

Logic of proofs and labels

Tatiana Yavorskaya
Moscow State University

August 2007

Plan.

1. Introduction to Logic of Proofs (S. Artemov, 1995)
2. Why do we need labels and what are they?
3. Operations on proofs and labels and the logics which describe them.
5. Example. Logic \mathcal{LPS}^* .
6. Logic of substitution (N. Rubtsova, 2003)

Propositional Logic of Proofs

S. Artemov, *Explicit provability and constructive semantics*, BSL 7(2001).

Provability logic: $\Box A$ means “ A is provable (in PA)”

Logic of proofs: eliminate existential quantifiers hidden in \Box and replace them by special terms representing concrete proofs.

$$\Box A \rightarrow \Box B$$

\Downarrow

“for every p there exists q s.t. if p is a proof of A then q is a proof of B ”

\Downarrow

“for every p , if p is a proof of A then $f(p)$ is a proof of B ”

\Downarrow

$$\llbracket p \rrbracket A \rightarrow \llbracket f(p) \rrbracket B$$

$$\llbracket t \rrbracket A \approx \text{“}t \text{ is a proof of } A\text{”}$$

Language of Logic of Proofs

The language contains objects of two types:

- proof terms for proofs; these are constructed from proof variables and constants by operations \times , $+$, $!$

$$t ::= ProofVar \mid ProofConst \mid t \times t \mid t + t \mid !t$$

- formulas for propositions; they are constructed from propositional variables by Booleans and proof operator

$$F ::= SentenceVar \mid F \rightarrow F \mid \neg F \mid F \wedge F \mid F \vee F \mid \llbracket t \rrbracket F$$

In the intended semantics

$$\llbracket t \rrbracket F \Rightarrow \text{“}t \text{ is a proof of } F\text{”}.$$

What can be expressed in this language.

- In Provability Logic: $\Box A \rightarrow A$
True but not provable (Gödel Incompleteness Theorem II).

In Logic of Proofs: $\llbracket t \rrbracket A \rightarrow A$
Both true and provable.

- In Provability Logic: $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$

In Logic of Proofs: $\llbracket t \rrbracket (A \rightarrow B) \rightarrow (\llbracket s \rrbracket A \rightarrow \llbracket t \times s \rrbracket B)$

Axioms and rules of Logic of Proofs \mathcal{LP} .

Axioms of \mathcal{LP} are classical propositional ones plus

$\llbracket t \rrbracket A \rightarrow A$	<i>weak reflexivity</i>
$\llbracket t \rrbracket (A \rightarrow B) \rightarrow (\llbracket s \rrbracket A \rightarrow \llbracket t \times s \rrbracket B)$	<i>application</i>
$\llbracket s \rrbracket A \rightarrow \llbracket s + t \rrbracket A, \quad \llbracket t \rrbracket A \rightarrow \llbracket s + t \rrbracket A$	<i>nondeterministic choice</i>
$\llbracket t \rrbracket A \rightarrow \llbracket !t \rrbracket \llbracket t \rrbracket A$	<i>positive proof checker</i>

Inference rules are modus ponens and **axiom necessitation**

$\vdash \llbracket c \rrbracket A$, where c is a proof constant, A is an axiom.

Note. \mathcal{LP} can be interpreted in a formal system enable to encode its own proof predicate, i.e. PA . Then $\times, +, !$ are total recursive and a proof predicate should be multi-conclusion due to $+$.

The main results on \mathcal{LP} (S. Artemov, 1995)

I. Arithmetical interpretation is $* = (Prf, \times^*, +^*, !^*, (\cdot)^*)$ where Prf is a normal proof predicate for PA , $\times^*, +^*, !^*$ are total r.e. operations on proofs and $(\cdot)^*$ is an evaluation of variables.

Theorem. $\mathcal{LP} \vdash A \iff \forall * PA \vdash A^* \iff \forall * A^*$ is true.

II. Realization $(\cdot)^r$ of a modal formula A assigns proof terms to all occurrences of \Box in A ; the result A^r is an \mathcal{LP} -formula.

Theorem. $S4 \vdash A \iff \exists (\cdot)^r \mathcal{LP} \vdash A^r$.

These two theorems provide $S4$ and therefore intuitionistic logic with a provability semantics.

Operations on proofs.

Complex proofs are constructed from atomic ones by admissible inference rules. Any such rule generates an operation on proofs:

$$\begin{array}{l} \vdash A_1, \dots, \vdash A_n \Rightarrow \vdash F \quad \text{an admissible rule} \\ \Downarrow \\ \Box A_1 \wedge \dots \wedge \Box A_n \rightarrow \Box F \quad \Box \text{ stands for provability (in } PA) \\ \Downarrow \\ \llbracket p_1 \rrbracket A_1 \wedge \dots \wedge \llbracket p_n \rrbracket A_n \rightarrow \Box F \quad \text{equivalently, } \Box A \sim \exists p (\llbracket p \rrbracket A) \end{array}$$

An operation on proofs is a recursive procedure which finds a proof of F under the assumption that all p_i are proofs of A_i .

The parameters of such an operation are

- proofs, i.e. $\llbracket p \rrbracket (F \rightarrow G) \wedge \llbracket q \rrbracket F \rightarrow \Box G$
- propositions, i.e. $\llbracket p \rrbracket A \rightarrow \Box(A \vee B)$

Labels for propositions and storage predicates

We extend the class of operations by admitting

- **negative information**, e.g. $\neg[[p]]F \rightarrow \Box\neg[[p]]F$
- **choice**, e.g. $[[p]]A \vee [[q]]B \rightarrow \Box(A \vee B)$

In the examples above propositions are among parameters. Propositions can be long; it is more convenient to have special names for them. We call these names labels.

For example, “by theorem 3”, “from *Consis(PA)*” etc.

We develop the language with proofs, labels and a storage predicate

$$x \ni F \Leftrightarrow \text{“}x \text{ is a label for } F\text{”}.$$

Specification language \mathcal{LPS}_0

Specification language contains objects of three types:

- proof variables p, q, \dots
- label variables x, y, \dots
- formulas $F ::= SVar \mid F \rightarrow F \mid \neg F \mid F \wedge F \mid F \vee F \mid \llbracket p \rrbracket F \mid x \ni F$.

Informally: every label is assigned to a finite set of formulas; the functions “ $label \mapsto content$ ” and “ $formula \mapsto label$ ” are recursive.

Formally: an interpretation $*$ of \mathcal{LPS}_0 has three components:

- an evaluation of variables $(\cdot)^*$.
- a normal proof predicate $Prf(x, y)$; $(\llbracket p \rrbracket F)^* \rightleftharpoons Prf(p^*, \lceil F^* \rceil)$
- a storage predicate $Str(x, y)$ $(x \ni F)^* \rightleftharpoons Str(x^*, \lceil F^* \rceil)$

Proof and storage predicates.

A **proof predicate** is a Δ_1 -formula $Prf(x, y)$ s.t. $T \vdash \phi$ iff $PA \vdash Prf(n, [\phi])$ for some n .

Prf is **normal** if every n proves a finite set of φ and the function which maps n to these sets is total recursive. For multi-conclusion Prf any finite set of theorems has a common proof.

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

A **storage predicate** is a Δ_1 -formula $Str(x, y)$ s.t. every x labels a finite set of formulas and the function which maps x to these sets is total recursive.

$Store(x, y) \Leftrightarrow$ “ x is a Gödel number of a finite set of formulas, y is the number of one of them”

Specification of operations on proofs and labels.

A specification is an arithmetically valid formula $\delta \rightarrow \Box F$ where δ is a boolean combination of formulas of the form $\llbracket p_i \rrbracket A_i$ and $x_j \ni B_j$.

Given a specification $\delta \rightarrow \Box F$, there is an algorithm which finds a proof of F^* for every $(\cdot)^*$ provided that δ^* is true:

```
x := 0;  
if  $\delta^*$  then repeat inc(x) until  $Prf(x, [F^*])$ ;  
return x.
```

The parameters of the procedure are $(\cdot)^*$ and δ, F .

A complete specification: parameters of the procedure above are variables p_i and x_j .

Complete specifications: examples.

- $\llbracket p \rrbracket (S_1 \rightarrow S_2) \wedge \llbracket q \rrbracket S_1 \rightarrow \Box S_2$

$x := 0; \quad \varphi := \iota F. Prf(p, [F]); \quad \theta := \iota F. Prf(q, [F]);$
if $\varphi = (\theta \rightarrow \psi)$ then repeat $inc(x)$ until $Prf(x, [\psi])$;
return x

- $\llbracket p \rrbracket S \vee \llbracket q \rrbracket S \rightarrow \Box S$

$x := 0; \quad T := \{\varphi \mid Prf(p, [\varphi])\} \cup \{\varphi \mid Prf(q, [\varphi])\}$
if $T \neq \emptyset$ then repeat $inc(x)$ until $Prf(x, [\varphi])$ for all $\varphi \in T$;
return x

- $\llbracket p \rrbracket A \wedge \llbracket q \rrbracket \neg A \rightarrow \Box B$

return 0

Incomplete specifications: examples.

Incomplete specifications

$$\neg \llbracket p \rrbracket S \rightarrow \Box \neg \llbracket p \rrbracket S$$

$$\llbracket p \rrbracket S_1 \rightarrow \Box (S_1 \vee S_2)$$

$$\Box (S_1 \rightarrow (S_2 \rightarrow S_1))$$

$$\Box (\llbracket p \rrbracket S \rightarrow S)$$

But we can make them complete

$$\neg \llbracket p \rrbracket S \wedge (x \ni S) \rightarrow \Box \neg \llbracket p \rrbracket S$$

$$\llbracket p \rrbracket S_1 \wedge (x \ni S_2) \rightarrow \Box (S_1 \vee S_2)$$

$$(x \ni S_1) \wedge (y \ni S_2) \rightarrow \Box (S_1 \rightarrow (S_2 \rightarrow S_1))$$

$$(x \ni \llbracket p \rrbracket S) \rightarrow \Box (\llbracket p \rrbracket S \rightarrow S)$$

Completeness criteria.

Let $\delta \rightarrow \Box F$ be a specification where δ is a boolean combination of q -atoms $\llbracket p_k \rrbracket A_k$ and $x_j \ni B_j$. By \vec{p} and \vec{x} denote the sets of all p_k and x_j resp.

Find for δ a disjunctive normal form $\bigvee_{i=1}^n (\bigwedge \delta_i^+ \wedge \bigwedge \delta_i^-)$, where δ_i^+ consists of q -atoms and δ_i^- consists of negated q -atoms.

Lemma 1. Specification is complete iff for all i either $\mathcal{LPS}_0 \vdash \neg(\bigwedge \delta_i \wedge \bigwedge \delta_i^-)$ or $\text{Var}(F) \subseteq \vec{p} \cup \vec{x} \cup \text{Var}(\delta_i^+)$.

Corollary. Definition of a complete specification is decidable.

Lemma 2. An incomplete specification can be made complete by adding assumptions of the form $x \ni C$ to δ .

Specification axioms

A complete specification: for every normal multi-conclusion Prf and every Str there exists a total r.e. function $f^*(\vec{p}, \vec{x})$ such that for every $(\cdot)^*$ the following formula is true

$$\delta^* \rightarrow Prf(f^*(\vec{p}^*, \vec{x}^*), \lceil F^* \rceil).$$

A specification axiom for a given complete specification $\delta \rightarrow \Box F$ and a fresh functional symbol f is a formula

$$\delta \rightarrow \llbracket f(\vec{p}, \vec{x}) \rrbracket F$$

Interpretation of f for a given pair Prf, Str , is a total recursive function f^* , s.t. for all $(\cdot)^*$,

$$PA \vdash \delta^* \rightarrow Prf(f^*(\vec{p}^*, \vec{x}^*), \lceil F^* \rceil)$$

Logic of proofs with operations $\mathcal{LPS}(\mathcal{F})$.

Let \mathcal{F} be a finite set of specification axioms. Define

$$\underline{\mathcal{LPS}(\mathcal{F})} = \text{propositional logic} + (\llbracket p \rrbracket A \rightarrow A) + \mathcal{F}$$

Theorem 1. If \mathcal{F} consists of “good” specifications then $\mathcal{LPS}(\mathcal{F})$ is decidable; the satisfiability problem is NP-complete.

Arithmetical \mathcal{F} -interpretation is $* = (\text{Prf}, \text{Str}, \mathcal{F}^*, (\cdot)^*)$.

Theorem 2. If \mathcal{F} consists of “good” specifications then $\mathcal{LPS}(\mathcal{F}) \vdash A \iff \forall *, PA \vdash A^* \iff \forall *, A^*$ is true.

Example: \mathcal{LPS}^* is $\mathcal{LPS}(\mathcal{G})$ where \mathcal{G} consists of

- application

$$\llbracket p \rrbracket (A \rightarrow B) \wedge \llbracket q \rrbracket A \rightarrow \llbracket p \cdot q \rrbracket B$$

- deterministic choice

$$(\llbracket p \rrbracket A \vee \llbracket q \rrbracket A) \wedge (x \ni A) \rightarrow \llbracket \text{choice}(p, q; x) \rrbracket A$$

- checkers

$$\llbracket p \rrbracket A \rightarrow \llbracket !_1(p) \rrbracket \llbracket p \rrbracket A \qquad \neg \llbracket p \rrbracket A \wedge (x \ni A) \rightarrow \llbracket ?_1(x, p) \rrbracket \neg \llbracket p \rrbracket A$$

$$(x \ni A) \rightarrow \llbracket !_2(x) \rrbracket (x \ni A) \qquad \neg(y \ni A) \wedge (x \ni A) \rightarrow \llbracket ?_2(x, y) \rrbracket \neg(y \ni A)$$

- propositional oracles

$$(x \ni A) \wedge (y \ni B) \rightarrow \llbracket ax_1(x, y) \rrbracket (A \rightarrow (B \rightarrow A)) \text{ etc.}$$

Properties of \mathcal{LPS}^*

Data base DB is a finite set of formulas of the form $(x_i \ni A_i)$

- For every Prf there exists Str such that \mathcal{F} can be realized.

Take any $Str(x, y)$ which assigns not more than one formula to a label.

- \mathcal{LPS}^* is decidable and arithmetically complete.

From the general theorem since all specification from \mathcal{F} are “good”.

- If $\mathcal{LPS}^* \vdash F$ then there exist t, DB such that $DB \vdash \llbracket t \rrbracket F$.

Induction on the derivation of F . For propositional axioms t is an instance of a propositional oracle. For Modus Ponens use application. Specification axioms and reflexivity are more interesting.

Delta-completeness in \mathcal{LPS}^*

- For δ a boolean combination of q -atoms there exist t, \mathcal{DB} such that $\mathcal{DB} \vdash \delta \rightarrow \llbracket t \rrbracket \delta$.

Induction on δ . Consider the case $\delta = \delta_1 \vee \delta_2$.

$$\begin{array}{ll}
 \mathcal{DB}_i \vdash \delta_i \rightarrow \llbracket t_i \rrbracket \delta_i & \text{by induction hypothesis} \\
 \llbracket u_i \rrbracket (\delta_i \rightarrow \delta_1 \vee \delta_2) & \text{by propositional oracles} \\
 \llbracket t_i \rrbracket \delta_i \rightarrow \llbracket u_i \cdot t_i \rrbracket (\delta_1 \vee \delta_2) & \text{application} \\
 \llbracket u_i t_i \rrbracket (\delta_1 \vee \delta_2) \rightarrow \llbracket \text{choice}(u_1 t_1, u_2 t_2; x) \rrbracket (\delta_1 \vee \delta_2) & \text{choice} \\
 \delta_1 \vee \delta_2 \rightarrow \llbracket \text{choice}(u_1 t_1, u_2 t_2; x) \rrbracket (\delta_1 \vee \delta_2) & \text{by propositional logic}
 \end{array}$$

Note. All specification axioms of \mathcal{LPS}^* are boolean combinations of q -atoms. So, if A is a specification axiom of \mathcal{LPS}^* then $\mathcal{DB} \vdash \llbracket t \rrbracket A$ for some t, \mathcal{DB} .

How \mathcal{LPS}^* proves reflexivity

- For every F , s there exist t , \mathcal{DB} such that $\mathcal{DB} \vdash \llbracket t \rrbracket (\llbracket s \rrbracket F \rightarrow F)$.
 1. $\llbracket c \rrbracket (F \rightarrow (\llbracket s \rrbracket F \rightarrow F))$ by a propositional oracle
 2. $\llbracket s \rrbracket F \rightarrow \llbracket c \cdot s \rrbracket (\llbracket s \rrbracket F \rightarrow F)$ application
 3. $\llbracket u \rrbracket (\neg \llbracket s \rrbracket F \rightarrow (\llbracket s \rrbracket F \rightarrow F))$ by a propositional oracle
 4. $\neg \llbracket s \rrbracket F \rightarrow \llbracket v \rrbracket \neg \llbracket s \rrbracket F$ by negative proof checker
 5. $\llbracket v \rrbracket \neg \llbracket s \rrbracket F \rightarrow \llbracket u \cdot v \rrbracket (\llbracket s \rrbracket F \rightarrow F)$ application; use 3
 6. $\neg \llbracket s \rrbracket F \rightarrow \llbracket u \cdot v \rrbracket (\llbracket s \rrbracket F \rightarrow F)$ by prop. logic from 4, 5
 7. $\llbracket c \cdot s \rrbracket (\llbracket s \rrbracket F \rightarrow F) \vee \llbracket u \cdot v \rrbracket (\llbracket s \rrbracket F \rightarrow F)$ by prop. logic from 2, 6
 8. $\llbracket \text{choice}(c \cdot s, u \cdot v; x) \rrbracket (\llbracket s \rrbracket F \rightarrow F)$ choice, add $x \ni (\llbracket s \rrbracket F \rightarrow F)$ to \mathcal{DB}

We extend the data base on steps 2,3,4 and 8.

\mathcal{LPS}^* and modal logic

Let \mathcal{L}_\square be propositional modal language with the modality \square .

A forgetful projection of an \mathcal{LPS}^* -formula A is an \mathcal{L}_\square -formula A_\circ obtained from A by replacing all proof terms by \square and all subformulas of the form $x \ni F$ by \top .

A realization of an \mathcal{L}_\square -formula B is an \mathcal{LPS}^* -formula B^r obtained by assigning \mathcal{LPS}^* -terms to all \square 's in B . A realization is *normal* if all negative \square 's are assigned proof variables.

Theorem 1. If $\mathcal{LPS}^* \vdash A$ then $S5 \vdash A_\circ$.

Theorem 2. If $S4 \vdash B$ then there exists a normal realization B^r and a data base \mathcal{DB} such that $\mathcal{DB} \vdash_{\mathcal{LPS}^*} B^r$.

Functional completeness

The set of operations \mathcal{F} is *f-complete* if for every specification $\delta \rightarrow \Box F$ there exists a data base \mathcal{DB} and a term $t \in Tm(\mathcal{F})$ such that $\mathcal{DB} \vdash_{\mathcal{LPS}(\mathcal{F})} \delta \rightarrow \llbracket t \rrbracket F$.

Theorem. The set of operations \mathcal{G} is *f-complete*.

To describe more operations, take $\mathcal{LPS}(\mathcal{F})$ as a specification language. For example, an “application checker”

$$\llbracket p \rrbracket (A \rightarrow B) \wedge \llbracket q \rrbracket A \rightarrow \Box \llbracket p \times q \rrbracket B.$$

An *\mathcal{F} -specification* is a formula $\delta \rightarrow \Box F$, which is true under every \mathcal{F} -interpretation, where δ is a boolean combination of q -atomic $\mathcal{LPS}(\mathcal{F})$ -formulas and F is an $\mathcal{LPS}(\mathcal{F})$ -formula.

Strong functional completeness

The set of operations \mathcal{F} is *s.f-complete* if for every set of “good” operations \mathcal{F}' , for every \mathcal{F}' -specification $\delta \rightarrow \Box F$ there exists \mathcal{DB} and a term $t \in Tm(\mathcal{F})$ such that $\mathcal{DB} \vdash \delta \rightarrow \llbracket t \rrbracket F$ in $\mathcal{LPS}(\mathcal{F}, \mathcal{F}')$.

Theorem. The set of operation \mathcal{G} is *s.f-complete*.

For example, let us express application checker by an \mathcal{LPS}^* -term:

$$\mathcal{LPS}^* \vdash \llbracket p \rrbracket (A \rightarrow B) \wedge \llbracket q \rrbracket A \rightarrow \llbracket !_1(p \times q) \rrbracket \llbracket p \times q \rrbracket B$$

Note. \mathcal{LP} is not *f-complete* (cannot realize negative checkers)

$$\neg \llbracket p \rrbracket A \wedge x \ni A \rightarrow \Box \neg \llbracket p \rrbracket A.$$

\mathcal{LP} suffices to realize operations specified by formulas with positive assumptions, but it is not *s.f-complete* in this class of operations.

Logic of substitution \mathcal{LPS}_{sub} (N.Rubtsova)

The aim is to describe substitution $L \vdash A \Rightarrow L \vdash A(B/S_i)$.

Arguments: $\llbracket p \rrbracket A, (x \ni B)$. Result: $\llbracket subst_i(x; p) \rrbracket A(B/S_i)$.

Language contains functional symbols

$$subst_i(x; p) : label \rightarrow (proof \rightarrow proof)$$

Proof and storage operators bind sentence variables:
in $\llbracket t \rrbracket B$ and $x \ni B$ all sentence variables are bound.

Substitutions replace only free variables. Otherwise

$$x \ni B, \llbracket p \rrbracket \llbracket q \rrbracket S_1 \vdash \llbracket subst_1(x; p) \rrbracket \llbracket q \rrbracket B.$$

So, the meaning of $\llbracket t \rrbracket F$ is different: "*t is a proof of a schema F*".

Logic of substitution – II

Axioms of \mathcal{LPS}_{sub} : propositional calculus plus

- another form of reflexivity

$$\llbracket t \rrbracket A \rightarrow A(\vec{B} / \vec{S})$$

- specification of substitution

$$(x \ni B) \wedge \llbracket t \rrbracket A \rightarrow \llbracket subst_i(x; t) \rrbracket A(B/S_i), \text{ for all } i$$

Note. \mathcal{LPS}_{sub} is not a particular case of $\mathcal{LPS}(\mathcal{F})$. The operation $subst_i$ cannot be specified by one schema because substitution of formulas is not in our language but in the metalanguage.

Theorem (N.Rubtsova). Logic \mathcal{LPS}_{sub} is decidable.

Arithmetical semantics for \mathcal{LPS}_{sub}

We extend the language of PA by propositional variables to incorporate substitution

- Axioms of PA' are formulated in the extended language. The following connection between PA and PA' holds:

$$PA' \vdash \varphi(\vec{S}) \iff PA \vdash \varphi(\vec{\psi}/\vec{S}) \text{ for all } \vec{\psi}$$

- We define Prf , Str for PA' in the usual way. Evaluation $(\cdot)^*$ maps proof and label variables to numbers and sentence variables to propositional variables of PA' .

Theorem (N.Rubtsova). $\mathcal{LPS}_{sub} \vdash A \iff PA' \vdash A^*$ for all $*$

Thank you!