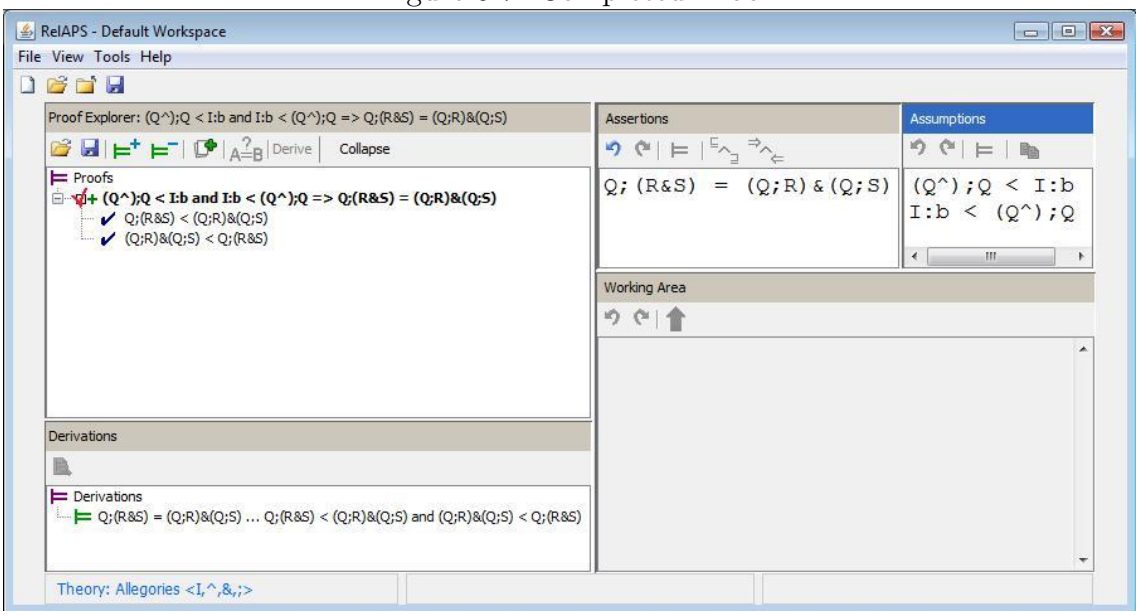


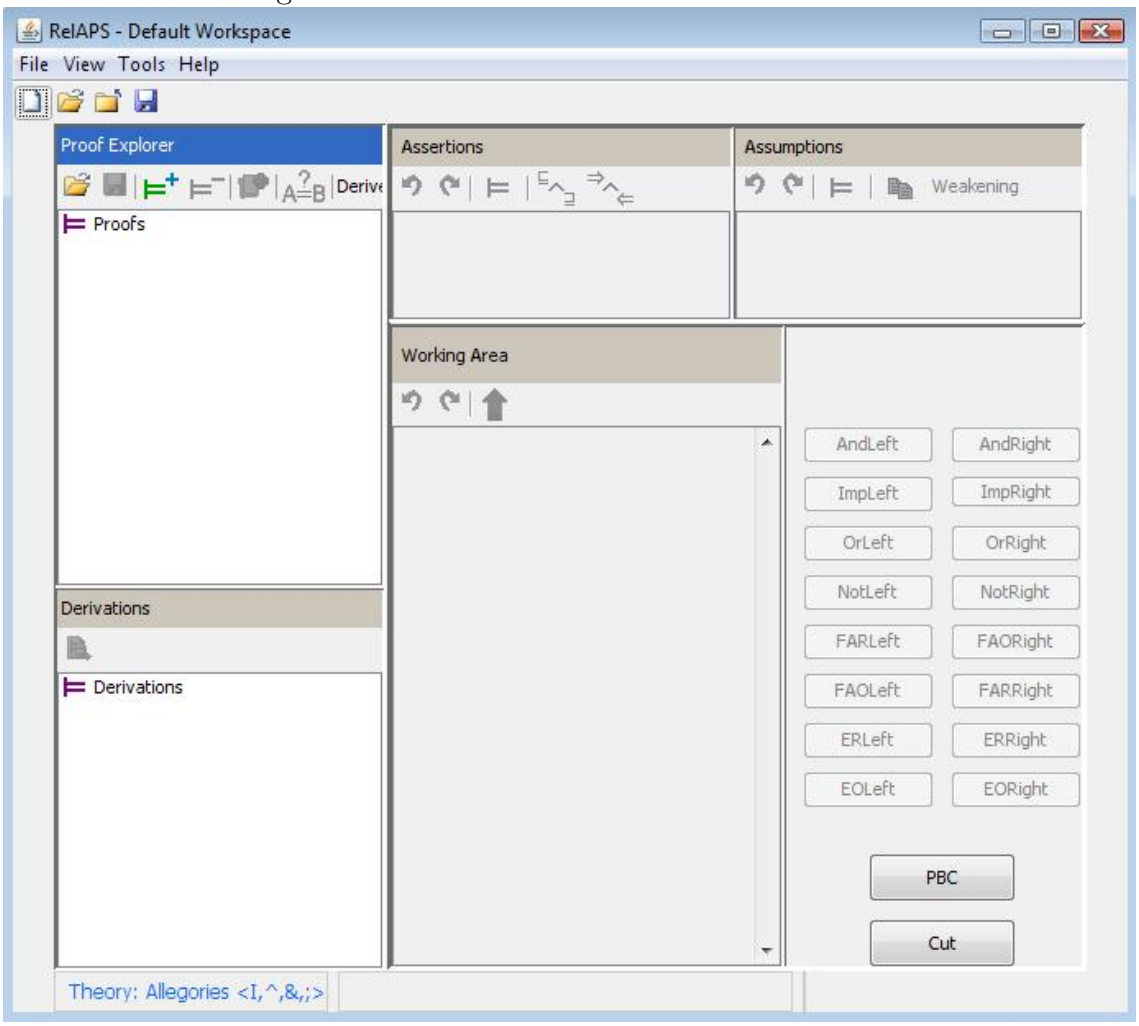
Figure 5.7: Completed Proof



## 5.2 Extending RelAPS

The new version of RelAPS accepts first order formulas. During the proof the user can apply any of the rules mentioned in Section 3.1. As shown in Figure 5.8 new buttons have been added to the application interface for this purpose.

Figure 5.8: View of the New Version of RelAPS




### 5.2.1 A Manual for the System

The language of the new system is the same as the previous one except that the new version accepts formulas containing $\perp$. For simplicity we used symbol '%' to

represent $\perp$.

As in the previous version, the system has 'Assertions' and 'Assumptions' windows. The 'Assertions' window displays the assertion of the current proof which is the right hand side of $\vdash$ in a derivation. The text area of the 'Assertions' window simply displays the current state of the assertion being worked with. The user may only work with one assertion at a time. This is specified by clicking the appropriate assertion in the tree view of the 'Proof Explorer' window.

The 'Assumptions' window displays the assumptions that are associated with the current proof. This corresponds with the sequence $\Gamma$ on the left-hand side of $\vdash$. The buttons on the tool bar work in the same manner as before, with the exception of 'Weakening' and 'Duplicate' buttons. The 'Weakening' button implements the Weakening rule by removing a selected assumption from the current proof. The 'Duplicate' button implements the Contraction rule by duplicating a selected assumption. The text area of the 'Assumptions' window allows for the selection of those parts of any assumption that user wish to modify. Multiple assumptions are always in view in this window, and any of them may be selected at any time.

The buttons on the right side of 'Working Area' are used to apply derivation rules, mentioned in Section 4.1, on the current proof. All derivation buttons are disabled by default except PBC and Cut. The right hand rule buttons are enabled based on subtree that has been selected in 'Proof Explorer'. An appropriate left hand rule button will be enabled when user selects an assumption in the 'Assumptions' window. Note that the Permutation rule is already implemented within the system since the formulas in the 'Assumption' window can be selected freely, i.e., they are not ordered. The Axiom rule is automatically checked by the system. In that step the system actually checks whether the assertion is among the formulas in the assumption window, i.e., the Weakening rule is implicitly used in this process.

In the following we give an example showing how the different components of the system work. Suppose we want to prove the following formula.

$$
\begin{aligned}
&\{(\forall a)(\forall X : a \to a)(X; X = X \wedge X^{\smile} = X) \Rightarrow \\
&(\exists b)(\exists R : b \to a)(R; R^{\smile} = \mathbb{I}_b \wedge R^{\smile}; R = X)\} \Rightarrow \\
&\{(\forall a)(\forall X : a \to a)(\forall Y : a \to a)(X; X = X \wedge X^{\smile} = X \wedge X; Y = Y) \Rightarrow \\
&(\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z; V = Y\}.
\end{aligned}
$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\varphi_3 \vdash \varphi_1 \quad \varphi_2, \varphi_3 \vdash \varphi_4}{\varphi_1 \Rightarrow \varphi_2, \varphi_3 \vdash \varphi_4}\,{\to}\mathrm{L}}{\varphi_1 \Rightarrow \varphi_2 \vdash \varphi_3 \Rightarrow \varphi_4}\,{\to}\mathrm{R}}{(\forall X : a \to a)\varphi_1 \Rightarrow \varphi_2 \vdash \varphi_3 \Rightarrow \varphi_4}\,\forall\mathrm{L}\ (\mathrm{rel})}{(\forall a)(\forall X : a \to a)\varphi_1 \Rightarrow \varphi_2 \vdash \varphi_3 \Rightarrow \varphi_4}\,\forall\mathrm{L}\ (\mathrm{obj})}{(\forall a)(\forall X : a \to a)\varphi_1 \Rightarrow \varphi_2 \vdash (\forall Y : a \to a)\varphi_3 \Rightarrow \varphi_4}\,\forall\mathrm{R}\ (\mathrm{rel})}{(\forall a)(\forall X : a \to a)\varphi_1 \Rightarrow \varphi_2 \vdash (\forall X : a \to a)(\forall Y : a \to a)\varphi_3 \Rightarrow \varphi_4}\,\forall\mathrm{R}\ (\mathrm{rel})}{(\forall a)(\forall X : a \to a)\varphi_1 \Rightarrow \varphi_2 \vdash (\forall a)(\forall X : a \to a)(\forall Y : a \to a)\varphi_3 \Rightarrow \varphi_4}\,\forall\mathrm{R}\ (\mathrm{obj})}{((\forall a)(\forall X : a \to a)\varphi_1 \Rightarrow \varphi_2) \Rightarrow ((\forall a)(\forall X : a \to a)(\forall Y : a \to a)\varphi_3 \Rightarrow \varphi_4)}\,{\to}\mathrm{R}$$

Figure 5.9: Derivation Tree for Example

By replacing $\varphi_1$, $\varphi_2$, $\varphi_3$ and, $\varphi_4$ by actual formulas we get:

$$\dfrac{\dfrac{X; X = X \wedge X^{\smile} = X \vdash X; X = X \wedge X^{\smile} = X}{X; X = X \wedge X^{\smile} = X, X; Y = Y \vdash X; X = X \wedge X^{\smile} = X}\,\mathrm{Weak}}{X; X = X \wedge X^{\smile} = X \wedge X; Y = Y \vdash X; X = X \wedge X^{\smile} = X}\,{\wedge}\mathrm{L}$$

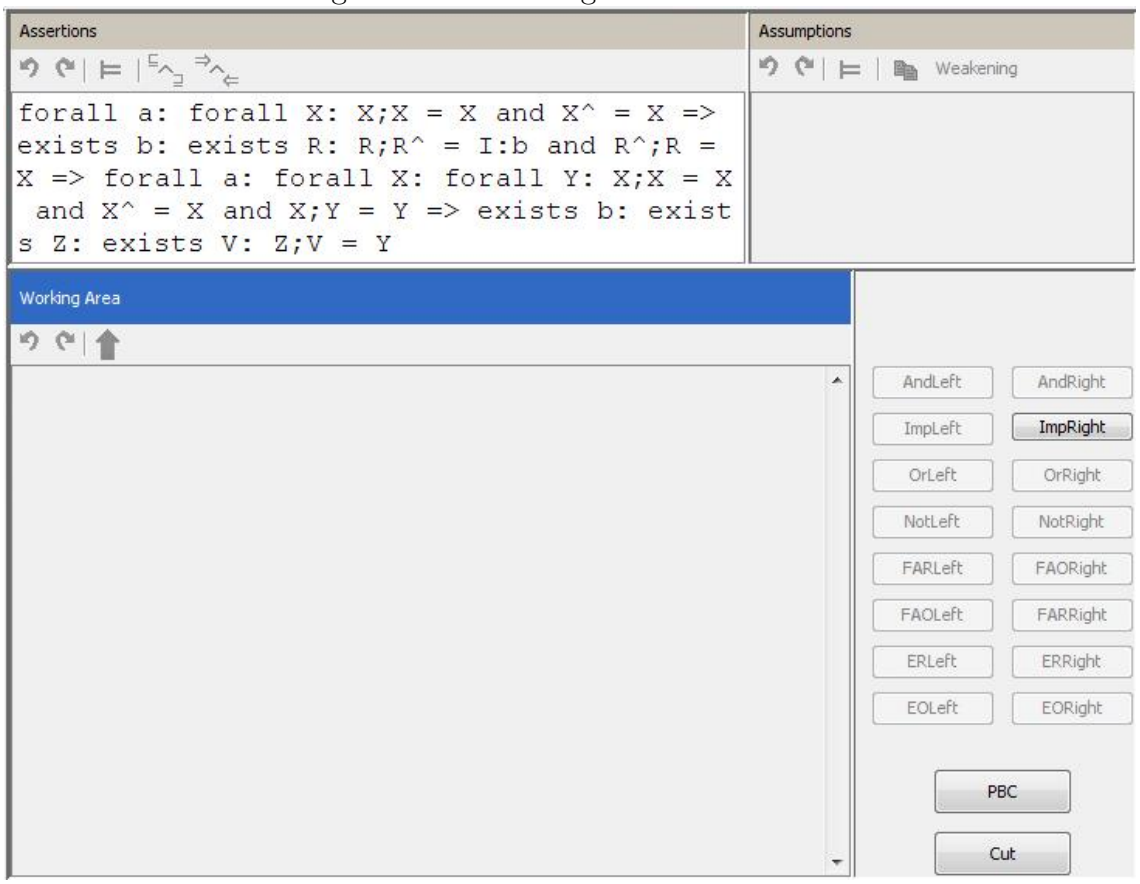
And:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{R^{\smile}; R = X \vdash R^{\smile}; R = X}{R; R^{\smile} = I_b, R^{\smile}; R = X, X; Y = Y \vdash R^{\smile}; R = X}\,\mathrm{Weak} \quad \dfrac{X; Y = Y \vdash X; Y = Y}{R; R^{\smile} = I_b, R^{\smile}; R = X, X; Y = Y \vdash X; Y = Y}\,\mathrm{Weak}}{R; R^{\smile} = I_b, R^{\smile}; R = X, X; Y = Y \vdash R^{\smile}; R; Y = Y}\,{=}\mathrm{L}}{R; R^{\smile} = I_b, R^{\smile}; R = X, X; Y = Y \vdash (\exists V : b \to a)R^{\smile}; V = Y}\,\exists\mathrm{R}\ (\mathrm{rel})}{R; R^{\smile} = I_b, R^{\smile}; R = X, X; Y = Y \vdash (\exists Z : a \to b)(\exists V : b \to a)Z; V = Y}\,\exists\mathrm{R}\ (\mathrm{rel})}{R; R^{\smile} = I_b, R^{\smile}; R = X, X; Y = Y \vdash (\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z; V = Y}\,\exists\mathrm{R}\ (\mathrm{obj})}{R; R^{\smile} = I_b, R^{\smile}; R = X, (X; X = X \wedge X^{\smile} = X), X; Y = Y \vdash (\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z; V = Y}\,\mathrm{Weak}}{R; R^{\smile} = I_b, R^{\smile}; R = X, (X; X = X \wedge X^{\smile} = X \wedge X; Y = Y) \vdash (\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z; V = Y}\,{\wedge}\mathrm{L}}{(R; R^{\smile} = I_b \wedge R^{\smile}; R = X), (X; X = X \wedge X^{\smile} = X \wedge X; Y = Y) \vdash (\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z; V = Y}\,{\wedge}\mathrm{L}}{(\exists R : b \to a)(R; R^{\smile} = I_b \wedge R^{\smile}; R = X), (X; X = X \wedge X^{\smile} = X \wedge X; Y = Y) \vdash (\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z; V = Y}\,\exists\mathrm{L}\ (\mathrm{rel})}{(\exists b)(\exists R : b \to a)(R; R^{\smile} = I_b \wedge R^{\smile}; R = X), (X; X = X \wedge X^{\smile} = X \wedge X; Y = Y) \vdash (\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z; V = Y}\,\exists\mathrm{L}\ (\mathrm{obj})$$

In that formula, the assumption requires that the allegory has splittings. This means that for every partial equivalence relation, there is the set of its equivalence classes. A partial equivalence relation $X$ is similar to an equivalence relation except that it is not required to be reflexive. The required object $b$ is the set of the existing equivalence classes and $R$ relates any such class with its elements in the set $a$. The conclusion of the formula says that if you have such a partial equivalence relation and a relation $Y$ that respects the equivalence classes ($X;Y = Y$), then there is a relation $V$ relating the equivalence classes in the same way as $Y$.

Let $\varphi_1 = X;X = X \wedge X^\smile = X$, $\varphi_2 = (\exists b)(\exists R : b \to a)(R;R^\smile = I_b \wedge R^\smile;R = X)$, $\varphi_3 = X;X = X \wedge X^\smile = X \wedge X;Y = Y$ and, $\varphi_4 = (\exists b)(\exists Z : a \to b)(\exists V : b \to a)Z;V = Y$. Figure 5.9 shows corresponding derivation steps verifying that implication. The first derivation is the starting point of the verification and the second and third derivations are left and right sub trees of the first derivation.

Figure 5.10: Creating a New Derivation

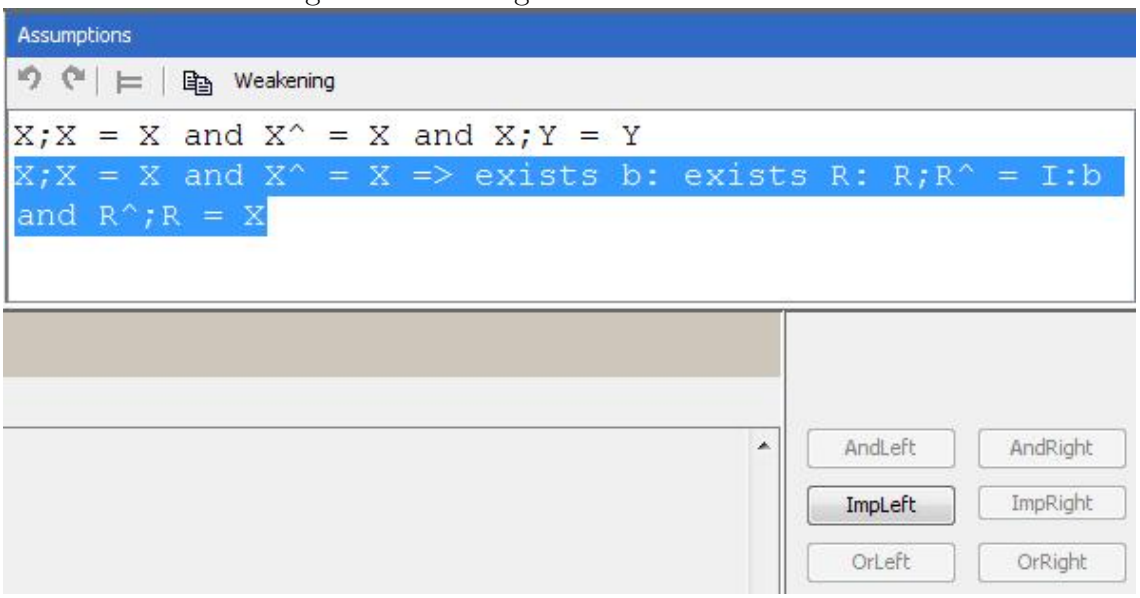


To start proof by RelAPS, we first need to enter the formula to the system. We can add it to the assertion window by using the 'Start Proof' button in the 'Proof Explore' window. Figure 5.10 shows the system after creating the initial proof obligation.

As shown in Figure 5.10 the only enabled 'Rule' buttons are 'ImplicationRight', 'CUT', and 'PBC' which are the only legal rules for the current situation. By pressing 'ImplicationRight' button $\rightarrow$R rule will be applied to the current state of the proof. The right hand buttons are automatically enabled and disabled according to the right hand formula of the current derivation. The application of other right hand buttons are almost similar except 'ExistsRelationRight' and 'ExistsObjectRight' which we will talk about later.

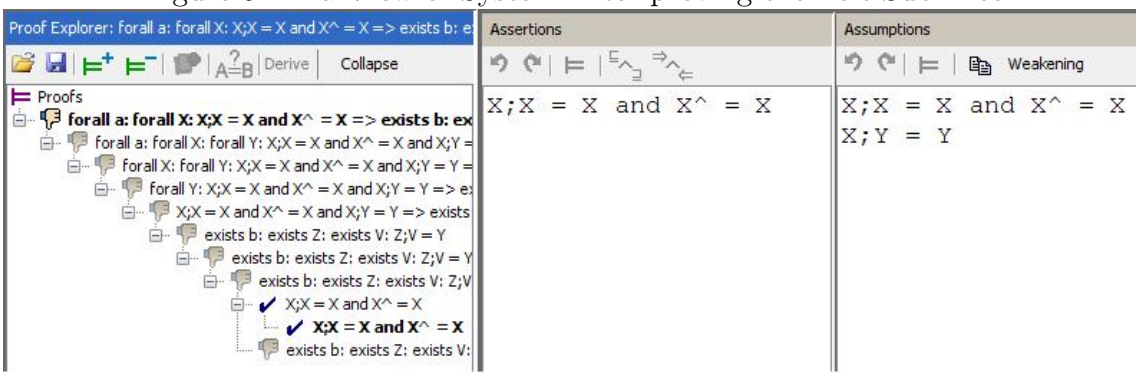
To enable left hand buttons, we have to select one of the formulas in the 'Assumption' window. Figure 5.11 shows how to select one of 'Assumption' window formulas enabling the proper rule button.

Figure 5.11: Using Left Hand Rule Buttons



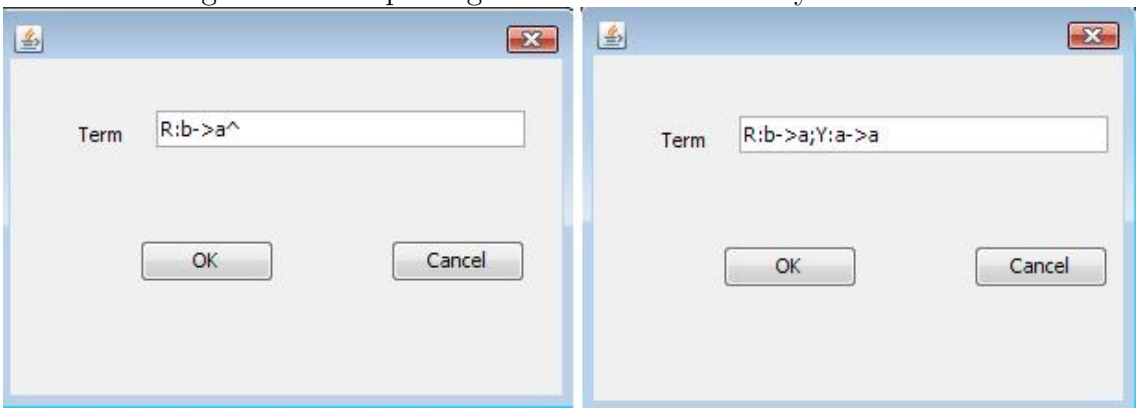

Some rules, like ($\rightarrow$L), create two subtrees. In order to complete the proof we need to provide derivations for both the left and the right subtree. As shown in Figure 5.12, after applying the ($\rightarrow$L) rule two subtrees have been created in 'Proof Explore'. Figure 5.12 also shows that the left subtree has been verified only in one step. This is because Axiom and Weakening rules are applied automatically by the system.

Figure 5.12: a View of System After proving the Left Sub Tree




When applying either one of the ∃R (obj), ∃R (rel), ∀L (obj) and ∀L (rel) rules, the user may need to replace a variable by a term. To do so, a new window will appear where the user can enter the new term. Note that if a relational term is being entered, the type of each relational variable has to be mentioned after it. Otherwise the system will not accept the new term. Figure 5.13 shows examples of acceptable relational terms.

Figure 5.13: Replacing Relational Variables by New Terms




As can be seen, there is no button for (=L) and (=R) rules. These rules can be applied by using the 'Working Area' window. Figure 5.14 shows the last step of our proof which is the application of (=L) rule.

Figure 5.14: Using Working Area Window for Applying (=L) Rule

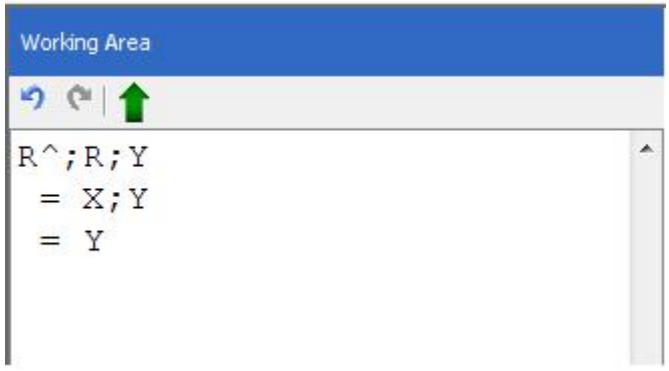
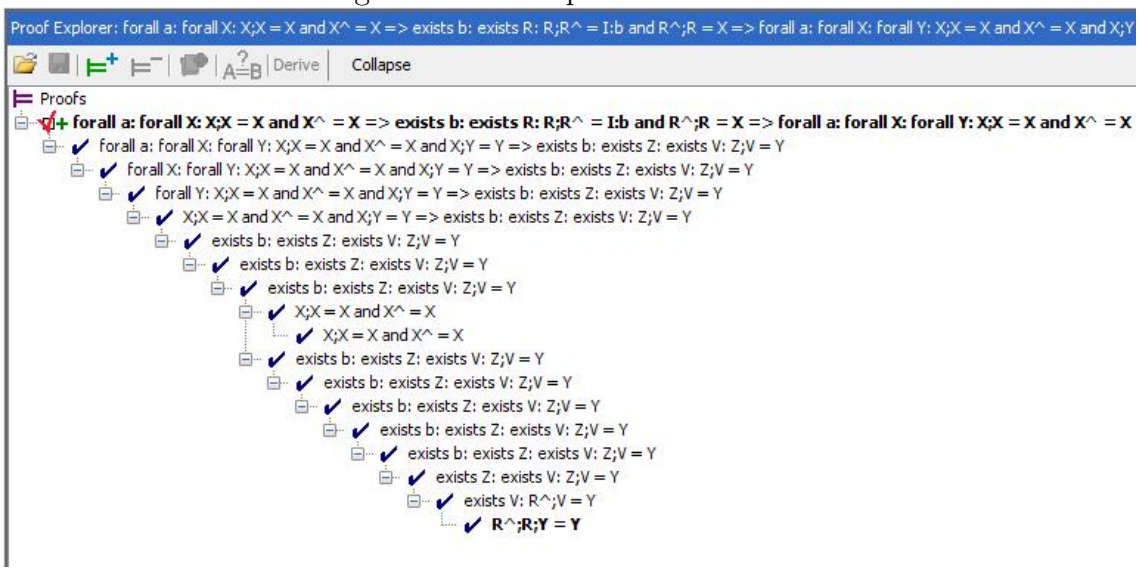



Figure 5.15 shows the 'Proof Explore' window after completion of our proof. The 'Tree View' shows the right hand of derivations. Left formulas of each derivation can be seen in the 'Assumption' window by selecting that derivation in the 'Proof Explore' tree.

Figure 5.15: Complete Tree of Proof



### 5.2.2 Implementation

The RelAPS system has been developed by the Java programming language within the NetBeans Interactive Development Environment. More specifically, version 1.5 of the Java Development Kit and version 4.1 of the NetBeans IDE has been used. In order to extend RelAPS to first order logic, we have created some new classes and also modified some existing classes of the code. In the following we review the main changes in the code.

The main changes have happened in the ProofFactory, GUI and Formula packages. The classes in the ProofFactory package handle the application of the rules while the GUI package handles necessary changes in the interface of the application. For each derivation rule, one new class has been created in the ProofFactory package as well as one button in the main frame of the application. For example, when user applies the $\wedge$R rule, first the 'ApplyRule' class, which is a static class in the ProofFactory package, splits the 'and' formula on the right hand side of the derivation into two formulas and creates two new proofs using that formulas. After that, it adds the assumptions of original formula to them and sends them back to GUI. Then the classes in the GUI package update tree view of the 'Proof Explorer' window and the 'Assumptions' and 'Assertion' window.

In the Formula package, a class have been already defined for each type of formulas, but some of their main methods like 'isTrivial', 'subFormulas', 'update' and 'equals' have not been fully implemented. Also to implement formulas that were not in the previous system language, like the ones containing $\perp$, some new classes have been added to the hierarchy of classes with root 'Formula' class. In addition, some methods have been modified for all classes to satisfy new requirements such as finding or replacing a variable.

Other packages like FormulaParser, Terms, Rules and Operations have had some minor modifications in order to be made compatible with other parts of the system. For example classes 'ObjectVar' and 'RelationalVar' have been added to Terms package, representing object and relational variables.

# Chapter 6

# Conclusions

In this chapter we want to briefly review the content of the thesis and compare it with other theorem proving systems. Then we will motivate and suggest some ideas for future work.

## 6.1 Summary and Related Works

We presented RelAPS, an interactive theorem proving system for relation algebra which accepts first order formulas. It provides an environment where a user can perform a relation-algebraic proof similar to doing it by hand. The main difference between RelAPS and most proving systems is that the language of the system is typed. The base theory of the system is the theory of allegories which provides identity relations on each object and a converse, an intersection and a composition operation for suitable typed relations. In addition, the system is capable of accepting new theories by combining user defined operations with arbitrary axioms. The system also includes a user friendly interface.

RelAPS is a stand alone system supporting manual proofs of theorems. Any proof step performed is correct by design. However, currently the system is not capable of any automatic derivation.

We started by introducing a formal language for allegories. We provided the syntax and semantics of the language. The language has two different kinds of terms; object terms and relational terms, where object terms are built from object constant symbols and object variables, and relational terms from typed relational constant symbols, typed relational variables, typed operation symbols and the regular operations available in any allegory. Then we introduced a first order logic calculus for relational categories. The calculus is formulated in a sequent style but with

exactly one formula on the right hand side. We also showed that the calculus is sound and complete. The soundness proof has been done by induction on the structure of the calculus rules. The completeness proof is based on Henkins' completeness proof. However, that proof had to be modified extensively since the language contains two different kinds of terms and, more importantly, it is typed. Finally we gave a brief overview about the system, how to use it and its implementation.

Many systems supporting theorem proving have been developed during the past years. As a typical example for a fully automated system, we have chosen Prover9 [13]. Prover9 is an automated theorem prover for first order and equational logic. Therefore, the language of the system is not typed. The typing contained in the theory of allegories could be modeled by partial operations within an untyped language. However, this would produce additional proof obligations verifying that all entities are well defined. Due to its typed language and the corresponding type checker of RelAPS this is already handled even before the user starts a proof. Prover9 also works fully automatically. This feature is limited by the complexity of the property to be verified, of course. The calculus used and the proofs generated are tailored for automatic proving. As a consequence they are usually very hard to read for a human being. RelAPS, on the other hand, uses a very intuitive version of natural deduction that mimics human reasoning very closely.

In another project, a semi-automated proof system has been developed for basic category-theoretic reasoning [11]. It is based on a first order sequent calculus that captures the basic properties of categories, functors and natural transformations as well as a small set of proof tactics that automate proof search in this calculus. Since it is a automated system, it has similar problems as Prover9. Typing is not part of its languages, hence, like Prover9, this would produce additional proof obligations. The system is also based on fixed theory and no additional operations or theories can be added to it. Therefore, it is not possible to work within the theory of allegories since the system only supports basic category theory.

RALF [2] was designed as a special purpose proof assistant for heterogeneous relation algebras with the goal of supporting proofs in a calculational style. Therefore, its basic motivation and design was similar to RelAPS. RALF has a graphical user interface which represents theorems as trees, i.e., every term is displayed as a tree where the leaves are constants and variables and the nodes are the relational operations. This makes some terms hard to read. Also, during the interactions, only the current sub goal is visible. The system is based on a fixed axiomatization; the axioms of a heterogeneous relation algebra. It is not possible to work with weaker and/or stronger theories within the system. Furthermore, it is no longer supported and there is no working version any longer available.

RALL [12] is another theorem proving system for heterogeneous relation algebra which has the ability of automatic proving for small theorems. It uses the Isabelle/HOL type system to support reasoning within abstract heterogeneous relation algebras with minimal effort. However, RALL limits itself to reasoning within representable relation algebras. The system works by translating relation-algebraic formulas into higher-order logic. As a consequence the system is incomplete. Moreover, this method cannot be applied to weaker structures like allegories. A further consequence of this method is that the proofs generated are proofs of the translated formulas within a fully automated system. One can easily imagine that they are extremely hard to read.

## 6.2 Future Work

There is plenty of further work that can be performed to develop the capabilities of this system and to improve its performance. The main focus in the future should be on automating some proof steps, particularly very basic steps. Certain sub theories of allegories, such as the equational theory, are known to be decidable. Once it has been implemented, the system could suggest to the user that the current proof obligation is in a certain sub theory that can be decided. If the user chooses to let the system finish the proof, the corresponding algorithm is used to find that proof.

More flexible user-defined operations are also another necessity. Currently there are no function definitions for object terms and the user only can define relational functions. In addition, function symbols could take a mixture of relational and object terms as parameters which would be useful for relational constructions such as splittings. User defined predicate symbols have not been implemented either, i.e., the only defined predicate symbols are '$<$', '$>$', and '$=$'. The user cannot specify anymore predicate symbols at the moment. As in Definition 6.3, user defined predicate symbols are part of our language, and hence implementing this feature is another essential future work.

Producing LaTeX output of the proofs is another possible project. A researcher could use the system to prove theorems while automatically being checked for correctness and use the LaTeX output for publications.

# Bibliography


[1] Asperti A., Longo G.: Categories, Types, and Structures. Foundation of Computing Series MIT Press (1991).

[2] Berghammer R., Hattensperger C., Schmidt G.: RALF:A Relation Algebraic Formula manipulation system and proof checker. *In:* Nivat M., Rattray C., Rus T., Scollo G. (Eds.), Proc. 3rd Conference on Algebraic Methodology and Software Technology, Workshops in Computing, Springer Verlag, 407-408 (1994).

[3] Enderton H.B.: Mathematical Introduction to Logic (2nd edition). Hartcourt Academic Press (2001).

[4] Freyd P., Scedrov A.: Categories, Allegories. North-Holland (1990).

[5] Gentzen G.: Untersuchungen über das logische Schließen I. Mathematische Zeitschrift 39 (2), 176-210 (1934).

[6] Gentzen G.: Untersuchungen über das logische Schließen II. Mathematische Zeitschrift 39 (3), 405-431 (1935).

[7] Glanfield J.: Towards Automated Derivation in the Theory of Allegories. MSc. Thesis, Brock University (2008).

[8] Glanfield J.: Relaps: A Proof Assistant for Relational Categories. http://www.joelglanfield.com/relaps.

[9] Gödel K.: Die Vollständigkeit der Axiome des logischen Funktionenkalküls. Monatshefte für Mathematik 37, 349-360 (1930).

[10] Henkin L.: The Completeness of the First-Order Functional Calculus. Journal of Symbolic Logic 14, 159-166 (1949).

[11] Kozen D., Kreitz C., Richter E.: Automating Proofs in Category Theory. IJCAR 2006, Springer Verlag, 392-407 (2006).

[12] Oheimb D.V., Gritzner T.F.: RALL: Machine Supported Proofs for Relation Algebra. *In:* W. McCune, ed., Conference on Automated Deduction, CADE 14, LNCS 1249, Springer, 380-394 (1997).

[13] McCune W.: Prover9: Automated Theorem Prover for First Order and Equational Logic. http://www.cs.unm.edu/∼mccune/prover9.

[14] Schmidt G., Hattensperger C., Winter M.: Heterogeneous Relation Algebras. *In:* Brink C., Kahl W., Schmidt G. (eds.), Relational Methods in Computer Science, Advances in Computing Science, Springer Vienna (1997).

[15] Schmidt G., Ströhlein T.: Relationen und Graphen. Springer (1989); English version: Relations and Graphs. Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoret. Comput. Sci., Springer (1993).

[16] Tarski A.: On the Calculus of Relations, J. Symbolic Logic 6, 73-89 (1941).

[17] Winter M.: A new Algebraic Approach to $L$-Fuzzy Relations Convenient to Study Crispness. INS Information Science 139, 233-252 (2001).

[18] Winter M.: Goguen Categories - A Categorical Approach to L-Fuzzy Relations. Trends in Logic 25, Springer (2007).