Center for Foundations of Intelligent Systems

Technical Report
97-08

On Proof Realization of Intuitionistic Logic

S. N. ARTEMOV

October 1997

CORNELL
UNIVERSITY

625 Rhodes Hall, Ithaca, NY 14853 (607) 255-8005

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

Public Reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimates or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188,) Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 31 March 1998	3. REPORT TYPE AND DATES COVERED **Technical Report**
4. TITLE AND SUBTITLE **On Proof Realization of Intuitionistic Logic**		5. FUNDING NUMBERS **DAAH04-96-1-0341**
6. AUTHOR(S) **S. N. Artemov**		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Regents of the University of California** **c/o Sponsored Projects Office** **336 Sproul Hall** **Berkeley, CA 94720-5940**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211		10. SPONSORING / MONITORING AGENCY REPORT NUMBER ARO 35873.74-MA-MUR

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by the documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT	12 b. DISTRIBUTION CODE
Approved for public release; distribution unlimited.	

13. ABSTRACT (Maximum 200 words)

In 1933 Godel introduced an axiomatic system, currently known as S4, for a logic of an absolute provability. The problem of finding a fair probability model for S4 was left open. In the current paper we demonstrate how the intuitionistic propositional logic Int can be directly realized by proof polynomials. It is shown that Int is complete with respect to this proof realizability.

14. SUBJECT TERMS	15. NUMBER OF PAGES
proof polynomials, arithmetical semantics, S4	**17**
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

19980519 159

Technical Report
97-08

On Proof Realization of Intuitionistic Logic

S. N. ARTEMOV

October 1997

DTIC QUALITY INSPECTED

On Proof Realization of Intuitionistic Logic

Sergei N. Artemov*

Abstract

In 1933 Gödel introduced an axiomatic system, currently known as S4, for a logic of an absolute provability, i.e. not depending on the formalism chosen ([7]). The problem of finding a fair provability model for S4 was left open. The famous formal provability predicate which first appeared in the Gödel Incompleteness Theorem does not do this job: the logic of formal provability is not compatible with S4. As was discovered in [2], this defect of the formal provability predicate can be bypassed by replacing hidden quantifiers over proofs by *proof polynomials* in a certain finite basis. The resulting *Logic of Proofs* enjoys a natural arithmetical semantics and provides an intended provability model for S4, thus answering a question left open by Gödel in 1933. Proof polynomials give an intended semantics for some other constructions based on the concept of provability, including intuitionistic logic with its Brouwer-Heyting- Kolmogorov interpretation, λ-calculus and modal λ-calculus. In the current paper we demonstrate how the intuitionistic propositional logic $\mathcal{I}nt$ can be directly realized by proof polynomials. It is shown, that $\mathcal{I}nt$ is complete with respect to this proof realizability.

1 Introduction

A functional completeness theorem from [2] (cf. Section 5 below) demonstrates, that three basic operations on proofs: "$\cdot$"(*application*), "+" (*nondeterministic choice*), and "!" (*proof checker*) constitute a basis for all absolute operations on proofs, expressible in the propositional language. Along with *proof constants* for proofs of certain "simple facts", like propositional tautologies, these three operations "$\cdot$", "+", and "!" define so called *proof polynomials.* Logic of Proofs ($\mathcal{LP}$) extends a boolean logic by new formulas $[\![t]\!]F$, where t is a proof polynomial, and F a formula, with the intended reading "*t is a proof of F*" (cf. [2]).

The language of $\mathcal{LP}$ has an exact intended semantics, where "t is a proof of F" is interpreted as a corresponding arithmetical formula about the codes of t and F. The completeness and decidability of $\mathcal{LP}$ was established in ([2]).

The intuitionistic logic $\mathcal{I}nt$ was supplied ([8], [9], cf.[14], [6], [15]) with an informal Brouwer-Heyting-Kolmogorov (*BHK*) operational semantics , which was given in terms of

*Center for Foundations of Intelligent Systems, 625 Rhodes Hall, Cornell University, Ithaca NY, 14853 artemov@hybrid.cornell.edu. Research supported by the ARO under the MURI program "Integrated Approach to Intelligent Systems", grant number DAA H04-96-1-0341.

logical conditions on the formulas and their proofs, *e.g.* the "implication" clause is "p proves $A \to B$ iff p is a construction transforming any proof c of A into a proof $p(c)$ of B". In 1933 Gödel made a step to formalize *BHK* semantics by introducing a faithful embedding of $\mathcal{I}nt$ into a "natural born" provability logic $\mathcal{S}4$; this attempt has remained incomplete, since, in turn, $\mathcal{S}4$ has lacked the intended provability semantics.

$$\mathcal{I}nt \hookrightarrow \mathcal{S}4 \hookrightarrow ?$$

As it was also established in [2], an immediate forgetful translation of $\mathcal{LP}$ gives exactly $\mathcal{S}4$; in particular, there is a realization algorithm recovering proof polynomials in any $\mathcal{S}4$-proof. So, $\mathcal{LP}$ provides an intended provability interpretation for the modal logic $\mathcal{S}4$

$$\mathcal{S}4 \hookrightarrow \mathcal{LP} \hookrightarrow \textit{Arithmetic},$$

thus completing the Gödels embedding of $\mathcal{I}nt$ into $\mathcal{S}4$ to the fair arithmetical provability semantics for both $\mathcal{I}nt$ and $\mathcal{S}4$

$$\mathcal{I}nt \hookrightarrow \mathcal{S}4 \hookrightarrow \mathcal{LP} \hookrightarrow \textit{Arithmetic}.$$

In the current paper we give a direct realization algorithm of $\mathcal{I}nt$ into $\mathcal{LP}$, Gödel style. This proof realizability provides a fair semantics for $\mathcal{I}nt$:

$$\mathcal{I}nt \vdash F \quad \Leftrightarrow \quad F \textit{ is proof realizable.}$$

2 Logic of Proofs

The language of $\mathcal{LP}$ contains

boolean constants $\top, \bot$, sentence variables $p_0, \ldots, p_n, \ldots$,
proof variables $x_0, \ldots, x_n, \ldots$, proof constants $a_0, \ldots, a_n, \ldots$,
boolean connectives $\to, \ldots$,
functional symbols: monadic !, binary $+$ and $\cdot$,
operator symbol of the type $[\![\textit{term}]\!]\ (\textit{formula})$.

Terms (alias *proof polynomials*) and formulas are defined in a natural way: a proof variable and an axiom constant is a term; a sentence variable and a boolean constant is a formula; whenever s, t are terms $!t$, $(s+t)$, $(s \cdot t)$ are again terms, boolean connectives behave conventionally, and for t a term and F a formula $[\![t]\!]F$ is a formula. A term (proof polynomial) is *ground* if is does not contain variables.

We will write st instead of $s \cdot t$ and skip parentheses when convenient. If $\vec{x} = (x_1, \ldots, x_n)$ and $\Gamma = (A_1, \ldots, A_n)$, then we will write $[\![\vec{x}]\!]\Gamma$ for $[\![x_1]\!]A_1, \ldots, [\![x_n]\!]A_n$.

2.1 Definition. System $\mathcal{LP}$. The axioms are all formulas of the form

A0. All classical tautologies in the language of $\mathcal{LP}$

A1. $[\![t]\!]F \to F$ *"reflexivity"*

A2. $[\![t]\!](F \to G) \ \to ([\![s]\!]F \to [\![ts]\!]G)$ *"application"*

A3. $[\![t]\!]F \ \to \ [\![!t]\!][\![t]\!]F$ *"proof checker"*

A4. $[\![s]\!]F \ \to \ [\![s{+}t]\!]F, \qquad [\![t]\!]F \ \to \ [\![s{+}t]\!]F$ *"choice"*

Rules of inference:

$$\frac{\Gamma \vdash F, \ \ \Gamma \vdash F \to G}{\Gamma \vdash G} \quad \textit{for any formulas } F,\ G, \textit{ a set of formulas } \Gamma$$

"modus ponens"

$$\frac{\vdash F}{c{:}F} \quad \textit{for any formula } F, \textit{ and any constant } c$$

"necessitation"

With any derivation D in $\mathcal{LP}$ we associate a *Specification of Constants* (*SpeC*) that is a list $[\![c_1]\!]A_1, \ldots, [\![c_n]\!]A_n$ of all formulas introduced in D by the necessitation rule. Under $\mathcal{LP}_{SpeC} \vdash F$ we mean "there is a derivation of F in $\mathcal{LP}$ with the specification *SpeC*".

The intended understanding of $\mathcal{LP}$ is as a logic of operations on proofs, where $[\![t]\!]F$ stands for

"t is a proof of F".

For the usual Gödel proof predicate $Proof(x,y)$ in $\mathcal{PA}$ there are primitive recursive functions from codes of proofs to codes of proofs corresponding to "$\cdot$" and "!": "$\cdot$" stands for a operation on proof sequences which corresponds to the *modus ponens* rule, and "!" is a "proof checker" operation, appearing in the proof of the second Gödel Incompleteness theorem. The usual proof predicate has a natural nondeterministic version $PROOF(x,y)$ called *standard nondeterministic proof predicate*

"x is a code of a derivation containing a formula with a code y".

$PROOF$ already has all three operations of the $\mathcal{LP}$-language: the operation $s+t$ is now just a concatenation of (nondeterministic) proofs s and t.

A logic of the deterministic proof predicates is different from $\mathcal{LP}$ and is described in [1], [10].

2.2 Comment. System $\mathcal{LP}$ is not a multimodal logic, since no single modality $[\![t]\!](\cdot)$ satisfies the property $[\![t]\!](p \to q) \ \to \ ([\![t]\!]p \ \to \ [\![t]\!]q)$ in $\mathcal{LP}$. This makes $\mathcal{LP}$ different from numerous multimodal logics. However, the entire variety of labeled modalities in $\mathcal{LP}$ can emulate $\mathcal{S}4$([2], cf. Theorem 3.4).

2.3 Comment. The usual deduction theorem holds for $\mathcal{LP}$:

$$\Gamma, A \vdash_{\mathcal{LP}} B \ \ \Rightarrow \ \ \Gamma \vdash_{\mathcal{LP}} A \to B.$$

2.4 Lemma. (Substitution lemma for $\mathcal{LP}$). *If $\Gamma(x,p) \vdash_{\mathcal{LP}} B(x,p)$ for a propositional varaiable p and a proof variable x, then for any proof polynomial t and any formula F*

$$\Gamma(x/t, p/F) \vdash_{\mathcal{LP}} B(x/t, p/F).$$

Proof is trivial, since all axioms and rules of $\mathcal{LP}$ remain axioms and rules after a substitution.

2.5 Lemma. *The following rules are admissible in $\mathcal{LP}$. Here A,B are $\mathcal{LP}$-formulas, Γ, Δ are finite sets of $\mathcal{LP}$-formulas, y is a proof variable, t, r are proof polynomials, $\vec{y}$ and $\vec{s}$ are vectors of proof variables and proof polynomials correspondingly, "$\vdash$" means "$\vdash_{\mathcal{LP}}$".*

Necessitation: $$\frac{\vdash B}{\vdash [\![t]\!]B}$$ *for some ground t;*

Lifting: $$\frac{[\![\vec{s}]\!]\Gamma, \Delta \vdash B}{[\![\vec{s}]\!]\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![t(\vec{y})]\!]B}$$ *for some $t(\vec{y})$;*

Stripping: $$\frac{\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![t]\!]B}{\Gamma, \Delta \vdash B};$$
($\vec{y}$ does not occur in the conclusion)

Abstraction: $$\frac{[\![\vec{s}]\!]\Gamma, [\![y]\!]A \vdash [\![t(y)]\!]B}{[\![\vec{s}]\!]\Gamma \vdash [\![\lambda y.t(y)]\!](A \to B)}$$
for some proof polynomial denoted as $\lambda y.t(y)$
(y does not occur in the conclusion.)

Proof. *Necessitation* is a special case of *Lifting.*

Lifting. By induction on a proof of B from the premises $[\![\vec{s}]\!]\Gamma, \Delta$. If $B \in [\![\vec{s}]\!]\Gamma$, then $[\![\vec{s}]\!]\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![!s_i]\!]B$ for some $s_i \in \vec{s}$. If $B \in \Delta$, then $[\![y_j]\!]B \in [\![\vec{y}]\!]\Delta$. for some $y_j \in \vec{y}$. If B is an axiom A0 – A4, then $[\![c]\!]B$ is *SpecC.* If B is from A5, *i.e.* B is $[\![c]\!]A$ for some A from A0 – A4, then by A3, $\vdash [\![c]\!]A \to [\![!c]\!][\![c]\!]A$, and $[\![\vec{s}]\!]\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![!c]\!]B$. Let B be obtained from C,$C \to B$ by *modus ponens.* Then, by the induction hypothesis, $[\![\vec{s}]\!]\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![t_1(\vec{y})]\!](C \to B)$ and $[\![\vec{s}]\!]\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![t_2(\vec{y})]\!]C$. for some polynomials t_1 and t_2. By A2, $[\![\vec{x}]\!]\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![t_1 \cdot t_2]\!]B$

Stripping. From $\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![t]\!]B$ conclude $\Gamma, [\![\vec{y}]\!]\Delta \vdash B$. Note that none of the variables from $\vec{y} = (y_1, \ldots, y_n)$ occurs in Γ, Δ, B. Define an operation $'$ on $\mathcal{LP}$-formulas: $p = p'$ for a propositional variable p, $'$ commutes with boolean connectives and

$$([\![s]\!]F)' = \begin{cases} F', & \text{if } s \text{ contains a variable from } \vec{y} \\ [\![s]\!](F'), & \text{otherwise.} \end{cases}$$

By a straightforward induction on the derivation length show that for each F from the derivation $\Gamma, [\![\vec{y}]\!]\Delta \vdash B$

$$\begin{cases} \text{if } \Gamma, [\![\vec{y}]\!]\Delta \vdash F \text{ then } \Gamma, \Delta \vdash F' \\ \text{if } \vdash F \text{ then } \vdash F'. \end{cases}$$

In particular, $\Gamma, \Delta \vdash B$.

Case 1. F is from $\Gamma, [\![\vec{y}]\!]\Delta$. Easy, since $\Gamma' = \Gamma$ and $([\![\vec{y}]\!]\Delta)' = \Delta$.

Case 2. F is a propositional axiom. Then F' is the same axiom.

Case 3. $F = [\![s]\!]X \to X$.

a) s is $\vec{y}$-free. Then $F' = [\![s]\!]X' \to X'$, an axiom A1.

b) s is not $\vec{y}$-free. Then $F' = X' \to X'$.

Case 4. $F = [\![s]\!](X \to Y) \to ([\![r]\!]X \to lsrY)$.

a) s, r are both $\vec{y}$-free. Then F' is again an axiom A2.

b) s is $\vec{y}$-free, r is not. Then $F' = [\![s]\!](X' \to Y') \to (X' \to Y')$, axiom A1.

c) r is $\vec{y}$-free, s is not. Then $F' = (X' \to Y') \to ([\![r]\!]X' \to Y')$, derivable in $\mathcal{LP}$ since $[\![r]\!]X' \to Y'$.

d) s, r are both not $\vec{y}$-free. Then $F' = (X' \to Y') \to (X' \to Y')$.

Case 5. $F = [\![s]\!]X \to [\![!s]\!]([\![s]\!]X)$.

a) s is $\vec{y}$-free. Then F' is again an axiom A3.

b) s is not $\vec{y}$-free. Then $F' = X' \to X'$.

Case 6. $F = [\![s]\!]X \;\to\; [\![(s+r)]\!]X$.

a) s, r are both $\vec{y}$-free. Then F' is again an axiom A4.

b) s is $\vec{y}$-free, r is not. Then $F' = [\![s]\!]X' \to X'$, axiom A1.

c) s is not $\vec{y}$-free. Then $F' = X' \to X'$.

Case 7. $F = [\![t]\!]X \;\to\; [\![(s+r)]\!]X$. Similar to Case 6.

Case 8. F is obtained from $X, X \to F$ by *modus ponens*. Then F' is obtained from $G', X' \to X'$ by the same rule.

Case 9. $F = [\![s]\!]X$ is obtained by *necessitation* from X. Then $\vdash X$ and $\vdash [\![c]\!]X$ for some constant c. By the Induction hypothesis, $\vdash X'$. By the necessitation, $\vdash [\![s]\!]X'$.

Note that this proof delivers a linear time algorithm of transforming a derivation $\Gamma, [\![\vec{y}]\!]\Delta \vdash [\![t]\!]B$ into a derivation $\Gamma, \Delta \vdash B$.

Abstraction. From $[\![\vec{s}]\!]\Gamma, [\![y]\!]A \vdash [\![t(y)]\!]B$ by Stripping, get $[\![\vec{s}]\!]\Gamma, A \vdash B$, then by Deduction, $[\![\vec{s}]\!]\Gamma \vdash A \to B$, and then use Lifting to get $[\![\vec{s}]\!]\Gamma \vdash [\![r]\!](A \to B)$ for some proof polynomial r. ◀

2.6 Comment. A polynomial $t(\vec{y})$ introduced by the Lifting rule is nothing but a protocol of a proof of B from $[\![\vec{s}]\!]\Gamma, [\![\vec{y}]\!]\Delta$. The same holds for the rule of Abstraction, where $\lambda y.t(y)$ is a protocol of a proof of $A \to B$ from $[\![\vec{s}]\!]\Gamma$.

The Stripping rule is the only rule in this list which does not introduce a proof polynomial. Also, the proof of this rule does not look constructive. However, since $\mathcal{LP}$ is decidable there is a (primitive recursive) procedure, which constructs a proof from the conclusion given a proof from the premise. A more direct algorithm could be extracted from a cut-elimination theorem for $\mathcal{LP}$.

The Abstraction rule might not look like an operation on proofs either, because in the the process of constructing $\lambda y.t(y)$ from $t(y)$ we get rid of the latter and seemingly construct $\lambda y.t(y)$ from scratch. However, it is not the case. A polynomial $t(y)$ is a protocol of a proof of B from $[\![\vec{s}]\!]\Gamma, [\![y]\!]A$. From this proof we get a proof $[\![\vec{s}]\!]\Gamma, A \vdash B$, then a proof of $A \rightarrow B$ from $[\![\vec{s}]\!]\Gamma$. Finally, $\lambda y.t(y)$ is a protocol of the latter proof. All the procedures from this chain of transformations leading from $t(y)$ to $\lambda y.t(y)$ are constructive.

3 Realization of $\mathcal{S}4$ by proof polynomials .

3.1 Example. $\mathcal{S}4 \vdash (\Box A \wedge \Box B) \rightarrow \Box(A \wedge B)$. In $\mathcal{LP}$ this can be reproduced by the following:

1. $[\![c]\!](A \rightarrow (B \rightarrow A \wedge B))$, by necessitation
2. $[\![x]\!]A \rightarrow [\![cx]\!](B \rightarrow A \wedge B)$, from 1 and A2
3. $[\![x]\!]A \rightarrow ([\![y]\!]B \rightarrow [\![(cx)y]\!](A \wedge B))$, from 2 and A2
4. $[\![x]\!]A \wedge [\![y]\!]B \rightarrow [\![(cx)y]\!](A \wedge B)$ from 3 by propositional logic.

Here the specification of constants $SpeC$ consist of 1 only.

3.2 Example. $\mathcal{S}4 \vdash (\Box A \vee \Box B) \rightarrow \Box(A \vee B)$. In $\mathcal{LP}$ the corresponding derivation is

1. $[\![a]\!](A \rightarrow A \vee B)$, by necessitation
2. $[\![b]\!](B \rightarrow A \vee B)$, by necessitation
3. $[\![x]\!]A \rightarrow [\![ax]\!](A \vee B)$, from 1 and A2
4. $[\![y]\!]B \rightarrow [\![by]\!](A \vee B)$, from 2 and A2
5. $[\![x]\!]A \rightarrow [\![ax+by]\!](A \vee B)$, $[\![y]\!]B \rightarrow [\![ax+by]\!](A \vee B)$ by A4 from 3, 4.
6. $[\![x]\!]A \vee [\![y]\!]B \rightarrow [\![ax+by]\!](A \vee B)$, from 5 by propositional logic

Here the specification of constants consists of 1 and 2.

The fundamental fact about $\mathcal{S}4$ is that, **all** $\mathcal{S}4$-theorems have a corresponding dynamic reading in $\mathcal{LP}$.

3.3 Definition. By an *$\mathcal{LP}$-realization* $r = r(SpeC)$ of a modal formula F we mean

1. an assignment of proof polynomials to all occurrences of the modality in F,
2. a choice of a specification of constants $SpeC$;

Under F^r we denote the image of F under a realization r. Positive and negative occurrences of modality in a formula and a sequent are defined in the usual way. A realization r is *normal* if all negative occurrences of $\Box$ are realized by proof variables.

3.4 Theorem. ([2]) *If $\mathcal{S}4 \vdash F$, then $\mathcal{LP}_{SpeC} \vdash F^r$ for some specification of constants SpeC and some normal realization $r = r(SpeC)$.*

The proof describes an algorithm which for a given cut-free derivation $\mathcal{T}$ in $\mathcal{S}4$ assigns proof polynomials to all occurrences of the modality in $\mathcal{T}$.

3.5 Corollary.

$$\mathcal{S}4 \vdash F \quad \Leftrightarrow \quad \mathcal{LP} \vdash F^r \textit{ for some realization } r.$$

4 Arithmetical Semantics of proof polynomials

We use a new functional symbol $\iota z\varphi(z)$ for any arithmetical formula $\varphi(z)$ and assume that ι-terms could be eliminated in the usual way by using the small scope convention (cf. [5]). An arithmetical formula φ is *provably* Δ_1 iff both φ and $\neg\varphi$ are provably Σ_1. A term $\iota z\varphi$ is *provably recursive* iff φ is provably Σ_1. *Closed recursive term* is a provably total and provably recursive term $\iota z\varphi$ such that φ contains no free variables other than z.

> Close recursive terms represent all provably recursive names for natural numbers. We have to use all of them as proof realizers, since some operations on proofs, *e.g.* the *proof checker* "!", depend on the name of the argument, not on its value. Indeed, if $PROOF(\overline{n},\overline{k})$ holds, then $PROOF(\overline{n}+0,\overline{k})$ also holds, $!(\overline{n})$ is a proof of $PROOF(\overline{n},\overline{k})$ and $!(\overline{n}+0)$ is a proof of $PROOF(\overline{n}+0,\overline{k})$. However, $!(\overline{n})$ and $!(\overline{n}+0)$ deliver proofs of different formulas, thus, generally speaking, $!(\overline{n}) \neq !(\overline{n}+0)$.

A *proof predicate* is a provably Δ_1-formula $Prf(x,y)$ such that for all φ

$$\mathcal{PA} \vdash \varphi \quad \Leftrightarrow \quad \text{for some } n \in \omega \quad Prf(n, \ulcorner\varphi\urcorner) \text{ holds.}$$

A proof predicate *Prf(x,y)* is *normal* if
 1) for every proof k the set $T(k) = \{l \mid Prf(k,l)\}$ is finite and the function

$$\widetilde{T(k)} = \textit{the code of } T(k)$$

is provably recursive,
 2) for every finite set S of theorems of $\mathcal{PA}$, $S \subseteq T(k)$ for some proof k.

The nondeterministic proof predicate *PROOF* (above) is a normal proof predicate.

For every normal proof predicate Prf there are provably recursive terms $m(x,y)$, $a(x,y)$, $c(x)$ such that for all closed recursive terms s,t and for all arithmetical formulas φ,ψ the following formulas are valid:

$$Prf(s,\ulcorner\varphi\to\psi\urcorner)\wedge Prf(t,\ulcorner\varphi\urcorner)\to Prf(m(s,t),\ulcorner\psi\urcorner)$$

$$Prf(s,\ulcorner\varphi\urcorner)\to Prf(a(s,t),\ulcorner\varphi\urcorner),\qquad Prf(t,\ulcorner\varphi\urcorner)\to Prf(a(s,t),\ulcorner\varphi\urcorner)$$

$$Prf(t,\ulcorner\varphi\urcorner)\to Prf(c(\ulcorner t\urcorner),\ulcorner Prf(t,\ulcorner\varphi\urcorner)\urcorner).$$

Let $SpeC$ be a specification of constants. An arithmetical *SpeC-interpretation* $*$ of $\mathcal{LP}$-language has the following parameters: $SpeC$, a normal proof predicate Prf, an evaluation of sentence letters by sentences of arithmetic, an evaluation of proof letters and axiom constants by closed recursive terms. We put $\top^*\equiv(0=0)$ and $\bot^*\equiv(0=1)$, $*$ commute with boolean connectives, $(t\cdot s)^*\equiv m(t^*,s^*)$, $(t+s)^*\equiv a(t^*,s^*)$, $(!t)^*\equiv c(\ulcorner t^*\urcorner)$, $([\![t]\!]F)^*\equiv Prf(t^*,\ulcorner F^*\urcorner)$. We also assume, that $\mathcal{PA}\vdash G^*$ for all $G\in SpeC$.

Under any *SpeC-interpretation* $*$ a proof polynomial t becomes a closed recursive term t^* (*i.e.* a recursive name of a natural number), and an $\mathcal{LP}$-formula F becomes an arithmetical sentence F^*. In what follows "arithmetically $SpeC$-valid" means either "provable in $\mathcal{PA}$" or "true in the standard modal" under any $SpeC$-interpretation.

Note that the reflexivity principle is back, since $[\![t]\!]F\to F$ is provable in $\mathcal{PA}$ under any interpretation $*$. Indeed, let n be the value of t^*. If $Prf(n,\ulcorner F^*\urcorner)$ is true, then $\mathcal{PA}\vdash F^*$, thus $\mathcal{PA}\vdash Prf(n,\ulcorner F^*\urcorner)\to F^*$. If $Prf(n,\ulcorner F^*\urcorner)$ is false, then $\mathcal{PA}\vdash\neg Prf(n,\ulcorner F^*\urcorner)$, and again $\mathcal{PA}\vdash Prf(n,\ulcorner F^*\urcorner)\to F^*$.

4.1 Theorem. ([2], Arithmetical completeness of $\mathcal{LP}$)

$$\mathcal{LP}_{SpeC}\vdash F\quad\Leftrightarrow\quad F^*\ \textit{is arithmetically SpeC-valid}\,.$$

Combining 3.4 and 4.1, we obtain the arithmetical completeness of $\mathcal{S}4$:

$$\mathcal{S}4\vdash F\quad\Leftrightarrow\quad F^r\ \textit{is arithmetically valid for some realization } r \textit{ and some specification of constants SpeC.}$$

Gödel in [7] defined a translation tr of intuitionistic formulas, into $\mathcal{S}4$-formulas where $tr(F)$ is obtained from F by boxing all atoms and all implications in F. This Gödel translation is shown ([7], [11]) to provide a faithful embedding of $\mathcal{I}nt$ into $\mathcal{S}4$. The proof interpretation of $\mathcal{LP}$-polynomials above provides a faithful proof arithmetical realization of $\mathcal{I}nt$:

$$\mathcal{I}nt\vdash F\quad\Leftrightarrow\quad [tr(F)]^r\ \textit{is arithmetically valid for some normal realization } r \textit{ and some specification of constants SpeC.}$$

5 Functional completeness of proof polynomials

We recall a result from [2] that proof polynomials represent *all* absolute propositional operations on proofs. The basic operations $\cdot, !, +$ thus play for proofs a role similar to that boolean connectives play for classical logic.

Consider an arbitrary scheme of a specification of an operation of proofs in arithmetic. Such a specification is an arithmetical formula

$$\forall \vec{x} \in C\ \exists y \text{ “}y \textit{ is a proof of } G(\vec{x})\text{”},$$

or, equivalently

$$\forall \vec{x}(C(\vec{x}) \rightarrow \exists y \text{ “}y \textit{ is a proof of } G(\vec{x})\text{”}),$$

true in the standard model of arithmetic, where C and G are arbitrary arithmetical conditions. A *propositional* specification language should contain tools to express a notion "x is a proof of F", at least for a proof variable x. Let SL be a language with

boolean constants $\top, \bot$, sentence variables $p_0, \ldots, p_n, \ldots$
proof variables $x_0, \ldots, x_n, \ldots$
boolean connectives $\rightarrow, \ldots$
operator symbol $[\![term]\!]\ (formula)$.

Note, that SL is a fragment of the language of $\mathcal{LP}$ where no functions are presupposed. The only proof terms in the specification language are the proof variables. Now we can make precise the following question:

what absolute operations on proofs can be specified by a propositional language?

To answer this question we assume that $C(\vec{x})$ and $G(\vec{x})$ are conditions in the specification language SL. Also, we express the existential quantifier $\exists y$ "*y is a proof of $G(\vec{x})$*" by the usual provability modality $\Box$, extending the definition of F^* by one more item : $(\Box F)^*$ is $\exists x Prf(x, \ulcorner F^* \urcorner)$, i.e. by the arithmetical *provability* predicate associated with a proof predicate Prf from an interpretation $*$.

Finally we restrict C's to "*positive*" conditions, *i.e.* the ones where the outermost subformulas $[\![x]\!]F$ occur positively.

Indeed, a condition of the sort

$$\neg[\![x]\!]\, P \rightarrow \Box\neg[\![x]\!]P,$$

although valid for any proof predicate, may hardly be accepted as a specification of an operation on proofs equally as good as $\cdot, !, +$, because it derives conclusions from *negative* information about proofs, i.e. from "*x IS NOT a proof of a formula*".

It seems that now we have found a balanced definition of an operation on proofs. The regular case

$$[\![x_1]\!]C_1 \wedge \ldots \wedge [\![x_n]\!]C_n \rightarrow \Box G,$$

which comes from the straightforward formalization of the notion of an admissible inference rule

$$\frac{C_1, \ldots, C_n}{G}$$

is covered. Further shrinking of C to conjunctions of formulas $[\![x_1]\!]C_1 \wedge \ldots \wedge [\![x_n]\!]C_n$ only would eliminate natural and useful nondeterministic proof systems.

5.1 Definition. We may define now an *absolute propositional operation on proofs* as a formula $C \rightarrow \Box G$, valid under all arithmetical interpretations, where C, G are formulas in the specification language SL and C is positive.

5.2 Comment. Operations $\cdot, !, +$ can be identified as absolute propositional operations on proofs. Indeed, formulas

$[\![x_1]\!](F \rightarrow G) \wedge [\![x_2]\!]\, F \rightarrow \Box\, G$

$[\![x]\!]F \rightarrow \Box[\![x]\!]\, F$

$[\![x_1]\!]F \vee [\![x_2]\!]F \rightarrow \Box F$

are valid under every arithmetical translation and Skolem functions for the existential quantifiers on proofs in $\Box$'s here can be realized by $m(x_1, x_2)$, $c(x)$, $a(x_1, x_2)$ from Section 4 correspondingly.

The following theorem from ([2]) demonstrates that proof polynomials and Logic of Proofs suffice to realize any absolute propositional operation on proofs.

5.3 Theorem. ([2]) *For any abstract propositional operation on proofs $C \rightarrow \Box G$ there exists a proof polynomial t such that*

$$C \rightarrow [\![t]\!]G$$

is derivable in the Logic of Proofs, and thus arithmetically valid.

6 Proof polynomials *vs.* Provability Logic.

The Logic of Proofs gives a formalization of the arithmetical provability operator different from the one of the Provability Logic. In a certain sense, the Logic of Proofs introduces a new propositional language which is designed to get rid of the hidden quantifiers on proofs. The intended interpretation of a formula of the $\mathcal{LP}$-language gives a provably decidable arithmetical sentence, provided the evaluations of the propositions are. As a result, there is do direct way to interpret the Second Gödel Incompleteness theorem into $\mathcal{LP}$. The fixed point construction from [2] which establishes the arithmetical completeness of $\mathcal{LP}$ is totally different from the one used by R. Solovay in his proof of the arithmetical completeness of the Provability Logic (cf. [4]). However, the proof polynomials and the Provability Logic are clearly compatible; in

[1] in the proof of the arithmetical completeness of the system $\mathcal{B}$ it was shown how to build the arithmetical fixed point for the Logic of Proofs (without operations) in the top of the Solovay fixed point.

A natural problem of combining proof polynomials with formal provability operator within one logical system was solved recently by Tanya Sidon in [13]. Along with usual proof polynomials her logic contains two more operations, which rise in connection with the modality for a formal provability.

7 Proof polynomials *vs.* Modal Logic.

By 3.4, the Logic of Proofs is a version of $\mathcal{S}4$ presented in a more rich operational language, with no information being lost, since $\mathcal{S}4$ is the the exact term-forgetting projection of $\mathcal{LP}$. An easy inspection of the realizing algorithm shows that

$$\mathcal{LP}\text{-fomula} \quad = \quad \mathcal{S}4\text{-formula} + \text{its } \mathcal{S}4\text{-proof}.$$

A transliretating of an $\mathcal{S}4$-theorem into $\mathcal{LP}$-language may result in an exponential growth of its length. However, this increase looks much less dramatic if we calculate the complexity of the input $\mathcal{S}4$-theorem F in an "honest" way as the length of a proof of F in $\mathcal{S}4$: the proof polynomials appearing in the realization algorithm have a size linear of the length of the proof, so, the total length of an $\mathcal{LP}$-realization of an $\mathcal{S}4$-formula F is bounded by the quadratic function of the length of a given $\mathcal{S}4$-proof of F.

The decomposition of the $\mathcal{S}4$-modality into a finitely generated set of proof polynomials is a general fact, which may be used in other applications of the modal logic. Similar dynamic decompositions of the modalities could be done for some other major modal logics: *K, K4, S5, etc.*. However, $\mathcal{S}4$ is the one which corresponds to the provability reading of polynomials arising from this dynamic readings of the modalities.

8 Proof polynomials *vs.* Intuitionistic logic.

Kleene recursive realizability (cf. [14]) of the intuitionistic language does not use the logical provability constraints from the original *BHK* formulation and referes to all recursive functions, not just operations on proofs. As a result, too many formulas become realizable, more than $\mathcal{I}nt$ can derive:

$$\mathcal{I}nt \subsetneq \textit{Kleene realizable formulas}^1.$$

Proof realizability of $\mathcal{I}nt$ can be defined as a superposition of the realizations of $\mathcal{S}4$ in $\mathcal{LP}$ and $\mathcal{LP}$ in the arithmetic (above); $\mathcal{I}nt$ turns out to be complete with respect to the proof realizability

$$\mathcal{I}nt \quad = \quad \textit{proof realizable formulas}.$$

[1] Unless a metatheory is restricted, cf. [12].

In addition to the general algorithm of realization of $\mathcal{S}4$ in $\mathcal{LP}$ (3.4), we describe now its "light" version, which realizes $\mathcal{I}nt$ in $\mathcal{LP}$ directly.

We assume, that $\mathcal{I}nt$ is presented in the language with $\{\wedge, \vee, \rightarrow, \bot\}$ and recall, that the Gödel translation of an $\mathcal{I}nt$-formula F into a $\mathcal{S}4$-formula *tr(F)* consists in prefixing all subformulas in F by $\Box$ (we agree to skip $\Box$ prefixes of $\bot$). Our realization algorithm extends this Gödel translation to $\mathcal{LP}$-formulas.

We consider a cut-free sequential formulation of $\mathcal{S}4$, with sequents $\Gamma \Rightarrow \Delta$, where Γ and Δ are multisets of modal formulas. Axioms are sequents of the form $F \Rightarrow F$, where F is a formula. Along with usual structural rules and rules introducing boolean connectives there are two proper modal rules

$$\frac{A, \Gamma \Rightarrow \Delta}{\Box A, \Gamma \Rightarrow \Delta}\,(\Box \Rightarrow) \qquad \text{and} \qquad \frac{\Box\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A}\,(\Rightarrow \Box)$$

(A is a formula, Γ, Δ - multisets of formulas, $\Box\{A_1, \ldots, A_n\} = \{\Box A_1, \ldots, \Box A_n\}$).

Step 1. Take a sequential cut-free derivation of F in $\mathcal{I}nt$ with the axioms "$p \Rightarrow p$", where p is a propositional letter, and "$\bot\Rightarrow$ ". Replace every formula G in this derivation by its Gödel translation *tr(G)*. The resulting tree $\mathcal{T}$ is an "almost" $\mathcal{S}4$-derivation of *tr(F)* with the axioms of the form "$\Box p \Rightarrow \Box p$" with p a propositional letter, and "$\bot\Rightarrow$ ". More precisely, every $\mathcal{S}4$-sequent in $\mathcal{T}$ is provable in $\mathcal{S}4$; moreover, each step down in $\mathcal{T}$ can be regarded as a corresponding standard combination of $\mathcal{S}4$-rules, excluding *Cut*. For example, the intuitionistic rule

$$\frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \rightarrow B}\,(\Rightarrow \rightarrow)$$

will be presented as

$$\cfrac{\cfrac{\Box A', \Box\Gamma' \Rightarrow \Box B'}{\Box\Gamma' \Rightarrow \Box A' \rightarrow \Box B'}\,(\Rightarrow \rightarrow)}{\Box\Gamma' \Rightarrow \Box(\Box A' \rightarrow \Box B')}\,(\Rightarrow \Box).$$

Here under $\Box F'$ we mean a Gödel translation of an intuitionistic formula F. Similarly, all other intuitionistic rules of the "introduction to the right", and only them, produce a combination of $\mathcal{S}4$-rules, which contains the rule $(\Rightarrow \Box)$. Intuitionistic axiom sequents "$p \Rightarrow p$" become axiom sequents "$\Box p \Rightarrow \Box p$", axioms "$\bot \Rightarrow$ remain unchanged.

All the following steps are an adoption of the general realizing algorithm of $\mathcal{S}4$ into $\mathcal{LP}$ for $\mathcal{T}$.

Occurrences of $\Box$ in $\mathcal{T}$ are related if they occur in related formulas in premises and conclusions of nodes in $\mathcal{T}$; we extend this relationship by transitivity. All occurrences of $\Box$ in $\mathcal{T}$ are now naturally split into disjoint *families* of related ones. Since polarities of the $\Box$'s are respected in $\mathcal{T}$ we may speak about negative and positive families of related $\Box$'s. Two families are *close* if they contain $\Box$'s from an axiom $\Box p \Rightarrow \Box p$. We call a positive family *essential* if it contains at least one $\Box$ introduced by the $(\Rightarrow \Box)$ rule. In the tree $\mathcal{T}$, essential $\Box$'s appear only at the nodes, corresponding to the rules of introduction to the succedent. Since all $\Box$'s at the axiom nodes of $\mathcal{T}$ correspond to the atomic formulas, there are no essential $\Box$'s at leaves (axiom nodes). The basic observation here is that *no negative family is close to an essential positive family.* Indeed, every $(\Rightarrow \Box)$-rule introduces a prefix $\Box$ to a composite formula, not a sentence letter.

Step 2. Realize each negative family and each nonessential positive family by a fresh proof variable, realize close families by the same proof variable.

Step 3. For every essential positive family f enumerate all the nodes where the principal $\Box$ has been introduced, and let n_f be the total number of such nodes for a family f. Realize all $\Box$'s in an essential positive family f by the polynomial

$$(u_1 + \ldots + u_{n_f}),$$

where u_i's are fresh proof variables, which we call *provisional* variables. The resulting tree is called an *evaluated tree.*

Step 4. Perform the following leaves-root procedure of replacing provisional variables by proof polynomials, which will result in the desired realization r. After this procedure passes a node and perform corresponding changes of the labeling sequent $\Gamma \Rightarrow \Delta$, we will have

$$\Gamma \vdash_{\mathcal{LP}} \Delta. \qquad (\dagger)$$

The case of an axiom node $\Box p \Rightarrow \Box p$ in $\mathcal{T}$, is trivial, since the corresponding $\mathcal{LP}$-realization is $[\![x]\!]p \Rightarrow [\![x]\!]p$ for some proof variable x.

At the nodes of the evaluated tree, corresponding to the introduction to the antecedent rules, we don't perform any substitutions. It is an easy exercise in a propositional logic to verify, that the property $(\dagger)$ is respected.

A $(\Rightarrow \rightarrow)$ node in the evaluated tree looks like

$$\frac{[\![y]\!]A, [\![\vec{x}]\!]\Gamma \Rightarrow [\![s]\!]B}{[\![\vec{x}]\!]\Gamma \Rightarrow [\![t_1 + \ldots + u_i + \ldots + t_{n_f}]\!]([\![y]\!]A \rightarrow [\![s]\!]B)},$$

where u_i is a provisional variable, corresponding to this particular node. By the induction hypothesis,

$$[\![y]\!]A, [\![\vec{x}]\!]\Gamma \vdash_{\mathcal{LP}} [\![s]\!]B.$$

By the Deduction rule for $\mathcal{LP}$,

$$[\![\vec{x}]\!]\Gamma \vdash_{\mathcal{LP}} [\![y]\!]A \rightarrow [\![s]\!]B,$$

and by Lifting, there exists a proof polynomial $t(\vec{x})$ such that

$$[\![\vec{x}]\!]\Gamma \vdash_{\mathcal{LP}} [\![t(\vec{x})]\!]([\![y]\!]A \to [\![s]\!]B).$$

Substitute everywhere in the tree $t(\vec{x})$ for u_i. Since $t(\vec{x})$ does not contain provisional variables, u_i is no longer present in the evaluated tree. By the substitution lemma the property (†) survives for all sequents in the tree. Clearly,

$$[\![\vec{x}]\!]\Gamma \vdash_{\mathcal{LP}} [\![t_1+\ldots+t+\ldots+t_{n_f}]\!]([\![y]\!]A \to [\![s]\!]B).$$

The remaining cases of ($\Rightarrow \wedge$)-nodes and ($\Rightarrow \vee$)-nodes are treated similarly. After the process reaches the root, no provisional variables remain in the tree, the assignment of proof polynomials to the $\Box$'s in the root sequent is the desired realization of this sequent in $\mathcal{LP}$.

Since an $\mathcal{I}nt$-formula φ may be identified with the sequent $\Rightarrow \varphi$, we may define a realizer of φ as a ground proof polynomial r realizing the sequent $\Rightarrow \varphi$; the resulting evaluated tree will then have the root $\Rightarrow [\![r]\!]\widetilde{\varphi}$ for some $\mathcal{LP}$-formula $\widetilde{\varphi}$. This r is a protocol of the derivation of $\Rightarrow \widetilde{\varphi}$.

The completeness theorem for proof realizations

$$\varphi \textit{ is provable in } \mathcal{I}nt \quad\Leftrightarrow\quad \varphi \textit{ is proof realizable}$$

follows now from the fairness of the embeddings

$$\mathcal{I}nt \hookrightarrow \mathcal{S}4 \hookrightarrow \mathcal{LP} \hookrightarrow \textit{Arithmetic}.$$

8.1 Example. The $\mathcal{I}nt$-derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{A\Rightarrow A}{A\Rightarrow A\vee B}\quad \bot\Rightarrow\bot}{\neg(A\vee B),A\Rightarrow\bot}}{\neg(A\vee B)\Rightarrow\neg A}\qquad \dfrac{\dfrac{\dfrac{B\Rightarrow B}{B\Rightarrow A\vee B}\quad \bot\Rightarrow\bot}{\neg(A\vee B),B\Rightarrow\bot}}{\neg(A\vee B)\Rightarrow\neg B}}{\neg(A\vee B)\Rightarrow\neg A\wedge\neg B}$$

produces the following evaluated tree (we use $t{:}F$ instead $[\![t]\!]F$ to simplify the picture):

$$\dfrac{\dfrac{\dfrac{\dfrac{x{:}A\Rightarrow x{:}A}{x{:}A\Rightarrow u_1{+}u_2{:}(x{:}A\vee y{:}B)}\quad \bot\Rightarrow\bot}{z{:}\neg u_1{+}u_2{:}(x{:}A\vee y{:}B),x{:}A\Rightarrow\bot}}{z{:}\neg u_1{+}u_2{:}(x{:}A\vee y{:}B)\Rightarrow v{:}\neg x{:}A}\qquad \dfrac{\dfrac{\dfrac{y{:}B\Rightarrow y{:}B}{y{:}B\Rightarrow u_1{+}u_2{:}(x{:}A\vee y{:}B)}\quad \bot\Rightarrow\bot}{z{:}\neg u_1{+}u_2{:}(x{:}A\vee y{:}B),y{:}B\Rightarrow\bot}}{z{:}\neg u_1{+}u_2{:}(x{:}A\vee y{:}B)\Rightarrow w{:}\neg y{:}B}}{z{:}\neg u_1{+}u_2{:}(x{:}A\vee y{:}B)\Rightarrow p{:}(v{:}\neg x{:}A\wedge w{:}\neg y{:}B)}$$

Here u_1, u_2, v, w, p are provisional proof variables corresponding to all four essential positive families in the tree. According to the algorithm, all the provisional variables will be evaluated by polynomials rising from the lifting lemma used at the corresponding nodes.

The variable u_1 should be specified at the node labeled by the sequent

$$[\![x]\!]A \Rightarrow [\![u_1{+}u_2]\!]([\![x]\!]A \vee [\![y]\!]B).$$

For that we apply Lifting to

$$[\![x]\!]A \vdash_{\mathcal{LP}} [\![x]\!]A \vee [\![y]\!]B.$$

In this particular case it is easy to write down a polynomial for u_1 explicitly. Let a be a proof constant satisfying

$$\vdash_{\mathcal{LP}} [\![a]\!]([\![x]\!]A \rightarrow ([\![x]\!]A \vee [\![y]\!]B)).$$

Since also

$$[\![x]\!]A \vdash_{\mathcal{LP}} [\![!x]\!][\![x]\!]A,$$

we have

$$[\![x]\!]A \Rightarrow [\![a{\cdot}!x]\!]([\![x]\!]A \vee [\![y]\!]B),$$

and u_1 should be evaluated by $a{\cdot}!x$. Similarly, u_2 should be evaluated by $b{\cdot}!y$, where b is a proof constant specified by the condition

$$\vdash_{\mathcal{LP}} [\![b]\!]([\![y]\!]B \rightarrow ([\![x]\!]A \vee [\![y]\!]B)).$$

To find a polynomial $s(z)$ for v consider a node labeled by

$$[\![z]\!]\neg[\![a!x{+}b!y]\!]([\![x]\!]A \vee [\![y]\!]B) \Rightarrow [\![v]\!]\neg[\![x]\!]A.$$

From its preceeding sequent we have

$$[\![z]\!]\neg[\![a!x{+}b!y]\!]([\![x]\!]A \vee [\![y]\!]B), [\![x]\!]A \vdash_{\mathcal{LP}} \perp,$$

by the deduction lemma, we get

$$[\![z]\!]\neg[\![a!x{+}b!y]\!]([\![x]\!]A \vee [\![y]\!]B) \vdash_{\mathcal{LP}} \neg[\![x]\!]A,$$

and by Lifting, we get $s(z)$ such that

$$[\![z]\!]\neg[\![a!x{+}b!y]\!]([\![x]\!]A \vee [\![y]\!]B) \vdash_{\mathcal{LP}} [\![s(z)]\!]\neg[\![x]\!]A.$$

Similarly, we evaluate w by a polynomial $r(z)$ such that

$$[\![z]\!]\neg[\![a!x{+}b!y]\!]([\![x]\!]A \vee [\![y]\!]B) \vdash_{\mathcal{LP}} [\![r(z)]\!]\neg[\![y]\!]B.$$

Finally, the provisional variable p is evaluated by a polynomial $t(z)$ such that

$$[\![z]\!]\neg[\![a!x{+}b!y]\!]([\![x]\!]A \vee [\![y]\!]B) \vdash_{\mathcal{LP}} [\![t(z)]\!]([\![v]\!]\neg[\![x]\!]A \wedge [\![w]\!]\neg[\![y]\!]B).$$

9 Proof polynomials *vs.* typed λ-calculi.

The rule of the λ-abstraction can be realized as an admissible rule of inference in the Logic of Proofs (lemma 2.5). This shows a way to realize the entire types λ-calculus in $\mathcal{LP}$ by emulating the formation rules for λ-terms by the corresponding admissible rules in $\mathcal{LP}$. This realization gives a direct arithmetical provability semantics for the types λ-calculus. A straightforward combination of realization algorithms for the modal logic $\mathcal{S}4$ and for the types λ-calculus gives a realization procedure for the modal λ-calculus [3].

Acknowledgements.

The research described in this publication was supported in part by the Russian Foundation for Basic Research, grant 96-01-01222, and by ARO under the MURI program "Integrated Approach to Intelligent Systems", grant number DAAH04-96-1-0341.

References

[1] S. Artëmov, "Logic of Proofs," *Annals of Pure and Applied Logic*, v. 67 (1994), pp. 29-59.

[2] S. Artëmov, "Operational Modal Logic," *Tech. Rep. MSI 95-29*, Cornell University, December 1995.

[3] S. Artëmov, "Proof realizations of typed λ-calculi," *Tech. Rep. MSI 97-02*, Cornell University, May 1997.

[4] G. Boolos, *Logic of Provability.*, CUP, 1993.

[5] D. van Dalen, *Logic and Structure*, Springer-Verlag, 1994.

[6] J.-Y. Girard, Y. Lafont and P. Taylor, *Proofs and Types*, Cambridge University Press, 1989.

[7] K. Gödel, "Eine Interpretation des intuitionistischen Aussagenkalkuls", *Ergebnisse Math. Colloq.*, Bd. 4 (1933), S. 39-40.

[8] A. Heyting, "Die intuitionistische Grundlegung der Mathematik", *Erkenntnis*, Bd. 2 (1931), S. 106-115.

[9] A. Kolmogoroff, "Zur Deutung der intuitionistischen Logik," *Math. Ztschr.*, Bd. 35 (1932), S.58-65.

[10] V.N. Krupski, "Operational Logic of Proofs with Functionality Condition on Proof Predicate", Lecture Notes in Computer Science, v. 1234, *Logical Foundations of Computer Science' 97, Yaroslavl'*, pp.167-177, 1997

[11] J.C.C. McKinsey and A. Tarski, "Some theorems about the sentential calculi of Lewis and Heyting", *Journal of Symbolic Logic,* v. 13 (1948), pp. 1-15.

[12] V.E. Plisko, "On arithmetic complexity of certain constructive logics", transl from *Mat Zametki,*v. 52, pp. 701-709, 1992.

[13] T. Sidon, "Provability Logic with Operations on Proofs", Lecture Notes in Computer Science, v. 1234, *Logical Foundations of Computer Science' 97, Yaroslavl'*, pp. 342-353, 1997

[14] A.S. Troelstra and D. van Dalen, *Constructivism in Mathematics. An Introduction*, v. 1, Amsterdam; North Holland, 1988.

[15] A.S. Troelstra and H. Schwichtenberg, *Basic Proof Theory*, Cambridge University Press, 1996.