Negative operations on proofs and labels

Tatiana Yavorskaya (Sidon) [*]

June 6, 2005

Abstract

Logic of proofs $\mathcal{LP}$ introduced by S. Artemov in 1995 describes properties of proof predicate *"t is a proof of F"* in the propositional language extended by atoms of the form $[\![t]\!]F$. Proof are represented by terms constructed by three elementary recursive operations on proofs.

In order to make the language more expressive we introduce the additional *storage predicate* with the intended interpretation *"x is a label for F"*. It can play the role of syntax encoding, so it is useful for specification of operations that require formula arguments.

In this paper we study operations on proofs and labels that can be specified in the extended language. First, we give a formal definition of an operation on proofs and labels. Then, for an arbitrary finite set of operations $\mathcal{F}$ the logic $\mathcal{LPS}(\mathcal{F})$ is defined. We provide $\mathcal{LPS}(\mathcal{F})$ with symbolic semantics and arithmetical semantics.

The main result of the paper is the uniform completeness theorem for this family of logics with respect to the both types of semantics.

1 Introduction

In [1, 2], S. Artemov defined the Logic of Proofs $\mathcal{LP}$. It is formulated in the propositional language enriched by formulas of the form $[\![t]\!]F$ with the intended meaning *"t is a proof of F"*. Here t is a *proof term* which represent arithmetical proofs. Proof terms are constructed from *proof variables* and

[*]The work is partially supported by grants from Russian Foundation for Basic Research, Utrecht University and Netherlands Organization for Scientific Research (NWO).

proof constants by means of three functional symbols representing elementary computable operations on proofs. These operations are binary $\cdot$ and $+$ and unary !. The Logic of Proofs $\mathcal{LP}$ is axiomatized over propositional calculus by the weak reflexivity principle and axioms for operations "$\cdot$", "$+$" and "!"

$[\![t]\!]A \to A$	*weak reflexivity*
$[\![t]\!](A \to B) \to ([\![s]\!]A \to [\![t \cdot s]\!]B)$	*application*
$[\![t]\!]A \to [\![t+s]\!]A, \quad [\![s]\!]A \to [\![t+s]\!]A$	*nondeterministic choice*
$[\![t]\!]A \to [\![!t]\!][\![t]\!]A$	*positive proof checker*

and *axiom necessitation rule*, which allows to specify proof constants as proofs of the concrete axioms

$$\frac{}{[\![a]\!]A}, \quad \text{where } a \text{ is an axiom constant, } A \text{ is an axiom of } \mathcal{LP}.$$

The intended semantics for $\mathcal{LP}$ is formalized in Peano Arithnmetic **PA**; $[\![t]\!]F$ is interpreted by an arithmetical proof predicate which numerates theorems of **PA**. It is proved in [2] that $\mathcal{LP}$ is complete with respect to arithmetical interpretations based on multi-conclusion proof predicates (i.e. those where a proof can prove more than one proposition). It is also shown that the modal counterpart of $\mathcal{LP}$ is Gödel's provability logic $S4$. This fact gives possibility to provide $S4$ and, therefore, intuitionistic logic with the exact provability semantics.

In this paper we consider the language of $\mathcal{LP}$ as a convenient framework in which operations on proofs can be described and studied. To begin, let us look what operations on proofs can be represented by $\mathcal{LP}$-terms. The answer is given in [1]. It is proved there that terms constructed by $\{\cdot, +, !\}$ together with proof constants suffices to represent all positive operations on arithmetical proofs which allow description in the language of $\mathcal{LP}$. All these operations have proofs both as the arguments and the result, that is, they have the type

$$\underbrace{\mathit{proof} \times \ldots \times \mathit{proof}}_{n} \to \mathit{proof}.$$

Below are examples of other natural operations, which provide proofs, but require arguments of other types.

Negative proof checker. If we interpret $\Box$ as provability in Peano Arithmetic then the following principle is valid $\neg[\![p]\!]A \to \Box\neg[\![p]\!]A$. One can write down precisely the quantifier hidden in the provability operator $\Box$ and obtain the following principle (with the parameters p, A) $\exists q\, (\neg[\![p]\!]A \to [\![q]\!]\neg[\![p]\!]A)$.

The standard skolemization procedure gives the operation called *negative proof checker* which depends on two arguments: a proof p and a formula A. This operation $\mathsf{check}^- : \mathit{proof} \times \mathit{proposition} \rightarrow \mathit{proof}$ is specified as follows:

$$\neg [\![p]\!] A \rightarrow [\![\mathsf{check}^-(p, A)]\!] \neg [\![p]\!] A.$$

Axiom verifier. Let us choose an axiom of propositional logic, for example $A \rightarrow (B \rightarrow A)$. The formula $\Box(A \rightarrow (B \rightarrow A))$ is valid under every arithmetical interpretation. The corresponding Skolem function depends on two formula arguments A and B. That is, we can specify the operation ax_1, called *axiom verifier*, of the type $\mathsf{ax}_1 : \mathit{proposition} \times \mathit{proposition} \rightarrow \mathit{proof}$ by the principle

$$[\![\mathsf{ax}_1(A, B)]\!](A \rightarrow (B \rightarrow A)).$$

If we take a more nontrivial valid principle instead of the propositional axiom, then we obtain a more interesting operation which depends on formula arguments only. For example, the operation which finds the proof of weak reflexivity principle can be specified as follows

$$[\![\mathsf{ax}_r([\![p]\!] A)]\!]([\![p]\!] A \rightarrow A).$$

The arithmetical proof of the weak reflexivity is not trivial, while in $\mathcal{LP}$ it is represented by a proof constant, which erases all information about this proof.

These examples provide motivation to develop the language, which is able to specify operations of the type

$$\underbrace{\mathit{proof} \times \ldots \times \mathit{proof}}_{n} \times \underbrace{\mathit{proposition} \times \ldots \times \mathit{proposition}}_{k} \rightarrow \mathit{proof}.$$

The approach taken in [9] is to extend propositional language by two predicates: the proof predicate of $\mathcal{LP}$ *"p is a proof of F"* and the additional storage predicate *"x is a label (or code) for F"* denoted by the formula $x \ni F$. So, we introduce a new sort of objects which is called *labels.* We assume that every label is assigned to a finite set of propositions. Now we can specify operations which depend on both proofs and labels. We are going to use labels everywhere when we need formula arguments. For example, the operations of negative proof checker and reflexivity checker can be specified as follows:

$$x \ni A \wedge \neg [\![p]\!] A \rightarrow [\![\mathsf{check}^-(p, x)]\!] \neg [\![p]\!] A$$
$$(x \ni [\![p]\!] A) \rightarrow [\![\mathsf{ax}_r(p, x)]\!]([\![p]\!] A \rightarrow A).$$

Our approach is related to the one considered in [3, 4, 5, 6]. In these papers single-conclusion proof predicates are studied, namely, those predicates for which every proof may prove at most one formula. In [3, 6] it is suggested to consider single-conclusion proofs as pointers (labels) for propositions: if t is a (single-conclusion) proof of A, then one can use t as a label for A. In [6] the language is enriched with special constructions which allow to carry out pattern matching.

This paper continues the research started in [9] in which positive operations on proofs and labels were considered, namely, those which use only positive information about proofs and labels.

The structure of our paper is the following. In section 2 we define what an operation on proofs and labels is. In 2.1 we define the language $\mathcal{L}_0$ in which operations on proofs will be specified and describe two types of semantics (symbolic and arithmetical) for $\mathcal{L}_0$. In 2.2 we briefly remind results from [9] concerning the minimal logic of proofs and labels $\mathcal{LPS}_0$. It describes the most general properties of storage and proof predicates and does not contain operations on proofs, however, $\mathcal{LPS}_0$ provides us with the framework for description of operations on proofs and labels. In 2.3 specifications of an operation on proofs are defined semantically. The syntactical criteria for a formula to specify an operation is proved; the set of specifications is shown to be decidable.

In section 3 we study logics which describe operations specified by schemas in the language of $\mathcal{LPS}_0$. In 3.1 we define logics $\mathcal{LPS}(\mathcal{F})$ where $\mathcal{F}$ stands for a finite collection of operations. We describe symbolic and arithmetical semantics for these logics and prove soundness with respect to both types of semantics. Section 3.2 is devoted to the proof of completeness of the logic $\mathcal{LPS}(\mathcal{F})$ with respect to symbolic semantics. As a corollary of this theorem we obtain decidability and the complexity bounds, namely, satisfiability problem for all these logics is NP-complete. In 3.3 we prove arithmetical completeness of $\mathcal{LPS}(\mathcal{F})$. The both completeness theorems are uniform on the choice of $\mathcal{F}$.

2 Specification of operations on proofs and labels

2.1 Specification language $\mathcal{L}_0$ and its semantics

Definition 2.1. The language $\mathcal{L}_0$ contains objects of three sorts: *propositions*, *labels* and *proofs*. It consists of variables of every sort:

propositional variables $SVar = \{S_1, S_2, \ldots\}$
label variables $LVar = \{x_1, x_2, \ldots\}$
proof variables $PVar = \{p_1, p_2, \ldots\}$,

boolean connectives and two operational symbols

$[\![\cdot]\!](\cdot)$ of the type *proof* $\to$ (*proposition* $\to$ *proposition*)
$(\cdot) \ni (\cdot)$ of the type *label* $\to$ (*proposition* $\to$ *proposition*).

Formulas are constructed from sentence variables by means of boolean connectives and according to the rule: if p is a proof variable, x is a label variable and F is a formula then $x \ni F$ and $[\![p]\!]F$ are formulas too. Formulas of the form $x \ni F$ and $[\![p]\!]F$ are called *q-atomic.*

For any formula F by $SVar(F)$, $PVar(F)$ and $LVar(F)$ we denote the sets of all sentence variables, proof variables and label variables from F respectively; $Var(F)$ stands for $SVar(F) \cup PVar(F) \cup LVar(F)$.

Informally speaking, the language $\mathcal{L}_0$ is supposed to describe the structure containing objects of three types: *propositions* (represented by formulas), *labels* (or names of propositions) (represented by label variables) and *proofs* (represented by proof variables). The structure is supplied with two predicates: a *storage predicate* between labels and propositions "x is a label for a proposition F" (denoted by the formula $x \ni F$) and a *proof predicate* "p is a proof of a proposition F" (represented by the formula $[\![p]\!]F$).

We assume that both the storage predicate $x \ni F$ and the proof predicate $[\![p]\!]F$ are recursive relations. We suppose that a finite set of propositions is assigned to every label, the function that recovers the corresponding set being given a label is total recursive, and there exists a total recursive function which for every proposition returns a label assigned to it. For every proof the set of propositions proved by it is finite and the function that maps proofs to the corresponding sets is total recursive. We are going to specify operations on proofs and labels in the language $\mathcal{L}_0$; in the informal semantics they correspond to total recursive functions which have labels and proofs as their arguments and return proofs.

Now let us give the formal definition of the symbolic semantics for the language $\mathcal{L}_0$.

Definition 2.2. A *symbolic model* $\mathcal{M} = (\#, v)$ consists of two objects: a binary relation $\# \subseteq (PVar \cup LVar) \times Fm$ between label and proof variables and formulas and a truth evaluation of sentence variables $v : SVar \mapsto \{true,\ false\}$. Definition of the truth relation $\mathcal{M} \models A$ is inductive: for propositional variables $\mathcal{M} \models S$ iff $v(S) = true$, $\models$ commutes with boolean connectives and

$$\mathcal{M} \models (x \ni A) \ \rightleftharpoons\ \#(x, A); \quad \mathcal{M} \models [\![p]\!]A \ \rightleftharpoons\ \#(p, A) \text{ and } \mathcal{M} \models A.$$

A model $\mathcal{M} = (\#, v)$ is called *finite* if the relation $\#$ is finite and evaluation v is primitive recursive.

Now we are going to formalize the intended semantics for $\mathcal{L}_0$ in Peano Arithmetic. We will interpret proof and storage operators by proof and storage predicates respectively.

Definition 2.3. A *storage predicate* $Str(x, y)$ is an arithmetical Δ_1 formula such that for every arithmetical sentence ψ there exists x such that $Str(x, \psi)$, for every n the set $Cont(n) \rightleftharpoons \{m|\ Str(n, m)\}$ is finite and the function $\lambda n.\ulcorner Cont(n) \urcorner$ is total recursive.

A *proof predicate* Prf is an arithmetical Δ_1 formula such that for every arithmetical formula φ one has

$$\mathbf{PA} \vdash \varphi \ \text{ iff } \ \text{ there exists } \ n \text{ such that } Prf(n, \ulcorner \varphi \urcorner),$$

A proof predicate Prf is *single-conclusion* if $Prf(x, y_1) \wedge Prf(x, y_2) \rightarrow y_1 = y_2$ or *multi-conclusion* otherwise.

We say that Prf is normal if for every n the set $Th_{Prf}(n) \rightleftharpoons \{m \mid Prf(n, m)\}$ is finite and the function $\lambda n. Th_{Prf}(n)$ is total recursive. If Prf is multi-conclusion then we also require that for every finite set of arithmetical theorems Γ there exists a natural number n such that $\Gamma \subseteq Th_{Prf}(n)$.

Example 2.4. The standard Gödel proof predicate and its natural multi-conclusion version are arithmetical formulas $proof(x, y)$ and $Proof(x, y)$ which express the relations *proof*(x, y) $\rightleftharpoons$ *"x is the Gödel number of a derivation in* **PA***, y is the Gödel number of the last formula in this derivation"*

Proof(x, y) $\rightleftharpoons$ *"x is the Gödel number of a finite set of* **PA***-derivations, y is the Gödel number of the last formula in one of them".*

The following two formulas are storage predicates; the second one is functional in x.

Store(x, y) $\rightleftharpoons$ *"x is the Gödel number of a finite set of formulas, y is the Gödel number of one of them".*

Store$_1(x, y)$ $\rightleftharpoons$ *"x is the Gödel number of a formula and* $y = x$*".*

Comment 2.5. Without loss of generality we may assume that from $\mathit{Str}(x, y)$ it follows that y is a Gödel number of an arithmetical sentence and the same holds for *Prf*. For every storage predicate *Str* the function $\iota x.\mathit{Str}(x, y)$ which for every arithmetical sentence φ returns x such that $\mathit{Str}(x, \ulcorner\varphi\urcorner)$ is total recursive. For every proof predicate *Prf* the similar function $\iota x.\mathit{Prf}(x, y)$ is recursive but not total.

Definition 2.6. An *arithmetical interpretation* $* = (\mathit{Str}, \mathit{Prf}, (\cdot)^*)$ for the language $\mathcal{L}_0$ has the following parameters:

- a storage predicate *Str* and a normal proof predicate *Prf*;
- an evaluation $(\cdot)^*$ that assigns natural numbers to label and proof variables and arithmetical sentences to propositional variables.

We extend evaluation $(\cdot)^*$ to all $\mathcal{L}_0$-formulas in the following way: it commutes with the boolean connectives and

$$\begin{array}{lcl} (\llbracket p \rrbracket A)^* & \rightleftharpoons & \exists x\ (1 = 1 \wedge x = \ulcorner A^* \urcorner \wedge \mathit{Prf}(p^*, x)), \\ (l \ni A)^* & \rightleftharpoons & \exists x\ (2 = 2 \wedge x = \ulcorner A^* \urcorner \wedge \mathit{Str}(l^*, x)). \end{array}$$

Comment 2.7. Note that $\mathbf{PA} \vdash (\llbracket p \rrbracket A)^* \leftrightarrow \mathit{Prf}(p^*, \ulcorner A^* \urcorner)$ and, similarly, $\mathbf{PA} \vdash (l \ni A)^* \leftrightarrow \mathit{Str}(l^*, \ulcorner A^* \urcorner)$. We interpret the proof and storage predicates in the more sophisticated way because it makes the following problem decidable: being given *Prf*, *Str*, an arithmetical formula φ and an $\mathcal{L}_0$-formula F, decide whether there exists $(\cdot)^*$, such that $F^* = \varphi$. The values v^* for $v \in \mathit{Var}(F)$ are unique and can be found effectively.

2.2 The minimal logic of proofs and labels $\mathcal{LPS}_0$

The logic $\mathcal{LPS}_0$ is very basic. However, when we turn to specification of operations in the language $\mathcal{L}_0$, the logic $\mathcal{LPS}_0$ helps to formulate a simple criterion for a formula to specify an operation and to provide all the parameters for it (see the next subsection for details). The logics which describe operations on proofs (see definition 3.1) are extensions of $\mathcal{LPS}_0$.

Definition 2.8. The minimal logic of proofs and labels $\mathcal{LPS}_0$ is axiomatized by the following schemas (A is a formula, $p \in PVar$):

A0	classical propositional axioms	
A1	$[\![p]\!]A \to A$	*weak reflexivity*

The sole rule of inference is *modus ponens.*

The following results about $\mathcal{LPS}_0$ were proved in [9].

Theorem 2.9. $\mathcal{LPS}_0$ is sound and complete with respect to symbolic semantics, that is, for every formula A
1) if $\mathcal{LPS}_0 \vdash A$ then for every model $\mathcal{M}$ one has $\mathcal{M} \models A$;
2) if $\mathcal{LPS}_0 \not\vdash A$ then there exists a finite model $\mathcal{M}$ such that $\mathcal{M} \not\models A$.

Theorem 2.10. $\mathcal{LPS}_0$ is decidable. Satisfiability problem for $\mathcal{LPS}_0$ is NP-complete.

Theorem 2.11. (*Arithmetical soundness and completeness.*)
For every $\mathcal{LPS}_0$ formula A the following three propositions are equivalent:
1) $\mathcal{LPS}_0 \vdash A$;
2) for every interpretation $*$, $\mathbf{PA} \vdash A^*$;
3) for every interpretation $*$, A^* is true.

The completeness part of the last theorem asserts that if $\mathcal{LPS}_0 \not\vdash A$ then there exists $* = (Prf, Str, (\cdot)^*)$ such that A^* is false. Below we prove the stronger version of the arithmetical completeness for $\mathcal{LPS}_0$, namely, we show that Prf and Str can be taken independent on the choice of A.

Theorem 2.12. (*Uniform arithmetical completeness.*) There exist predicates Prf_0, Str_0 such that for every formula A if $\mathcal{LPS}_0 \not\vdash A$ then A^* is false for some $* = (Prf_0, Str_0, (\cdot)^*)$.

Proof. Let us define Str_0 and Prf_0. We fix an injective Gödel numbering of the joint syntax of $\mathcal{LPS}_0$ and **PA**. Consider all finite symbolic models for $\mathcal{LPS}_0$. Every such model $\mathcal{M}$ is defined by two finite objects: a relation $\#$ and a truth evaluation v. Therefore, one can numerate finite models by natural numbers in such a way that the relation "$\mathcal{M}_i \models A$" is primitive recursive in i and A.

For every i we define the evaluation $(\cdot)^{*i}$ as follows. Let ξ_i be the i-th prime number. For every proof variable p, label variable x and sentence variable S_j we put

$$p^{*i} \rightleftharpoons (\xi_i)^{\ulcorner p \urcorner}, \qquad x^{*i} \rightleftharpoons (\xi_i)^{\ulcorner x \urcorner}, \qquad S_j^{*i} \rightleftharpoons \begin{cases} j = j, & \text{if } v_i(S_j) = \mathit{true} \\ j = 0, & \text{otherwise} \end{cases}$$

if Prf and Str are given then for every formula B and number i the formula B^{*i} is defined. Moreover, the function which produces $\ulcorner B^{*i} \urcorner$, being given Prf, Str, i and $\ulcorner B \urcorner$, is primitive recursive.

Now the appropriate storage and proof predicates can be defined simultaneously by the multiple fixed point equation. Let $Proof$ and $Store$ be predicates from the example 2.4; without loss of generality we assume that if $Store(x, y) \vee Proof(x, y)$ is true then x is not a power of a prime number. The following formulas are provable in **PA**

$$\begin{aligned} Prf_0(x, y) \quad &\leftrightarrow \quad Proof(x, y) \vee \\ &\qquad \text{“}x = (\xi_i)^{\ulcorner p \urcorner} \text{ and } y = \ulcorner B^{*i} \urcorner \text{ and } \mathcal{M}_i \models [\![p]\!]B\text{”} \\ Str_0(x, y) \quad &\leftrightarrow \quad Store(x, y) \vee \\ &\qquad \text{“}x = (\xi_i)^{\ulcorner l \urcorner} \text{ and } y = \ulcorner B^{*i} \urcorner \text{ and } \mathcal{M}_i \models l \ni B\text{”} \end{aligned}$$

Note that Prf_0, Str_0 are Δ_1-formulas. It is easy to see that the interpretation $*i$ is injective, that is, for different formulas B and C we have $B^{i*} \neq C^{*i}$.

Let us prove that for every formula C and number i

$$\begin{array}{lcl} \mathcal{M}_i \models C & \Rightarrow & \mathbf{PA} \vdash C^{*i}; \\ \mathcal{M}_i \not\models C & \Rightarrow & \mathbf{PA} \vdash \neg C^{*i}. \end{array} \tag{1}$$

Induction on the formula C. The case of propositional variables and boolean connectives is trivial. Suppose that C has the form $[\![p]\!]B$. If $\mathcal{M}_i \models C$ then by the definition of Prf_0 we conclude that $\mathbf{PA} \vdash Prf_0((\xi_i)^{\ulcorner p \urcorner}, \ulcorner B^{*i} \urcorner)$ that is $\mathbf{PA} \vdash C^{*i}$. If $\mathcal{M}_i \not\models C$ then $\mathbf{PA} \vdash \neg Prf_0((\xi_i)^{\ulcorner p \urcorner}, \ulcorner B^{*i} \urcorner)$ because of the injectivity of the evaluation $(\cdot)^{*i}$. The remaining case when C has the form $x \ni B$ can be treated similarly.

It is clear that Str_0 is a storage predicate. Let us prove that Prf_0 is a normal proof predicate. Since for every i the model $\mathcal{M}_i$ is finite we conclude that for every n the set $Th(n)$ is finite. The function $n \mapsto Th(n)$ is primitive recursive due to the choice of enumeration $\mathcal{M}_i$. The only thing we need to verify is that if $Prf_0(n, \ulcorner\varphi\urcorner)$ is true then $\mathbf{PA} \vdash \varphi$. Suppose that $Prf_0(n, \ulcorner\varphi\urcorner)$. By the definition of Prf_0 there are two possibilities. The first case, when $Proof(n, \ulcorner\varphi\urcorner)$ holds, is trivial. In the second case we have $n = (\xi_i)^{\ulcorner p \urcorner}$, $\varphi = B^{*i}$ and $\mathcal{M}_i \models [\![p]\!]B$. Then $\mathcal{M}_i \models B$ whence $\mathbf{PA} \vdash B^{*i}$ by (1).

Finally, consider a formula A such that $\mathcal{LPS}_0 \nvdash A$. By theorem 2.9 there exists a finite symbolic model $\mathcal{M}_i = (\#_i, v_i)$ such that $\mathcal{M}_i \not\models A$. Take $* = (Prf_0, Str_0, (\cdot)^{*i}$; we conclude that $\mathbf{PA} \vdash \neg A^*$. ■

2.3 Specification of operations in $\mathcal{L}_0$

Our purpose is to formalize the general idea of what an operation on proofs and labels is. We start with the admissible inference rules, describes by the figures $A_1, \ldots, A_n \vdash F$ where all A_i and F are formulas of $\mathcal{L}_0$. This rule is admissible if for every arithmetical interpretation $*$ the formula F^* is provable as soon as all A_i^* are provable. We can describe this rule by a formula $\Box A_1 \wedge \ldots \wedge \Box A_n \rightarrow \Box F$ where $\Box$ is interpreted as a provability operator $\exists x Prf(x, \cdot)$. If we remove existential quantifiers from $\Box$'s in hypothesis then we obtain the formula $[\![p_1]\!]A_1 \wedge \ldots \wedge [\![p_n]\!]A_n \rightarrow \Box F$ which is true under every arithmetical interpretation. This formula specifies the operation which recovers a proof of F being given proofs of all A_i and possibly some additional information.

Our definition of the operation on proofs and labels extends the example considered above in three ways. Firstly, we add hypothesis of the form $x_i \ni C_i$ in order to describe more operations, for example, those which need formula arguments (see example 2.15). Secondly, we allow negative formulas of the form $\neg[\![q_i]\!]B_i$ and $\neg(y_i \ni D_i)$ in the hypothesis, so our operations may use negative information. Thirdly, we admit a kind of choice: if we can obtain F using one of several sets of hypothesis $\delta_1, \ldots, \delta_k$, then we allow to describe an operation of the form $\delta_1 \vee \ldots \vee \delta_k \rightarrow \Box F$. Finally, we try to choose the definition invariant with respect to the particular choice of Prf, Str.

Roughly speaking, we call a specification of an operation on proofs and labels every arithmetical valid formula of the form $\delta \to \Box F$ where δ is an arbitrary boolean combination of q-atomic formulas. However, in order to describe all parameters of the operation precisely, it is more convenient to have the formula δ in the disjunctive normal form with respect to its q-atomic subformulas. We will use special notation for the conjunction in the DNF for δ and for parts of these conjunctions, corresponding to positive and negative assumptions about proofs and labels. These notation are describes below.

Let n be a natural number and $0 \le i \le n$. For every i let us fix natural numbers a_i, b_i, c_i and d_i. Then, we fix four lists of formulas in the language $\mathcal{L}_0$ of the lengthes a_i, b_i, c_i and d_i respectively:

$$A_{i,1}, \dots, A_{i,a_i}, \quad B_{i,1}, \dots, B_{i,b_i}, \quad C_{i,1}, \dots, C_{i,c_i}, \quad D_{i,1}, \dots, D_{i,d_i}.$$

two lists of proof variables of the lengthes a_i and b_i

$$p_{i,1}, \dots, p_{i,a_i}, \quad q_{i,1}, \dots, q_{i,b_i},$$

and two lists of label variables of the lengthes c_i and d_i

$$x_{i,1}, \dots, x_{i,c_i}, \quad y_{i,1}, \dots, y_{i,d_i}$$

We admit that some variables of the same type may coincide. By $\overrightarrow{pq}$ and $\overrightarrow{xy}$ we denote the lists of all variables p_{ij}, q_{ij} (x_{ij}, y_{ij} resp.) without repetitions. If Σ is a set of formulas, then $\neg\Sigma$ stands for the set of negations of formulas from Σ. We define $\delta_i \rightleftharpoons \delta_i^+ \cup \neg\delta_i^-$ where

$$\begin{array}{lll@{\qquad}lll}
\delta_i^+ & \rightleftharpoons & \pi_i^+ \cup \lambda_i^+, & \delta_i^- & \rightleftharpoons & \pi_i^- \cup \lambda_i^- \\
\pi_i^+ & \rightleftharpoons & \{[\![p_{ij}]\!]A_{ij} \mid j \le a_i\} & \pi_i^- & \rightleftharpoons & \{[\![q_{ij}]\!]B_{ij} \mid j \le b_i\} \\
\lambda_i^+ & \rightleftharpoons & \{x_{ij} \ni C_{ij} \mid j \le c_i\} & \lambda_i^- & \rightleftharpoons & \{y_{ij} \ni D_{ij} \mid j \le d_i\}
\end{array} \tag{2}$$

Definition 2.13. Let δ_i be sets of formulas described above and let F be an $\mathcal{L}_0$-formula. A *specification of an operation on proofs and labels* is a formula of the form

$$\bigvee_{i=1}^{n} (\bigwedge \delta_i) \to \Box F, \tag{OP}$$

which is true under every arithmetical interpretation where $\Box F$ is interpreted as the provability formula $\exists x\ Prf(x, \ulcorner F^* \urcorner)$.

Comment 2.14. By theorem 3.14 the formula (OP) is a specification iff $\mathcal{LPS}_0 \vdash \bigvee_i (\bigwedge \delta_i) \to F$. Therefore, the set of specifications is decidable.

Example 2.15. The basic operations of the logic of proofs $\mathcal{LP}$ correspond to the following specifications of operations in the sense of definition 2.13

$[\![p]\!](S_1 \to S_2) \wedge [\![q]\!]S_1 \to \Box S_2$	*application*
$[\![p]\!]S \vee [\![q]\!]S \to \Box S$	*nondeterministic choice*
$[\![p]\!]S \to \Box [\![p]\!]S$	*positive proof checker.*

We can also specify other operations, for example

$(x \ni S_1) \wedge (y \ni S_2) \to \Box(S_1 \to (S_2 \to S_1))$	*axiom verifier.*
$\neg [\![p]\!]S \to \Box \neg [\![p]\!]S$	*negative proof checker*
$[\![p]\!]S_1 \vee [\![q]\!]S_2 \to \Box(S_1 \vee S_2)$	*disjunction.*

Comment 2.16. Definition 2.13 says that for every arithmetical interpretation $*$ formula F^* is provable as soon as $\bigvee_i (\wedge \delta_i)^*$ holds. So, the formula (OP) specifies the algorithm which recovers the proof of F^* under the assumption that one of $(\wedge\delta_i)^*$ is true. This algorithm works as follows. Given $(\mathit{Prf}, \mathit{Str}, (\cdot)^*)$, reconstruct F^* and A^* for all $A \in \bigcup_i \delta_i$. Check whether $(\bigvee_i (\wedge\delta_i))^*$ is true (this problem is decidable since $(\bigvee_i(\wedge\delta_i))^*$ is a Δ_1 formula). If no then return zero. If yes then consequently try $x = 0, 1, 2, \ldots$ until $\mathit{Prf}(x, \ulcorner F^* \urcorner)$ is true. Return the resulting x. The algorithm terminates since $\exists x \mathit{Prf}(x, \ulcorner F^* \urcorner)$ is true.

Let us find out what are the parameters of the algorithm. In general, these are all the formulas from $\bigcup_i \delta_i$ and F. However, our purpose is to describe operations in such a way that the parameters of the corresponding algorithms are proofs and labels only, and they can be recovered in a uniform way from the syntax analysis of (OP) without taking the semantics into account. Namely, we would like to restrict the parameters by the list $\overrightarrow{pq}, \overrightarrow{xy}$. This idea is formalized in the following definition.

Definition 2.17. A specification (OP) is called *complete* if for every normal multi–conclusion predicate *Prf* and storage predicate *Str* there exists a total recursive function f^* of arity $|\overrightarrow{pq}| + |\overrightarrow{xy}|$ such that for every interpretation $(\mathit{Prf}, \mathit{Str}, (\cdot)^*)$, the following formula is true

$$\bigvee_i (\bigwedge \delta_i)^* \to \mathit{Prf}(f^*(\overrightarrow{pq}^*, \overrightarrow{xy}^*), \ulcorner F^* \urcorner) \tag{3}$$

In this case a *specification axiom for the operation f* is the following formula in the extension of the language $\mathcal{L}_0$ by a new functional symbol f:

$$\bigvee_i (\bigwedge \delta_i) \to [\![f(\overrightarrow{pq}, \overrightarrow{xy})]\!]F. \qquad \mathit{Spec}(f)$$

The *arity* of (OP) and f is a pair of natural numbers (n_1, n_2) where n_1 and n_2 are the numbers of variables in the lists $\overrightarrow{pq}$ and $\overrightarrow{xy}$ respectively. We say that the pair *Prf*, *Str* is *compatible with Spec*(f) and function f^* *satisfies Spec*(f) *for Prf*, *Str* if for every $(\cdot)^*$ formula (3) is true.

Comment 2.18. The definition of a complete specification says that we can specify precisely the parameters of the algorithm which reconstructs the proof of F. Now we are going to formulate and prove a syntactical criteria for the operation to be complete. Before we do that let us look at the examples of complete and non-complete specifications.

All operations of $\mathcal{LP}$ and axiom verifier from example 2.15 are complete. The corresponding specification axioms are the following formulas (we keep the notation of $\mathcal{LP}$ for the first three operations)

$$\begin{array}{l}
[\![p]\!](S_1 \to S_2) \wedge [\![q]\!]S_1 \to [\![p \cdot q]\!]S_2 \\
[\![p]\!]S \vee [\![q]\!]S \to [\![p + q]\!]S_2 \\
[\![p]\!]S \to [\![!p]\!][\![p]\!]S \\
(x \ni S_1) \wedge (y \ni S_2) \to [\![\mathsf{ax}(x, y)]\!](S_1 \to (S_2 \to S_1)).
\end{array}$$

Let us show that specification of negative proof checker is not complete. Suppose that f^* is a total recursive function such that for every $(\cdot)^*$

$$\neg Proof(p^*, \ulcorner S^* \urcorner) \to Proof(f^*(p^*), \ulcorner \neg Proof(p^*, \ulcorner S^* \urcorner)\urcorner)$$

Put $p^* = 1$. Since for infinitely many formulas φ one has $\neg Proof(1, \ulcorner \varphi \urcorner)$, we conclude that $Th(f^*(1))$ is infinite, contradiction. Similar argument shows that specification of disjunction is incomplete too. However, we can turn these specifications into complete ones by adding new label arguments:

$$\begin{array}{l}
\neg[\![p]\!]S \wedge x \ni S \to [\![\mathsf{check}^-(p, x)]\!]\neg[\![p]\!]S \\
([\![p]\!]S_1 \vee [\![q]\!]S_2) \wedge x \ni S_1 \wedge y \ni S_2 \to [\![\mathsf{disj}(p, q, x, y)]\!](S_1 \vee S_2).
\end{array}$$

Lemma 2.19. Specification (OP) is complete iff for every $i = 1, \ldots, n$ either $\mathcal{LPS}_0 \vdash \neg(\bigwedge \delta_i)$ or $Var(F) \subseteq \{\overrightarrow{pq}, \overrightarrow{xy}\} \cup Var(\delta_i^+)$.

Proof. ($\Leftarrow$) Suppose that for every $i = 1, \ldots, n$ either $\mathcal{LPS}_0 \vdash \neg(\bigwedge \delta_i)$ or $Var(F) \subseteq \{\overrightarrow{pq}, \overrightarrow{xy}\} \cup Var(\delta_i^+)$. We take any normal multi-conclusion proof predicate *Prf* and a storage predicate *Str*. Let us describe the algorithm which calculates the desired function f^* for these predicates.

Suppose that the evaluation $\overrightarrow{pqxy}^*$ of variables $\overrightarrow{pqxy}$ is given.

Step 1. Construct the set T of arithmetical sentences which should be proven by $f^*(\overrightarrow{pq}^*, \overrightarrow{xy}^*)$ according to $Spec(f)$. Initially $T = \emptyset$. For all p^*_{ij} and x^*_{ij} find $Th(p^*_{ij})$ and $Cont(x^*_{ij})$ respectively. Take $i = 1, \dots, n$. For every i do the following.

1. Find all partial arithmetical evaluations $(\cdot)^\alpha$ with the domain $Var(\delta^+_i) \cup \overrightarrow{pqxy}$ such that $(\overrightarrow{pqxy})^\alpha = (\overrightarrow{pqxy})^*$ and $A^\alpha_{ij} \in Th(p^*_{ij})$ and $C^\alpha_{ij} \in Cont(x^*_{ij})$ for all j for the partial interpretation $\alpha = (Prf, Str, (\cdot)^\alpha)$. The number of appropriate $(\cdot)^\alpha$ is finite and they can be found effectively, since all the sets $Th(p^*_{ij})$ and $Cont(x^*_{ij})$ are finite.

2. Let $\delta^\pm_i$ be a maximal subset of δ^-_i such that $Var(\delta^\pm_i) \subseteq Var(\delta^+_i) \cup \overrightarrow{pqxy}$. For the every interpretation α from above check whether $(\bigwedge \neg\delta^\pm_i)^\alpha$ is true. If no, then $(\bigwedge \delta_i)^\alpha$ is clearly false, do nothing. If yes, then add F^α into T. Go to the next α or the next i.

Comment. If $(\bigwedge \neg\delta^\pm_i)^\alpha$ is true then the evaluation $(\cdot)^\alpha$ can be extended to an evaluation $(\cdot)^\beta$ of $Var(\delta^-_i)$ in such a way that $(\bigwedge \neg\delta^-_i)^\beta$ is true. Indeed, every number proves or stores a finite set of formulas, therefore for every formula of the form $[\![q]\!]B$ where B contains a variable $V \notin Dom(\alpha)$ there are infinitely many ways to extend α to V so that $[\![q]\!]B$ is false in the resulting interpretation. The same for formulas of the form $y \ni D$. Therefore, in this case $\mathcal{LPS}_0 \nvdash \neg(\bigwedge \delta_i)$ whence by the condition of the lemma $Var(F) \subseteq Var(\delta^+_i) \cup \overrightarrow{pqxy}$. Thus, $Var(F) \subseteq Dom(\alpha)$ and F^α coincides with F^β. Also note that $(\delta_i)^\beta$ is true, whence F^α is provable according to the definition of OP.

Step 2. After we looked through all i, the resulting set T is finite and consists of provable sentences. The algorithm returns $\mu n.T \subseteq Th(n)$. It terminates since Prf is a normal multi-conclusion proof predicate.

$(\Rightarrow)$ Prove by contradiction. Suppose that there exists i_0 such that $\mathcal{LPS}_0 \nvdash \neg(\bigwedge \delta_{i_0})$ and $Var(F) \nsubseteq \overrightarrow{pqxy} \cup Var(\delta^+_{i_0})$. By the uniform completeness theorem for $\mathcal{LPS}_0$ there exists an interpretation $* = (Prf_0, Str_0, (\cdot)^*$, where Prf_0 is a normal multi-conclusion proof predicate and $(\bigwedge \delta_{i_0})^*$ is true. Let V be a variable from $Var(F)$ such that $V \notin \overrightarrow{pqxy} \cup Var(\delta^+_{i_0})$. Consider all evaluations $(\cdot)^\alpha$ which coincide with $(\cdot)^*$ on all variables except V. Note that $(\cdot)^\alpha$ and $(\cdot)^*$ coincide on $\overrightarrow{pqxy} \cup Var(\delta^+_{i_0})$ whence $(\bigwedge \delta^+_{i_0})^\alpha$ is true. Since every number can prove or store only finitely many formulas, there are infinitely many evaluations α for which the formula $(\bigwedge \neg\delta^-_{i_0})^\alpha$ is true and, therefore,

$\delta^{\alpha}_{i_0}$ is true. According to the specification axiom, $f^*(\overrightarrow{pq}^*, \overrightarrow{xy}^*)$ should prove formulas F^α for all those α. So, $f^*(\overrightarrow{pq}^*, \overrightarrow{xy}^*)$ proves infinitely many formulas, contradiction. ■

In the rest of the paper we consider only complete specifications. Note that if $\mathcal{LPS}_0 \vdash \neg(\bigwedge \delta_i)$ then one can delete δ_i from the OP, the resulting formula is again a complete specification. In order to simplify the exposition in what follows we suppose that $\mathcal{LPS}_0 \nvdash \neg(\bigwedge \delta_i)$ for every i. The following definition describes classes of specifications we are going to deal with.

Definition 2.20. A specification (OP) describes a *positive operation* if for every i the set δ_i^- is empty. A specification is called *q-stable* if for every i all q-atomic subformulas of F either belong to δ_i^- or occur as subformulas in δ_i^+, that is $qSubFm(F) \subseteq (\delta_i^- \cup qSubFm(\delta_i^+))$. A specification is called *descriptive* if for every i from $(y \ni D) \in \lambda_i^-$ it follows that there is exists a renaming of label variables θ such that $(y \ni D)\theta \in SubFm(\lambda_i^+)$ and from $[\![q]\!]B \in \pi_i^-$ it follows that $Var(B) \cup qSubFm(B) \subseteq SubFm(\lambda_i^+)$.

Example 2.21. The following specifications are complete but not q-stable:

$$\begin{array}{l}
(x \ni [\![p]\!]S_1) \wedge [\![q]\!]S_2 \to \Box([\![p]\!]S_2 \to [\![q]\!]S_2)\\
[\![p]\!]S \to \Box([\![p]\!](S \to S) \to [\![p]\!]S)\\
(x \ni [\![p]\!]S) \to \Box([\![p]\!](S \wedge \neg S) \to [\![p]\!]S)\\
(x \ni [\![p]\!]S_1) \wedge (y \ni S_2) \to \Box([\![p]\!]S_2 \to S_2)
\end{array}$$

In the first two examples we can replace q-atoms in the conclusion which do not occur in premises by new propositional variables and add a label for this variable to the premise in order to make the new specification complete. As the result, we obtain a q-stable specification, and our original specification is a substitutional instance of the new one. However, we cannot do this in the last example.

The following specifications are q-stable, but not descriptive, the first two do not satisfy the conditions for variables occurring in π_i^-; for the last one the condition for λ_i^- does not hold:

$$\begin{array}{l}
[\![p]\!]S \wedge \neg[\![q]\!]S \to \Box\neg[\![q]\!]S\\
[\![p]\!](S \to S) \wedge \neg[\![q]\!]S \to \Box\neg[\![q]\!]S\\
\neg(y \ni (x \ni A)) \wedge (x \ni A) \to \Box\neg(y \ni (x \ni A)).
\end{array}$$

3 Logic of operations on proofs and labels

3.1 Logic $\mathcal{LPS}(\mathcal{F})$ and its semantics

Let $(\mathrm{OP})_i$ $(i = 1, \dots, n)$ be complete specifications of the types (a_i, b_i). Let $\mathcal{F} = f_1, \dots, f_n$ be the list of distinct functional symbols of the corresponding types $\mathit{proof}^{a_i} \times \mathit{label}^{b_i} \to \mathit{proof}$. By $\mathcal{L}(\mathcal{F})$ we denote the extension of the language $\mathcal{L}_0$ by these functional symbols. *Terms* of the language $\mathcal{L}(\mathcal{F})$ are generated from proof and label variables by functional symbols: all proof variables are terms and if f is a functional symbol of the type (a, b), $t_1, \dots, t_a$ are proof terms and $x_1, \dots, x_b$ are label variables then $f(t_1, \dots, t_a, x_1, \dots, x_b)$ is a proof term. *Formulas* are defined similarly with formulas of $\mathcal{LPS}_0$ with the only difference in the case of the proof operator: if t is a term and F is a formula then $[\![t]\!]F$ is a formula. We denote the sets of terms and formulas by $\mathit{Tm}(\mathcal{F})$ and $\mathit{Fm}(\mathcal{F})$ respectively.

A *substitution* for this language is a mapping which assigns proof terms to proof variables, label variables to label variables and formulas to sentence variables. We assume that substitutions have finite domains.

Definition 3.1. The logic of operations $f_1, \dots, f_n$ denoted by $\mathcal{LPS}(\mathcal{F})$ is the extension of the minimal logic $\mathcal{LPS}_0$ by formulas $\mathit{Spec}(f_i)$ as new schemas of axioms: if σ is a substitution, then $(\mathit{Spec}(f))\sigma$ is an axiom of $\mathcal{LPS}(\mathcal{F})$.

In [9] we considered the logic of positive q-stable specifications. We proved that in this case $\mathcal{LPS}(\mathcal{F})$ is arithmetically complete. Also, the logics of proof positive specifications was provided with the appropriate symbolic semantics and the corresponding completeness theorem was proved. In this paper we prove similar results for the logic of all descriptive q-stable specifications including negative ones.

Let us describe symbolic semantics for $\mathcal{LPS}(\mathcal{F})$. The definition below is an adjustment of symbolic semantics for $\mathcal{LPS}_0$ (see definition 2.2) to meet specifications from $\mathcal{F}$.

Definition 3.2. A *symbolic model* $\mathcal{M} = (\#, v)$ for $\mathcal{LPS}(\mathcal{F})$ consists of two objects: a binary relation $\#$ between $\mathit{LVar} \cup \mathit{Tm}(\mathcal{F})$ and formulas which is called *a proof assignment* and a truth evaluation of sentence variables v. The relation $\#$ should be *compatible with operations* $\mathcal{F}$, namely, for every operation $f \in \mathcal{F}$ and formula G the following holds:

$$\begin{gathered}\text{if there exists a substitution } \sigma \text{ such that } G = F\sigma \text{ and}\\ \exists i\ \forall j\ (\#(p_{ij}\sigma, A_{ij}\sigma),\ \#(x_{ij}\sigma, C_{ij}\sigma),\ \neg\#(q_{ij}\sigma, B_{ij}\sigma),\ \neg\#(y_{ij}\sigma, D_{ij}\sigma))\\ \text{then } \#(f(\overrightarrow{pq}\sigma, \overrightarrow{xy}\sigma), G).\end{gathered} \tag{4}$$

We define the truth relation $\mathcal{M} \models A$ by induction on A: for propositional variables $\mathcal{M} \models S$ iff $v(S) = \mathit{true}$, $\models$ commutes with boolean connectives and

$$\mathcal{M} \models x \ni A \rightleftharpoons \#(x, A), \qquad \mathcal{M} \models [\![t]\!]A \rightleftharpoons \#(t, A) \text{ and } \mathcal{M} \models A.$$

For every $e \in \mathit{LVar} \cup \mathit{Tm}(\mathcal{F})$ we denote $\#(e) = \{F \in \mathit{Fm}(\mathcal{F}) \mid \#(e, F)\}$. A model $\mathcal{M} = (\#, v)$ is called *finitely generated* if

- the set $\{x \in \mathit{PVar} \cup \mathit{LVar} \mid \#(x) \neq \emptyset\}$ is finite;
  for all $e \in \mathit{LVar} \cup \mathit{Tm}(\mathcal{F})$ the sets $\#(e)$ are finite;
- the reverse of (4) does not hold for a finite number of terms;
- v is a primitive recursive function.

Comment 3.3. Suppose that $f \in \mathcal{F}$, σ is a substitution and $\mathcal{A}_{ij}$, $\mathcal{B}_{ij}$, $\mathcal{C}_{ij}$ and $\mathcal{D}_{ij}$ are sets of $\mathcal{L}(\mathcal{F})$-formulas. By $f_\sigma(\mathcal{A}_{ij}, \mathcal{B}_{ij}, \mathcal{C}_{ij}, \mathcal{D}_{ij})$ we denote the set

$$\{F\sigma \mid \exists i\ \forall j\ (A_{ij}\sigma \in \mathcal{A}_{ij},\ B_{ij}\sigma \notin \mathcal{B}_{ij},\ C_{ij}\sigma \in \mathcal{C}_{ij},\ D_{ij}\sigma \notin \mathcal{D}_{ij})\}.$$

Condition (4) can be formulated as follows: for every term $f(\overrightarrow{ts}, \overrightarrow{zw})$

$$\bigcup_\sigma \{f_\sigma(\#(\vec{p}\sigma), \#(\vec{q}\sigma), \#(\vec{x}\sigma), \#(\vec{y}\sigma)) \mid (\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}\} \subseteq \#(t)$$

For a finitely generated model there exists a finite set of terms for which the inclusion above cannot be replaced by equality. Informally speaking, a model $\mathcal{M} = (\#, v)$ is finitely generated if the relation $\#$ is completely defined after it is defined for a finite set of terms, and these terms correspond to finite sets of formulas. Note that if $\mathcal{M}$ is finitely generated then the relation $\mathcal{M} \models B$ and the function $I_{\mathcal{M}}(t) \rightleftharpoons \{B \mid \mathcal{M} \models [\![t]\!]B\}$ are primitive recursive.

It is proves in [9] that if $\mathcal{F}$ consists of positive q-stable operations then the logic $\mathcal{LPS}(\mathcal{F})$ is sound and complete with respect to symbolic models described above. Namely, for every formula A one has $\mathcal{LPS}(\mathcal{F}) \vdash A$ iff $\mathcal{M} \models A$ for every finitely generated model $\mathcal{M}$. The semantical counterpart for the logic of descriptive q-stable operations is provided by so-called *reflexive models.*

Definition 3.4. A model $\mathcal{M} = (\#, v)$ is *reflexive* if for every term t and formula A from $\#(t, A)$ it follows that $\mathcal{M} \models A$.

Theorem 3.5. (*Soundness.*) Suppose that $\mathcal{F}$ consists of complete specifications. If $\mathcal{LPS}(\mathcal{F}) \vdash A$ then for every reflexive model $\mathcal{M}$ it is true that $\mathcal{M} \models A$.

Proof. Standard induction on the derivation of A in $\mathcal{LPS}(\mathcal{F})$. The only nontrivial thing here is to verify that all substitutional examples of the specification axioms are true. Consider a specification axiom $\bigvee_i(\bigwedge \delta_i) \to [\![f(\overrightarrow{pq}, \overrightarrow{xy})]\!]F$. Suppose that for a substitution σ and a number i one has that $\mathcal{M} \models (\bigwedge \delta_i)\sigma$. By the definition of the truth relation in a symbolic model the following conditions hold for all j: $\#(p_{ij}\sigma, A_{ij}\sigma)$, $\#(x_{ij}\sigma, C_{ij}\sigma)$, $\neg\#(y_{ij}\sigma, D_{ij}\sigma)$, and either $\neg\#(q_{ij}\sigma, B_{ij}\sigma)$ or $\mathcal{M} \not\models B_{ij}\sigma$. In the later case since $\mathcal{M}$ is reflexive we have $\neg\#(q_{ij}\sigma, B_{ij}\sigma)$.

Since $\#$ is compatible with f, we obtain $\#(f(\overrightarrow{pq}\sigma, \overrightarrow{xy}\sigma), F)$. Moreover, note that every $\mathcal{LPS}(\mathcal{F})$-model is an $\mathcal{LPS}_0$-model (if we treat proof terms as atomic objects like proof variables). By the definition of an operation on proofs we have that $\mathcal{LPS}_0 \vdash \bigwedge \delta_i \to F$, whence from $\mathcal{M} \models \bigwedge \delta_i\sigma$ it follows that $\mathcal{M} \models F\sigma$. Therefore, $\mathcal{M} \models [\![f(\overrightarrow{pq}\sigma, \overrightarrow{xy}\sigma)]\!]F\sigma$. ∎

Let us describe arithmetical semantics of $\mathcal{LPS}(\mathcal{F})$.

Definition 3.6. Consider the language $\mathcal{L}(\mathcal{F})$ where $\mathcal{F} = f_1, \ldots, f_n$ is a finite list of operations on proofs specified by the corresponding axioms $\mathit{Spec}(f_i)$. An *arithmetical interpretation* $* = (\mathit{Str}, \mathit{Prf}, \mathcal{F}^*, (\cdot)^*)$ of this language has the following parameters:

- a pair $(\mathit{Str}, \mathit{Prf})$ of a storage and a proof predicate compatible with all operations $f_1, \ldots, f_n$;
- $\mathcal{F}^* = \{f_1^*, \ldots, f_n^*\}$ where f_i^* are total recursive functions satisfying $\mathit{Spec}(f_i)$;
- an evaluation $(\cdot)^*$ that assigns natural numbers to label and proof variables and arithmetical sentences to propositional variables.

Arithmetical interpretation can be extended to all terms and formulas of the language $\mathcal{LPS}(\mathcal{F})$ in the usual way. For terms we put $f(\vec{t}, \vec{x})^* \rightleftharpoons f^*(\vec{t}^*, \vec{p}^*)$. For formulas we define F^* similarly with 2.6

Logic $\mathcal{LPS}(\mathcal{F})$ is sound and complete with respect to the described semantics. We formulate and prove soundness here and leave completeness until 3.3

Theorem 3.7. (*Arithmetical soundness.*) For every formula A if $\mathcal{LPS}(\mathcal{F}) \vdash A$ then for every interpretation $*$ of the language $\mathcal{L}(\mathcal{F})$ one has $\mathbf{PA} \vdash F^*$.

Proof. Induction on the derivation of A in $\mathcal{LPS}(\mathcal{F})$. ■

3.2 Completeness with respect to symbolic models

Theorem 3.8. *Completeness.* Suppose that $\mathcal{F}$ consists of descriptive q-stable specifications. If $\mathcal{LPS}(\mathcal{F}) \nvdash A$ then there exists a finitely generated reflexive model $\mathcal{M}$ such that $\mathcal{M} \not\models A$.

We suppose that $\mathcal{LPS}(\mathcal{F}) \nvdash A$ and construct a reflexive finitely generated model $\mathcal{M}$ such that $\mathcal{M} \not\models A$. Proof consists of three steps. On Step 1 we construct the set of formulas $Adeq(A)$ whose truth values can affect the truth value of our formula A. This is done by the *saturation algorithm.* On step 2 we define a so-called partial model for formulas from $Adeq(A)$ in which A does not hold. We find this model via reduction of $Adeq(A)$ to pure propositional language. On step 3 we extend the partial model from step 2 to the model which refutes A. We show that the resulting model is finitely generated and reflexive. Without loss of generality we assume that variables from $Var(A)$ do not occur in specification schemes.

Step 1: Saturation algorithm. Being given A it constructs a set of formulas $Adeq(A)$ which is called an *adequate set.* We need the following notation:

$$AdeqTm(A) := \{t\theta \mid t \in SubTm(A),\ \theta : LVar(A) \to LVar(A)\}.$$

Everywhere in this step σ ranges over substitutions with the domain $Dom(\sigma) \subseteq Var(\delta_i^+)$ and values for proof and label variables from $AdeqTm(A)$ and $LVar(A)$ respectively. The saturation algorithm works as follows:

1. Initialization. Calculate $AdeqTm(A)$ and put

$$Adeq_{-1}(A) := \{B\theta, \neg B\theta \mid B \in SubFm(A),\ \theta : LVar(A) \to LVar(A)\}$$

2. Construct the set $Adeq_0(A)$ as follows. Initially $Adeq_0(A) = Adeq_{-1}(A)$. Look through all specification axioms $Spec(f) \in \mathcal{F}$; every such axiom has the form $\bigvee_i(\bigwedge \delta_i) \to [\![f(\vec{p}, \vec{x})]\!]F$. For every i find all substitutions σ for which $\lambda_i^+\sigma \subseteq Adeq_{-1}(A)$. Extend $Adeq_0(A)$ by $SubFm(\delta_i^-\sigma)$.

3. Let $t_1, \ldots, t_N$ be the list of all terms from $AdeqTm(A)$ with the property: if $t_i \in SubTm(t_j)$ then $i < j$. For every $l = 1, \ldots, N$ calculate the set of formulas $Adeq_l(A)$ as follows.

   (a) Initially $Adeq_l(A) := Adeq_{l-1}(A)$.

   (b) Take the term t_l, if it is a proof variable then do nothing. If it has the form $f(\overrightarrow{ts}, \overrightarrow{zw})$ then take the specification axiom $Spec(f)$; it has the form $\bigvee_i(\bigwedge \delta_i) \to [\![f(\overrightarrow{pq}, \overrightarrow{xy})]\!]F$. For every i find all substitutions σ such that $(\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}$ and $\delta_i^+\sigma \subseteq Adeq_{l-1}(A)$. Extend $Adeq_l(A)$ by $SubFm([\![t_l]\!](F\sigma))$.

4. Put $Adeq(A) := Adeq_N(A)$.

Lemma 3.9. 1. $Adeq_0(A) \setminus Adeq_{-1}(A)$ consists of formulas of the form $[\![t]\!]E$ or formulas with the boolean connective as a principle symbol.

2. For every $l = 0, \ldots, N$ the following assertions are true

   (a) $Adeq_l(A)$ is closed under subformulas.

   (b) For every substitution σ if $\lambda_i^+\sigma \subseteq Adeq_l$ then $\delta_i^-\sigma \subseteq Adeq_l$.

   (c) If $G \in Adeq_{l+1}(A) \setminus Adeq_l(A)$ then G either has the form $[\![t_{l+1}]\!]E$ or G is a boolean combination of q-atomic formulas and sentence variables from $Adeq_l(A)$.

Proof. Let us prove item 1. Suppose that $G \in Adeq_0(A) \setminus Adeq_{-1}(A)$. Then there exists a specification axiom and a substitution σ such that $\delta_i^+\sigma \subseteq Adeq_{-1}(A)$ and $G \in SubFm(\delta_i^-\sigma)$, that is, either $G \in SubFm(\lambda_i^-\sigma)$ or $G \in SubFm(\pi_i^-\sigma)$. By the definition of a descriptive operation, in the first case there exists a formula $(x \ni C) \in \lambda_i^+$ and a renaming of label variables $\theta : LVar(A) \mapsto LVar(A)$ such that $G \in SubFm((x \ni C)\theta\sigma)$. Therefore, $G \in SubFm(\lambda_i^+\theta\sigma)$ whence $G \in Adeq_{-1}(A)$. In the second case $G \in SubFm(([\![q]\!]B)\sigma)$ where $[\![q]\!]B \in \pi_i^-$. There are three possibilities:

- $G = ([\![q]\!]B)\sigma = [\![q\sigma]\!](B\sigma)$; item 1 trivially holds.

- $G = E\sigma$, where $E \in \mathit{SubFm}(B\sigma)$. The case of a boolean E is trivial. If E is a sentence variable or q-atomic formula, then according to the definition of a q-stable descriptive operation we conclude that $E \in \mathit{SubFm}(\delta_i^+)$ whence $E\sigma \in \mathit{Adeq}_{-1}(A)$.

- $G \in \mathit{SubFm}(S_i\sigma)$, where $S_i \in \mathit{SVar}(B)$. By the definition of a descriptive operation $S_i \in \mathit{SVar}(\delta_i^+)$; therefore $\mathit{SubFm}(S_i\sigma) \subseteq \mathit{Adeq}_{-1}(A)$, thus $G \in \mathit{Adeq}_{-1}(A)$.

Now we will prove propositions of item 2 by joint induction on l.

Induction base. By definition, $\mathit{Adeq}_0(A)$ is closed under subformulas, so 2(a) is obvious. For 2(b) suppose that for some specification axiom, number i and substitution σ one has $\delta_i^+\sigma \subseteq \mathit{Adeq}_0(A)$. By item 1, $\lambda_i^+\sigma \subseteq \mathit{Adeq}_{-1}(A)$. Since $\mathit{Var}(\delta_i^-) \setminus \{\vec{q}, \vec{y}\} \subseteq \mathit{Var}(\lambda_i^+)$, this substitution σ was considered on step 2 of the saturation algorithm and $\mathit{SubFm}(\delta_i^-\sigma)$ was added to $\mathit{Adeq}_0(A)$.

For item 2(c) suppose that $G \in \mathit{Adeq}_1(A) \setminus \mathit{Adeq}_0(A)$. Then there exist a specification axiom $\mathit{Spec}(f)$, a number i and a substitution σ such that $\delta_i^+\sigma \subseteq \mathit{Adeq}_0(A)$, $f(\overrightarrow{pq}\sigma, \overrightarrow{xy}\sigma) = t_1$ and $G \in \mathit{SubFm}([\![t_1]\!](F\sigma))$. By item 2(b) we obtain $\delta_i^-\sigma \subseteq \mathit{Adeq}_0(A)$. There are three possible cases for G.

- $G = [\![t_1]\!](F\sigma)$; this case is trivial.

- $G = E\sigma$, where $E \in \mathit{SubFm}(F\sigma)$. The case of a boolean E is trivial. If E is a sentence variable or a q-atomic formula, then according to the definition of a q-stable descriptive operation we conclude that $E \in \mathit{SubFm}(\delta_i^+) \cup \delta_i^-$ whence $E\sigma \in \mathit{Adeq}_0(A)$.

- $G \in \mathit{SubFm}(S_i\sigma)$ where $S_i \in \mathit{SVar}(F)$. Since operation f is complete, by lemma 2.19 $S_i \in \mathit{SVar}(\delta_i^+)$. Therefore $\mathit{SubFm}(S_i\sigma) \subseteq \mathit{Adeq}_0(A)$.

Induction step. For 2(a) note that $\mathit{Adeq}_{l+1}(A)$ is closed under subformulas, since both $\mathit{Adeq}_l(A)$ and the set of additional formulas are, the first one by the induction hypothesis and the second one by the construction.

For 2(b) suppose that for some i and σ one has $\lambda_i^+\sigma \subseteq \mathit{Adeq}_{l+1}(A)$. Then by the induction hypothesis for 2(c) this yields $\lambda_i^+\sigma \subseteq \mathit{Adeq}_l(A)$ whence by the induction hypothesis for 2(b) $\lambda_i^-\sigma \subseteq \mathit{Adeq}_l(A)$.

Induction step for 2(c) is similar to the induction base. End of proof. ■

Lemma 3.10. Suppose that $t \in \mathit{AdeqTm}(A)$, $t = f(\overrightarrow{ts}, \overrightarrow{zw})$, and σ is a substitution such that $(\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}$. Suppose that there exists i such that $(\delta_i)\sigma \subseteq \mathit{Adeq}(A)$. Then $[\![t]\!](F\sigma) \in \mathit{Adeq}(A)$.

Proof. Since $f(\overrightarrow{ts}, \overrightarrow{zw}) \in AdeqTm(A)$ we have that $f(\overrightarrow{ts}, \overrightarrow{zw})$ coincides with some of the terms t_l. For all j terms t_{ij} are subterms of t_l. Thus, they occur in the list of terms of $AdeqTm(A)$ with the numbers smaller than l and they are processed by the saturation algorithm earlier. Then, by item 2(c) of lemma 3.9, all formulas $[\![t_{ij}]\!](A_{ij}\sigma)$ belong to $Adeq_{l-1}(A)$. From the same lemma we obtain that all formulas $(z_{ij} \ni C_{ij}\sigma)$ belong to $Adeq_0(A)$, therefore, $\delta_i^+\sigma \subseteq Adeq_{l-1}(A)$. From item 2(b) of lemma 3.9 we conclude that $\delta_i^-\sigma \subseteq Adeq_{l-1}(A)$; therefore after step l of the saturation algorithm is performed $[\![f(\overrightarrow{ts}, \overrightarrow{zw})]\!](F\sigma) \in Adeq_l(A)$. ■

Step 2. Partial model.

Definition 3.11. Suppose that $LV \subseteq LVar$, $SV \subseteq SVar$ and T is a set of terms such that $LVar(T) \subseteq LV$ and T is close under subterms and renaming of label variables in LV. By $Fm(SV, LV, T)$ we denote the set of all formulas F for which $LVar(F) \subseteq LV$, $SVar(F) \subseteq SV$ and $SubTm(F) \subseteq T$.

A *partial model* $\mathcal{M}_p = (\#_p, v_p)$ for $Fm(SV, LV, T)$ consists of two objects: $\#_p$ is a relation between $T \cup LV$ and $Fm(SV, LV, T)$; v_p is a truth evaluation of propositional variables from SV. The relation $\#_p$ should satisfy condition (4) restricted to terms from T, namely, for every operation $f \in \mathcal{F}$, substitution σ and number i if $f(\overrightarrow{pq}\sigma, \overrightarrow{xy}\sigma) \in T$ and $\#_p(p_{ij}\sigma, A_{ij}\sigma)$, $\#_p(x_{ij}\sigma, C_{ij}\sigma)$, $\neg\#_p(q_{ij}\sigma, B_{ij}\sigma)$, $\neg\#_p(y_{ij}\sigma, D_{ij}\sigma)$ for all j then $\#(f(\overrightarrow{pq}\sigma, \overrightarrow{xy}\sigma), F\sigma)$.

The relation "$\mathcal{M}_p \models F$" is defined for $F \in Fm(SV, LV, T)$ similarly with ordinary models. The definition of a *finitely generated partial model* and a *reflexive partial model* is the restriction of similar definitions for ordinary models to the sets SV, LV, T and $Fm(SV, LV, T)$.

For any formula A we denote $Fm(SVar(A), LVar(A), AdeqTm(A))$ by Fm_A.

Lemma 3.12. If $\mathcal{LPS}(\mathcal{F}) \not\vdash A$ then there exists a finitely generated reflexive partial model $\mathcal{M}_p$ for Fm_A such that $\mathcal{M}_p \not\models A$.

Proof. Suppose that $\mathcal{LPS}(\mathcal{F}) \not\vdash A$. Run the saturation algorithm on the input A; the result is the set $Adeq(A) \subseteq Fm_A$. For every term $t \in AdeqTm(A)$ let us reserve a fresh proof variable p_t. Suppose that $F \in Fm_A$. By F^0 we denote the result of replacing all uppermost occurrences of terms in F by the corresponding proof variables, namely,

$$([\![t]\!]F)^0 \rightleftharpoons [\![p_t]\!]F^0.$$

Let $AX(A)$ stand for the set of specification axioms whose all q-atomic subformulas belong to $Adeq(A)$ (here we replace an axiom $\bigvee_i(\bigwedge \delta_i) \to [\![f(\overrightarrow{pq}, \overrightarrow{xy})]\!]F$ by a finite set of axioms $(\bigwedge \delta_i) \to [\![f(\overrightarrow{pq}, \overrightarrow{xy})]\!]F$ for all i. For every $F \in Fm_A$ define

$$\widetilde{F} \rightleftharpoons \Big((\bigwedge AX(A)) \to F\Big)^0.$$

Since $\mathcal{LPS}(\mathcal{F}) \not\vdash A$ we conclude that $\mathcal{LPS}_0 \not\vdash \widetilde{A}$ (otherwise the reverse substitution of terms t for the corresponding proof variables p_t transforms the derivation of $\widetilde{A}$ in $\mathcal{LPS}_0$ into the derivation of A in $\mathcal{LPS}(\mathcal{F})$. According to theorem 2.9 there exists a model $\mathcal{M}_0 = (\#_0, v_0)$ for $\mathcal{LPS}_0$ such that $\mathcal{M}_0 \not\models \widetilde{A}$.

Now we defined the desired partial model $\mathcal{M}_p$ as follows: for every $x \in LVar(A)$, $S \in SVar(A)$ and $t \in AdeqTm(A)$ put

$$\#_p(x, B) \text{ iff } \#_0(x, B^0), \quad \#_p(t, B) \text{ iff } \mathcal{M}_0 \models [\![p_t]\!]B^0, \quad v_p(S) \rightleftharpoons v_0(S).$$

By induction on the formula $F \in Fm_A$ one can show that $\mathcal{M}_p \models F \iff \mathcal{M}_0 \models F^0$. Since $\mathcal{M}_0 \not\models \widetilde{A}$ we conclude that $\mathcal{M}_p \not\models A$. Let us prove now, that $\mathcal{M}_0$ is a reflexive finitely generated partial model.

Suppose that $t \in AdeqTm(A)$ where $t = f(\overrightarrow{ts}, \overrightarrow{zw})$ and σ is a substitution such that $(\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}$. Suppose that there exists i such that for all j one has

$$\#_p(t_{ij}, A_{ij}\sigma), \quad \neg\#_p(s_{ij}, B_{ij}\sigma), \quad \#_p(z_{ij}, C_{ij}\sigma), \quad \neg\#_p(w_{ij}, D_{ij}\sigma).$$

By definition of $\#_p$ it follows that

$$\mathcal{M}_0 \models [\![p_{t_{ij}}]\!](A_{ij}\sigma)^0,\ (z_{ij} \ni C_{ij}\sigma)^0, \quad \mathcal{M}_0 \not\models [\![p_{s_{ij}}]\!](B_{ij}\sigma)^0,\ (w_{ij} \ni D_{ij}\sigma)^0.$$

From the definition of $\mathcal{M}_0$ it immediately follows that $[\![t_{ij}]\!](A_{ij}\sigma) \in Adeq(A)$ and $(z_{ij} \ni C_{ij}\sigma) \in Adeq(A)$ for all j. Then by lemmas 3.10 and 3.9 we conclude that $[\![f(\overrightarrow{ts}, \overrightarrow{zw})]\!](F\sigma) \in Adeq(A)$ and $\delta_i^- \sigma \subseteq Adeq(A)$. Therefore, $(\bigwedge \delta_i \to [\![t]\!]F)\sigma$ belongs to $AX(A)$ whence the formula $((\bigwedge \delta_i\sigma) \to [\![t]\!](F\sigma))^0$ is true in $\mathcal{M}_0$. As we mentioned above, $\mathcal{M}_0 \models (\wedge\delta_i\sigma)^0$. Therefore, $\mathcal{M}_0 \models [\![p_t]\!](F\sigma)^0$ whence by the definition of $\#_p$ we conclude that $\#_p(t, F\sigma)$. So, $\mathcal{M}_p$ is a partial model.

It is clear that $\mathcal{M}_p$ is finitely generated since the set $AdeqTm(A)$ is finite and for every term $t \in AdeqTm(A)$ the set $\#_p(t)$ is finite. Let us prove that $\mathcal{M}_p$ is reflexive. Suppose that $\#_p(t, F)$. By the definition of $\#_p$ this yields $\mathcal{M}_0 \models [\![p_t]\!]F^0$ whence by the definition of a model for $\mathcal{LPS}_0$ one gets $\mathcal{M}_0 \models F^0$. Therefore, $\mathcal{M}_p \models F$. ■

Step 3. Completion.

Lemma 3.13. (*Completion.*) Let $\mathcal{M}_p = (\#_p, v_p)$ be a partial model for $Fm(SV, LV, T)$. Then there exists a model $\mathcal{M} = (\#, v)$ such that for every formula $F \in Fm(SV, LV, T)$ one has $\mathcal{M} \models F$ iff $\mathcal{M}_p \models F$. Moreover, if $\mathcal{M}_p$ is finitely generated, then $\mathcal{M}$ is, and if $\mathcal{M}_p$ is reflexive then $\mathcal{M}$ is.

Proof. We define # as follows: for all $x \in LVar \cup PVar$ and for all $t \in T$

$$\#(x, F) \text{ iff } \#_p(x, F), \quad \#(t, F) \text{ iff } \#_p(t, F).$$

For the remaining terms define # by induction on the complexity of terms:

$$\#(f(\overrightarrow{ts}, \overrightarrow{zw})) = \bigcup_\sigma \{f_\sigma(\#(\overrightarrow{pq}\sigma), \#(\overrightarrow{xy}\sigma)) \mid (\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}\}$$

Then let us define v: for every sentence variable S put

$$v(S) = \mathit{true} \text{ iff } v_p(S) = \mathit{true}.$$

By induction on the formula $F \in Fm(SV, LV, T)$ one can easily check that $\mathcal{M} \models F$ iff $\mathcal{M}_p \models F$.

Let us show that the resulting $\mathcal{M}$ is a model. Namely, we take any term $f(\overrightarrow{ts}, \overrightarrow{zw})$ and a substitution σ such that $(\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}$. We have to check that as soon as for all j one has $\#(t_{ij}, A_{ij}\sigma)$, $\neg\#(s_{ij}, B_{ij}\sigma)$, $\#(z_{ij}, C_{ij}\sigma)$ and $\neg\#(w_{ij}, D_{ij}\sigma)$ then $\#(f(\overrightarrow{ts}, \overrightarrow{zw}), F\sigma)$. The case when $f(\overrightarrow{ts}, \overrightarrow{zw}) \notin T$ is trivial by the definition of #. Suppose that $f(\overrightarrow{ts}, \overrightarrow{zw}) \in T$. Then for all j we have $A_{ij}\sigma, C_{ij}\sigma \subseteq Fm(SV, LV, T)$. Since T is closed under subterms and $LVar(T) \subseteq LV$, we get $\overrightarrow{ts} \subseteq T$ and $\overrightarrow{zw} \subseteq LV$. The operation f is descriptive, therefore $Var(\{B_{ij}, D_{ij} | j\}) \subseteq Var(\lambda_i^+)$ whence $B_{ij}\sigma, D_{ij}\sigma \in Fm(SV, LV, T)$ for all j. Since $\mathcal{M}_p$ is a partial model for $Fm(SV, LV, T)$ we conclude that $\#_p(f(\overrightarrow{ts}, \overrightarrow{zw}), F\sigma)$ whence $\#(f(\overrightarrow{ts}, \overrightarrow{zw}), F\sigma)$.

Let us show that $\mathcal{M}$ is finitely generated as soon as $\mathcal{M}_p$ is. The only thing which we have to verify here is the following: for every term t the set $\#(t)$ is finite. Induction on t. The case when $t \in T$ or $t \in PVar$ is trivial. Suppose that $t = f(\overrightarrow{ts}, \overrightarrow{zw})$ and $t \notin T$. Then according to the definition of # we have that

$$\#(t) = \bigcup_\sigma \{f_\sigma(\#(\overrightarrow{ts}), \#(\overrightarrow{zw}) \mid (\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}\}$$

that is, $G \in \#(t)$ iff there exists a substitution σ such that $\sigma(\overrightarrow{pqxy}) = \overrightarrow{tszw}$, $G = F\sigma$ and for all j one has $\#(t_{ij}, A_{ij}\sigma)$, $\neg\#(s_{ij}, B_{ij}\sigma)$, $\#(z_{ij}, C_{ij})$ and $\neg\#(w_{ij}, D_{ij}\sigma)$. All the sets $\#(t_{ij})$ are finite by the induction hypothesis and all the sets $\#(z_{ij})$ are finite due to the fact that $\mathcal{M}_p$ is finitely generated. Therefore, there are finitely many substitutions σ satisfying the conditions above. Since $Var(F) \in Var(\delta_i^+) \cup \overrightarrow{pqxy}$, we conclude that $\#(t)$ is finite.

Finally, it remains to prove that $\mathcal{M}$ is reflexive as soon as $\mathcal{M}_p$ is. Suppose that $\#(t, G)$, we should prove that $\mathcal{M} \models G$. Induction on the construction of t. The cases when t is a proof variable or $t \in T$ follow directly from the definition of $\#$ and the fact that $\mathcal{M}_p$ is reflexive. Let $t = f(\overrightarrow{ts}, \overrightarrow{zw})$ and $t \notin T$. Then by the definition of $\#$ we have that $\#(t, G)$ iff there exists a number i and a substitution σ such that $(\overrightarrow{pqxy})\sigma = \overrightarrow{tszw}$, $G = F\sigma$ and for all j one has $\#(t_{ij}, A_{ij}\sigma)$, $\neg\#(s_{ij}, B_{ij}\sigma)$, $\#(z_{ij}, C_{ij})$ and $\neg\#(w_{ij}, D_{ij}\sigma)$. By the induction hypothesis $\mathcal{M} \models A_{ij}\sigma$, therefore $\mathcal{M} \models [\![t_{ij}]\!]A_{ij}\sigma$ and $\mathcal{M} \models \bigwedge \delta_i\sigma$. By the definition of an operation we have $\mathcal{LPS}_0 \vdash \bigwedge \delta_i \to F$. Since $\mathcal{M}$ is an $\mathcal{LPS}_0$-model we conclude that $\mathcal{M} \models F\sigma$, that is, $\mathcal{M} \models G$. ■

3.3 Arithmetical completeness of $\mathcal{LPS}(\mathcal{F})$

Theorem 3.14. (*Arithmetical completeness.*) Suppose that $\mathcal{F}$ consists of descriptive q-stable operations. For every formula A if $\mathcal{LPS}(\mathcal{F}) \nvdash A$ then there exists an interpretation $*$ of the language $\mathcal{L}(\mathcal{F})$ such that A^* is false in the standard model.

Proof. The proof of soundness goes by standard induction on the derivation of A in $\mathcal{LPS}(\mathcal{F})$.

To prove the completeness, suppose that $\mathcal{LPS}(\mathcal{F}) \nvdash A$. We have to show that there exists an interpretation $*$ such that A^* is false. First we apply theorem 3.8 and find a finitely generated reflexive model $\mathcal{M} = (\#, v)$ such that $\mathcal{M} \not\models A$. We are going to define an embedding of this model into Peano Arithmetic. We fix an injective Gödel numbering of the joint syntax of $\mathcal{LPS}(\mathcal{F})$ and Peano Arithmetic. We first consider an auxiliary evaluation $(\cdot)^@$: for every sentence variable S_i, label variable l and proof term t put

$$t^@ \rightleftharpoons \ulcorner t \urcorner, \qquad l^@ \rightleftharpoons \ulcorner l \urcorner, \qquad S_i^@ \rightleftharpoons \begin{cases} i = i & \text{if } \mathcal{M} \models S_i \\ i = 0 & \text{otherwise} \end{cases}$$

We find the appropriate storage and proof predicates by the multiple fixed point equation. Since the model $\mathcal{M}$ is finitely generated, the relation "$\mathcal{M} \models$

$[\![t]\!]F$" is primitive recursive, thus, it can be represented by a Δ_1 arithmetical formula. The function, which calculates $\ulcorner B^{@}\urcorner$ being given $\ulcorner \mathit{Prf}\urcorner$, $\ulcorner \mathit{Str}\urcorner$ and $\ulcorner B\urcorner$ (B an $\mathcal{LPS}(\mathcal{F})$–formula), is primitive recursive too. Therefore, by the fixed point lemma for **PA**, there exist arithmetical Δ_1 formulas *Prf*, *Str*, such that the following formulas are provable in **PA**

$$\begin{array}{lcl} \mathit{Prf}(x,y) & \leftrightarrow & \mathit{Proof}(x,y)\vee \\ & & \text{“}x=\ulcorner t\urcorner \text{ and } y=\ulcorner B^{@}\urcorner \text{ and } \mathcal{M}\models [\![t]\!]B\text{”} \\ \mathit{Str}(x,y) & \leftrightarrow & \mathit{Store}(x,y)\vee \\ & & \text{“}x=\ulcorner l\urcorner \text{ and } y=\ulcorner B^{@}\urcorner \text{ and } \mathcal{M}\models (l\ni B)\text{”} \end{array} \qquad (FPE)$$

It is clear that the mapping $(\cdot)^{@}$ is injective, namely, for different formulas B and C we have $B^{@}\neq C^{@}$.

Lemma 3.15. Let α be an arithmetical interpretation of the language $\mathcal{LPS}_0$ based on predicates *Prf* and *Str* defined by (FPE). Suppose that $A_0,\dots,A_n$ are formulas in the language $\mathcal{L}_0$ and $\widetilde{A}_0,\dots,\widetilde{A}_n$ are formulas in the language $\mathcal{L}(\mathcal{F})$. Suppose that

$$A_j^{\alpha}=\widetilde{A}_j^{@} \text{ for all } j. \tag{5}$$

Then there exists a substitution σ such that for every $\mathcal{L}_0$–formula B

$$\text{if } \ \mathit{Var}(B)\subseteq\bigcup_j \mathit{Var}(A_j), \ \text{ then } \ B^{\alpha}=(B\sigma)^{@}. \tag{6}$$

Proof. For the sake of simplicity, we also allow in (5) equalities of the form $p^{\alpha}=t^{@}$ and $x^{\alpha}=y^{@}$ where p is a proof variable, t is a term and x, y are label variables. We call formulas of $\mathcal{L}_0$ and proof and label variables *expressions* of the language $\mathcal{L}_0$; formulas of $\mathcal{L}(\mathcal{F})$, proof terms and label variables are expressions of $\mathcal{L}(\mathcal{F})$. So, the most general form of (5) is $E_j^{\alpha}=\widetilde{E_j}^{@}$ ($j=1,\dots,n$) where E_j are expressions in the language $\mathcal{L}_0$ and $\widetilde{E_j}$ are expressions on the language $\mathcal{L}(\mathcal{F})$ of the same type.

We define the complexity of an expression E in the language $\mathcal{L}_0$ as the total number of connectives in E (namely, $|S|=0$, $|p|=|x|=0$, $|A\vee B|=|A\wedge B|=|A\rightarrow B|=|A|+|B|+1$, $|\neg A|=|[\![p]\!]A|=|x\ni A|=|A|+1$). We prove the existence of σ satisfying (6) by induction on the parameter $\sum_j |E_j|$.

If $\sum_j |E_j|=0$ then all E_j are variables. If some of these variables coincide, say $A_l=A_k$, we consequently obtain that $E_l^{\alpha}=E_k^{\alpha}$, whence $\widetilde{E}_l^{@}=\widetilde{E}_k^{@}$, and finally $\widetilde{E}_l = \widetilde{E}_k$ since the arithmetical evaluation @ is injective, namely, it maps different formulas of $\mathcal{LPS}(\mathcal{F})$ into different arithmetical formulas, and different proof and label variables into different numbers. Therefore, one can define a desired substitution σ by $\sigma(E_j) \rightleftharpoons \tilde{E}_j$.

If $\sum_j |E_j| > 0$ then there exists a formula, say A_j, which is not a sentence variable. In this case the main connective of A_j coincides with the main connective of A_j^α, which in turn coincides with the main connective of $\widetilde{A}_j^@$. Note that the interpretation @ maps sentence variables to atomic arithmetical formulas, therefore, the main connective of $\widetilde{A}_j^@$ coincides with the main connective of $\widetilde{A}_j$. Therefore, we can replace the condition $\widetilde{A}_j^@ = A_j^\alpha$ in (5) by the corresponding equalities for the components of A_j. Doing so, we decrease the induction parameter. For example, consider the most interesting case is $A_j = [\![p]\!]B_j$, $\widetilde{A}_j = [\![t]\!]\widetilde{B}_j$. In this case the corresponding equation in (5) has the form $Prf(p^\alpha, \ulcorner B_j^\alpha \urcorner) = Prf(\ulcorner t \urcorner, \ulcorner \widetilde{B}_j^@ \urcorner)$. We can replace this equality by two new ones: $p^\alpha = t^@$ and $B_j^\alpha = \widetilde{B}_j^@$. ■

Lemma 3.16. For every formula C

$$\begin{array}{lcl} \mathcal{M} \models C & \Rightarrow & \mathbf{PA} \vdash C^@; \\ \mathcal{M} \not\models C & \Rightarrow & \mathbf{PA} \vdash \neg C^@. \end{array} \tag{7}$$

Proof. Induction on the formula C. The induction base when C is a propositional variable is a direct corollary of the definition of @. The case of boolean connectives immediately follows from the induction hypothesis.

Suppose that C has the form $[\![t]\!]B$. If $\mathcal{M} \models C$ then by the definition of *Prf* we conclude that $Prf(\ulcorner t \urcorner, \ulcorner B^@ \urcorner)$ is true, whence $\mathbf{PA} \vdash Prf(\ulcorner t \urcorner, \ulcorner B^@ \urcorner)$ that is $\mathbf{PA} \vdash C^@$. If $\mathcal{M} \not\models C$ then $Prf(\ulcorner t \urcorner, \ulcorner B^@ \urcorner)$ is false because of the injectivity of the evaluation $(\cdot)^@$. Therefore $\mathbf{PA} \vdash \neg Prf(\ulcorner t \urcorner, \ulcorner B^@ \urcorner)$, that is, $\mathbf{PA} \vdash \neg C^@$. The remaining case when C has the form $x \ni B$ can be treated similarly. ■

Lemma 3.17. *Prf* is a normal proof predicate, *Str* is a storage predicate.

Proof. From properties of finitely generated models and definition of *Prf*, *Str* it follows that for every proof term t and label variable l the sets $Th(\ulcorner t \urcorner)$ and $Cont(\ulcorner l \urcorner)$ are finite and the functions $x \mapsto Th(x)$ and $x \mapsto Cont(x)$ are recursive for *Prf* nd *Str*.

It only remains to verify that if $\mathit{Prf}(n, \ulcorner\varphi\urcorner)$ is true then $\mathbf{PA} \vdash \varphi$. Suppose that $\mathit{Prf}(n, \ulcorner\varphi\urcorner)$. By the definition of Prf there are two possibilities. The first case when $\mathit{Proof}(n, \ulcorner\varphi\urcorner)$ is trivial. For the second case suppose that $n = \ulcorner t\urcorner$, $\varphi = B^{@}$ and $\mathcal{M} \models [\![t]\!]B$. In this case $\mathcal{M} \models B$ whence $\mathbf{PA} \vdash B^{@}$ by lemma 3.16. ■

Now we can define interpretation of operations on proofs. Consider an operation f described by the specification axiom $\bigvee_i \delta_i \to [\![f(\vec{p,q}, \vec{x,y})]\!]F$. Let $f^{@}$ be a total recursive function which satisfies this specification axiom for the standard predicates *Proof*, *Store*. The following function which converts *Prf*-proofs and *Str*-labels into proofs and labels according to *Proof* and *Store* is total recursive:

$$cv(x) \rightleftharpoons \begin{cases} \mu y.\forall z(\mathit{Prf}(x,z) \leftrightarrow \mathit{Proof}(y,z)) & \text{if } x = \ulcorner t\urcorner \text{ for } t \in \mathit{Tm}(\mathcal{F}) \\ \mu y.\forall z(\mathit{Str}(x,z) \leftrightarrow \mathit{Store}(y,z)) & \text{if } x = \ulcorner l\urcorner \text{ for } l \in \mathit{LVar} \\ x & \text{otherwise} \end{cases}$$

For every functional symbol $f \in \mathcal{F}$ let $f^{@}$ be a total recursive function satisfying $\mathit{Spec}(f)$ for *Proof* and *Str*. We define total recursive function f^* as follows:

$$f^*(\overrightarrow{n}, \overrightarrow{m}) \rightleftharpoons \begin{cases} \ulcorner f(\overrightarrow{n}, \overrightarrow{m})\urcorner & \text{if for all } i,j\ n_{ij} = \ulcorner t_{ij}\urcorner,\ m_{ij} = \ulcorner x_{ij}\urcorner, \\ & \text{where } t_{ij} \in \mathit{Tm}(\mathcal{F}),\ x_{ij} \in \mathit{LVar} \\ f^{@}(\overrightarrow{cv(n_{ij})}, \overrightarrow{cv(m_{ij})}) & \text{otherwise} \end{cases}$$

Lemma 3.18. Functions f^* defined above satisfy the corresponding specification axioms $\mathit{Spec}(f)$.

Proof. Consider a specification axiom $\bigvee_i(\wedge\delta_i) \to [\![f(\vec{p}, \vec{x})]\!]F$. Let $\alpha = (\mathit{Prf}, \mathit{Str}, (\cdot)^\alpha)$ be an arithmetical interpretation. Suppose that δ_i^α is true for some i. Then there are two possibilities: 1) for every i, j it is true that $p_{ij}^\alpha = \ulcorner t_{ij}\urcorner$, $q_{ij}^\alpha = \ulcorner s_{ij}\urcorner$, $x_{ij}^\alpha = \ulcorner z_{ij}\urcorner$, $y_{ij}^\alpha = \ulcorner w_{ij}\urcorner$ for some terms t_{ij}, s_{ij} and label variables z_{ij}, w_{ij} or 2) it is not true for some i, j. The second case is trivial.

Let us consider the first possibility. In this case there exists i such that for all j the formulas $\mathit{Prf}(\ulcorner t_{ij}\urcorner, \ulcorner A_{ij}^\alpha\urcorner)$ and $\mathit{Str}(\ulcorner z_{ij}\urcorner, \ulcorner C_{ij}^\alpha\urcorner)$ are true and $\mathit{Prf}(\ulcorner s_{ij}\urcorner, \ulcorner B_{ij}^\alpha\urcorner)$ and $\mathit{Str}(\ulcorner w_{ij}\urcorner, \ulcorner D_{ij}^\alpha\urcorner)$ are false. We fix i as above. By (FPE) we can conclude that for all j there exist formulas $\widetilde{A}_{ij}$, $\widetilde{C}_{ij}$ such that

$$\mathcal{M} \models [\![t_{ij}]\!]\widetilde{A}_{ij}, \quad \mathcal{M} \models x_{ij} \ni \widetilde{C}_{ij} \quad \text{and} \tag{8}$$

$$A_{ij}^\alpha = \widetilde{A}_{ij}^{@}, \quad C_{ij}^\alpha = \widetilde{C}_{ij}^{@}. \tag{9}$$

According to lemma 3.15 there exists a substitution σ such that for every formula B satisfying the condition $Var(B) \subseteq \bigcup_i Var(\delta_i^+)$ one has $B^\alpha = (B\sigma)^@$. In particularly, in view of (9) we have $\widetilde{A}_{ij}^@ = A_{ij}^\alpha = (A_{ij}\sigma)^@$, and the same for C_{ij}. Since interpretation @ is injective, this yields for all j

$$A_{ij}\sigma = \widetilde{A}_{ij},\ C_{ij}\sigma = \widetilde{C}_{ij}. \tag{10}$$

Combining (10) and (8) we obtain that $\mathcal{M} \models [\![t_{ij}]\!](A_{ij}\sigma)$, $\mathcal{M} \models (x_{ij} \ni (C_{ij}\sigma))$.

Let us show that for all j we have $\mathcal{M} \not\models [\![s_{ij}]\!](B_{ij}\sigma)$ and $\mathcal{M} \not\models (y_{ij} \ni D_{ij}\sigma)$. Suppose for the converse that for some j one has $\mathcal{M} \models [\![s_{ij}]\!](B_{ij}\sigma)$. By the definition of a q-stable descriptive operation we have $Var(B) \subseteq Var(\delta_i^+)$ whence by lemma 3.15 we obtain $B_{ij}^\alpha = (B_{ij}\sigma)^@$ From $\mathcal{M} \models [\![s_{ij}]\!](B_{ij}\sigma)$ by (FPE) it follows that $Prf(\ulcorner s_{ij} \urcorner, \ulcorner (B_{ij}\sigma)^@ \urcorner)$, that is, $Prf(\ulcorner s_{ij} \urcorner, \ulcorner B_{ij}^\alpha \urcorner)$, contradiction. The case of the storage operator can be treated similarly.

Now we have $\mathcal{M} \models \bigwedge(\delta_i\sigma)$. By the correctness of $\mathcal{LPS}(\mathcal{F})$ with respect to the symbolic models we conclude that $\mathcal{M} \models [\![f(\vec{t}, \vec{x})]\!](F\sigma)$ whence $Prf(\ulcorner f(\vec{t}, \vec{x}) \urcorner, \ulcorner (F\sigma)^@ \urcorner)$ by the definition of Prf. Since $Var(F) \subseteq Var(\delta_i^+) \cup \overrightarrow{pqxy}$, we have that $(F\sigma)^@ = F^\alpha$. From the definition of f^* we derive $f^*(\vec{t}^\alpha, \vec{x}^\alpha) = \ulcorner f(\vec{t}, \vec{x}) \urcorner$ whence $Prf(f^*(\vec{t}^\alpha, \vec{x}^\alpha), \ulcorner F^\alpha \urcorner)$. ■

Now we define the interpretation $* = (Prf, Str, \mathcal{F}^*, (\cdot)^*)$ where for every $V \in Var$

$$V^* \rightleftharpoons V^@.$$

By induction on the term t we can show that $t^* = t^@$. Therefore, for every formula F we have $F^* = F^@$. So from lemma 3.16 we deduce that for every formula C

$$\begin{array}{lcl} \mathcal{M} \models C & \Rightarrow & \mathbf{PA} \vdash C^* \\ \mathcal{M} \not\models C & \Rightarrow & \mathbf{PA} \vdash \neg C^*. \end{array}$$

Since $\mathcal{M} \not\models A$, we conclude that $\mathbf{PA} \vdash \neg A^*$. ■

4 Acknowledgement

I am thankful to Sergei Artemov who introduced me to the subject and provided advise and encouragement. I am also grateful to Vladimir Krupski for numerous discussions on the subject and his helpful remarks.

References

[1] S. Artemov, *Uniform provability realization of intuitionistic logic, modality and λ-terms*, Electronic Notes in Theoretical Computer Science 23 (1999).

[2] S. Artemov, *Explicit provability and constructive semantics*, Bulletine of Symbolic Logic 7 (2001), 1–36.

[3] S. Artemov, V. Krupski, *Data storage interpretation of labeled modal logic*, Annals of Pure and Applied Logic 78 (1996), 57–71.

[4] V. N. Krupski, *Operational logic of proofs with functionality condition on proof predicate*, Lecture notes in computer science 1234 (1997), 167–177.

[5] V. Krupski, *The single-conclusion proof logic and inference rules specification*, Annals of Pure and Applied Logic 113 (2002), 181-206.

[6] V. Krupski, *Referential logic of proofs*, Theoretical Computer Science, to appear.

[7] R. Kuznets, *On the complexity of explicit modal logics*, Computer Science Logic 2000, Lecture Notes in Computer Science 1862 (2000), 371–383.

[8] A. Mkrtychev, *Models for the Logic of Proofs*, Lecture Notes in Computer Science, v. 1234 (1997), *Logical Foundations of Computer Science '97, Yaroslavl'*, pp. 266–275.

[9] N.Rubtcova, T.Yavorskaya, *Operations on proofs and labels*, Journal of applied nonclassical logic, to appear.

[10] C. Smorynsky, *Self-reference and Modal Logic*, Springer, New York, 1985.

[11] T. Yavorskaya (Sidon), *Logic of proofs and provability*, Annals of Pure and Applided logic 113 (2002), 345-372.