Deep isomorphism of modal derivations and λ-terms

Sergei N. Artemov *

June, 1999

Abstract

The traditional Curry-Howard isomorphism given a derivation of F from assumptions Γ in intuitionistic modal logic constructs the λ-term $t(\vec{x})$ such that the sequent $\vec{x}:\Gamma \Rightarrow t(\vec{x}):F$ is derivable in the corresponding calculus of typed modal λ-terms. Technically speaking this isomorphism is a realization by λ-terms of the outermost modalities in some (not all) derivable modal sequents $\Box\Gamma \Rightarrow \Box F$.

In this talk we describe a system of λ-terms (in the combinatory terms form) **LPi** sufficient for a uniform realization of all modalities (not just the outermost ones) in any derivable modal sequent. The expressive power of **LPi** surpasses the one of the usual modal λ-calculus. **LPi** also extends the natural provability semantics of Curry-Howard terms under which $t:F$ can be interpreted as "t is a proof of F in a given sufficiently rich system, e.g. Heyting Arithmetic **HA**".

1 Introduction

The traditional Curry-Howard isomorphism given a derivation $\Gamma \Rightarrow F$ constructs a λ-term $t(\vec{x})$ such that the sequent $\vec{x}:\Gamma \Rightarrow t(\vec{x}):F$ is derivable in the corresponding calculus of typed λ-terms. Here term $t(\vec{x})$ is nothing but the blueprint of a natural deduction derivation of F from the assumptions Γ. One can note, however, that modal logic also has a tool for internal specification of a given derivation of $\Gamma \Rightarrow F$, namely, the necessitation rule

$$\frac{\Gamma \Rightarrow F}{\Box\Gamma \Rightarrow \Box F}\ .$$

*Department of Computer Science, 4130 Upson Hall, Cornell University, Ithaca NY, 14853 U.S.A. and Moscow University, Russia. `Email:artemov@cs.cornell.edu`

The intended meaning of this rule is that given a modal derivation of the fact that $\Gamma \Rightarrow F$ one can construct a derivation of the fact

$$\text{“\textit{there are proofs of formulas from} } \Gamma\text{”} \;\Rightarrow\; \text{“\textit{there is a proof of} } F\text{”.}$$

An equivalent reading of the latter is

“for each x there is y such that if x is a proof of Γ then y is a proof of F”;

the above combination of quantifiers $\forall x \exists y \ldots$ in principle specifies y as a function of x.

The major advantage of the λ-terms presentation is that it gives explicit term $t(\vec{x})$ for a proof of F depending on the variables $\vec{x}$ denoting unspecified proofs of Γ (versus the implicit denotation of those proofs as $\Box$'s in modal logic). The main advantage of the modal language here is its ability to iterate modal operator and to mix it with the other connectives, that allows expressions such as $\Box F \to F$, $\Box F \to \Box\Box F$, $(\Box F \vee \Box G) \to \Box(F \vee G)$, etc.

The usual Curry-Howard modal λ-term $t{:}F$ may be considered as a explicating construction which realizes the outermost $\Box$'s in $\Box\Gamma$ and $\Box F$ but leaves the internal $\Box$'s in Γ and F untouched.

In this talk we describe the propositional calculus **LPi** based on intuitionistic logic with additional atoms $t{:}F$ where t is a term (proof polynomial) and F a formula. The system **LPi** may be regarded as both the explicit version of the intuitionistic modal logic **IS4** and as a system of dependent combinatory terms with additional operations “proof checker” and “choice” and the most importantly with the possibility to iterate the type assignment “:” and to mix “:” with the intuitionistic connectives.

In this talk we present two major facts about **LPi**

All modalities in any given derivation in intuitionistic modal logic **IS4** *can be realized by explicit* **LPi***-terms .* (Section 4)

LPi *has a natural provability semantics under which $t{:}F$ can be interpreted as “t is a proof of F in a given sufficiently rich system, e.g. Heyting Arithmetic* **HA**”. (Section 7).

Modulo to the standard realization of λ-abstraction operator in the typed combinatory logic, the above results extend the Curry-Howard isomorphism of modal derivations and modal λ-terms from its “shallow” form to the “deep” one where all modalities in $\Box\Gamma \Rightarrow \Box F$ are realized by explicit proof terms.

We will also show that the expressive power of **LPi** surpasses the one of the modal λ-calculus (Section 6).

2 System LPi

2.1 Definition. (cf. [3]) The language of **LPi** contains

the usual language of classical propositional logic
proof variables $x_0, \ldots, x_n, \ldots$, proof constants $a_0, \ldots, a_n, \ldots$
function symbols: monadic !, binary $\cdot$ and $+$
operator symbol of the type "*term : formula*".

We will use $a, b, c, \ldots$ for proof constants, $u, v, w, x, y, z, \ldots$ for proof variables. Terms in **LPi** are called *proof polynomials* and can be defined by the grammar

$$p ::= x_i \mid a_i \mid !p \mid p_1 \cdot p_2 \mid p_1 + p_2.$$

By analogy we refer to constants as coefficients. Constants correspond to proofs of a finite fixed set of propositional schemas. We will also omit $\cdot$ whenever it is safe. We also assume that $(a \cdot b \cdot c)$, $(a \cdot b \cdot c \cdot d)$, *etc.* should be read as $((a \cdot b) \cdot c)$, $(((a \cdot b) \cdot c) \cdot d)$, etc.

Using t to stand for any proof polynomial and S for any propositional letter, the formulas are defined by the grammar

$$\sigma ::= S \mid \sigma_1 \rightarrow \sigma_2 \mid \sigma_1 \wedge \sigma_2 \mid \sigma_1 \vee \sigma_2 \mid \neg\sigma \mid t : \sigma$$

We will use $A, B, C, F, G, H, X, Y, Z$ for the formulas in this language, and $\Gamma, \Delta, \ldots$ for the finite sets (also finite multisets, or finite lists) of formulas. We will also use $\vec{x}, \vec{y}, \vec{z}, \ldots$ and $\vec{p}, \vec{r}, \vec{s}, \ldots$ for vectors of proof variables and proof polynomials respectively. If $\vec{s} = (s_1, \ldots, s_n)$ and $\Gamma = (F_1, \ldots, F_n)$, then $\vec{s} : \Gamma$ denotes $(s_1 : F_1, \ldots, s_n : F_n)$, $\bigvee \Gamma = F_1 \vee \ldots \vee F_n$, $\bigwedge \Gamma = F_1 \wedge \ldots \wedge F_n$. We assume the following precedences from highest to lowest: $!, \cdot, +, :, \neg, \wedge, \vee, \rightarrow$.

The intended semantics for $p : F$ is "p is a proof of F", which will be formalized in the next section. Note that proof systems which provide a formal semantics for $p : F$ are *multi-conclusion* ones, i.e. p may be a proof of several different F's.

2.2 Definition. Hilbert style formulation of **LPi**.

Axioms and rules:

A0. Finite set of axiom schemes of intuitionistic propositional logic in the language of **LPi**

A1. $t : F \rightarrow F$ *"verification"*
A2. $t : (F \rightarrow G) \rightarrow (s : F \rightarrow (t \cdot s) : G)$ *"application"*
A3. $t : F \rightarrow\ !t : (t : F)$ *"proof checker"*
A4. $s : F \rightarrow (s+t) : F, \quad t : F \rightarrow (s+t) : F$ *"choice"*

Rules of inference:

R1. $$\frac{F \rightarrow G \qquad F}{G}$$ *"modus ponens".*

R2. $\dfrac{}{c : \mathbf{A}}$, *if* $\mathbf{A}$ *is an axiom* $A0$ – $A4$, *and* c *a proof constant.*

A *Constant Specification* ($\mathcal{CS}$) is a finite set of formulas $c_1 : A_1, \ldots, c_n : A_n$ such that c_i is a constant, and A_i an axiom $A0$ – $A4$. Each derivation in **LP** naturally generates the $\mathcal{CS}$ consisting of all formulas introduced in this derivation by *R2*.

2.3 Comment. Proof constants in **LPi** stand for proofs of "simple facts", namely propositional axioms and axioms *A1* – *A4*. Proof constants may be regarded as atomic constant terms (*combinators*) of typed combinatory logic (cf. [9]). A constant c_1 specified as $c_1 : (A \to (B \to A))$ can be identified with the combinator $\mathbf{k}^{A,B}$ of the type $A \to (B \to A)$. A constant c_2 such that $c_2 : [(A \to (B \to C)) \to ((A \to B) \to (A \to C))]$ corresponds to the combinator $\mathbf{s}^{A,B,C}$ of the type $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$. The proof variables may be regarded as term variables of combinatory logic, the operation "$\cdot$" as the application of terms. In general an **LPi**-formula $t : F$ can be read as a combinatory term t of the type F. Typed combinatory logic $\mathbf{CL}_{\to}$ thus corresponds to a fragment of **LPi** consisting only of formulas of the sort $t : F$ where t contains no operations other than "$\cdot$" and F is a formula built from the propositional letters by "$\to$" only.

There is no restriction on the choice of a constant c in *R2* within a given derivation. In particular, *R2* allows to introduce a formula $c : A(c)$, or to specify a constant several times as a proof of different axioms from *A0* – *A4*.

The system **LPi** admits the deduction theorem

$$\Gamma, A \vdash B \quad \Rightarrow \quad \Gamma \vdash A \to B,$$

and the substitution lemma: *If* $\Gamma(x, P) \vdash_{\mathbf{LPi}} B(x, P)$ *for a propositional variable* P *and a proof variable* x*, then for any proof polynomial* t *and any formula* F

$$\Gamma(x/t, P/F) \vdash_{\mathbf{LPi}} B(x/t, P/F).$$

2.4 Proposition. (Lifting lemma) *Given a derivation* $\mathcal{D}$ *of the type*

$$\vec{s} : \Gamma, \Delta \vdash_{\mathbf{LPi}} F$$

one can construct a proof polynomial $t(\vec{x}, \vec{y})$ *such that*

$$\vec{s} : \Gamma, \vec{y} : \Delta \vdash_{\mathbf{LPi}} t(\vec{s}, \vec{y}) : F.$$

Proof. By induction on the derivation $\vec{s}{:}\Gamma, \Delta \vdash F$. If $F = s_i{:}G_i \in \vec{s}{:}\Gamma$, then put $t :=!s_i$ and use *A3*. If $F = D_j \in \Delta$, then put $t := y_j$. If F is an axiom *A0* – *A4*, then pick a fresh proof constant c and put $t := c$; by *R2*, $F \vdash c{:}F$. Let F be introduced by *modus ponens* from $G \to F$ and G. Then, by the induction hypothesis, there are proof polynomials $u(\vec{s}, \vec{y})$ and $v(\vec{s}, \vec{y})$ such that $u{:}(G \to F)$ and $v{:}G$ are both derivable in **LPi** from $\vec{s}{:}\Gamma, \vec{y}{:}\Delta$. By *A2*, $\vec{s}{:}\Gamma, \vec{y}{:}\Delta \vdash (uv){:}F$, and we put $t := uv$. If F is introduced by *R2*, then $F = c{:}A$ for some axiom A. Use the same *R2* followed by *A3*: $c{:}A \to !c{:}c{:}A$, to get $\vec{s}{:}\Gamma, \vec{y}{:}\Delta \vdash !c{:}F$, and put $t :=!c$.
◀

It is easy to see from the proof that the lifting polynomial $t(\vec{x}, \vec{y})$ is nothing but a blueprint of $\mathcal{D}$. Thus **LPi** accommodates its own proofs as proof polynomials. The necessitation rule

$$\vdash_{\mathbf{LPi}} F \quad \Rightarrow \quad \vdash_{\mathbf{LPi}} p{:}F \quad \textit{for some proof polynomial } p$$

is a special case of Lifting. Note that p is a ground polynomial, i.e. it does not contain variables.

3 Gentzen style formulation of LPi

By *sequent* we mean a pair $\Gamma \Rightarrow \Delta$, where Γ and Δ are finite multisets of **LPi**-formulas and Δ contains not more then one formula.

Axioms of **LPGi** are sequents of the form $\Gamma, F \Rightarrow F$ and $\Gamma, \bot \Rightarrow \Delta$. Along with the usual Gentzen sequent rules for intuitionistic propositional logic, including the cut and construction rules (e.g. like **G2c** from [9]), the system **LPGi** contains the rules

$$\frac{A, \Gamma \Rightarrow \Delta}{t{:}A, \Gamma \Rightarrow \Delta}\ (: \ \Rightarrow) \qquad\qquad \frac{\Gamma \Rightarrow t{:}A}{\Gamma \Rightarrow !t{:}t{:}A}\ (\Rightarrow !)$$

$$\frac{\Gamma \Rightarrow t{:}A}{\Gamma \Rightarrow (t+s){:}A}\ (\Rightarrow +) \qquad\qquad \frac{\Gamma \Rightarrow t{:}A}{\Gamma \Rightarrow (s+t){:}A}\ (\Rightarrow +)$$

$$\frac{\Gamma \Rightarrow s{:}(A \to B) \qquad \Gamma \Rightarrow t{:}A}{\Gamma \Rightarrow (s \cdot t){:}B}\ (\Rightarrow \cdot)$$

$$\frac{}{\Gamma \Rightarrow c{:}\mathbf{A}}\ (\Rightarrow c),$$

where **A** is an axiom *A0* – *A4* from section 2, and c is a proof constant.

3.1 Proposition. ([2], [3]) $\mathbf{LPGi} \vdash \Gamma \Rightarrow F$ *iff* $\mathbf{LPi} \vdash \bigwedge \Gamma \to F$.

The proof proceeds by a straightforward induction both ways.

The cut elimination theorem for **LPGi** was established in [2], [3].

4 Realizing IS4 in LPi

LPi may be regarded as an explicit version of the intuitionistic modal logic **IS4**. Not only the forgetful projection of every theorem of **LPi** is provable in **IS4**, but every theorem F of **IS4** admits an instantiation of the modalities by proof polynomials such that the resulting formula F^r is derivable in **LPi**. The following example show how the realization of **IS4** in **LPi** works.

4.1 Example. $\mathbf{IS4} \vdash (\Box A \vee \Box B) \to \Box(A \vee B)$.

In **LPi** the corresponding derivation is

1. $a{:}(A \to A \vee B)$, $\quad b{:}(B \to A \vee B)$, by *R2*,
3. $x{:}A \to (a{\cdot}x){:}(A \vee B)$, $\quad y{:}B \to (b{\cdot}y){:}(A \vee B)$, from 1, by *A2*,
4. $(a{\cdot}x){:}(A \vee B) \to (a{\cdot}x+b{\cdot}y){:}(A \vee B)$, $\quad (b{\cdot}y){:}(A \vee B) \to (a{\cdot}x+b{\cdot}y){:}(A \vee B)$, by *A4*,
5. $(x{:}A \vee y{:}B) \to (a{\cdot}x+b{\cdot}y){:}(A \vee B)$, from 3, by propositional logic.

Let F^o be the result of substituting $\Box X$ for all occurrences of $t{:}X$ in F, and $\Gamma^o = \{F^o \mid F \in \Gamma\}$.

4.2 Lemma. *If* $\mathbf{LPi} \vdash F$*, then* $\mathbf{IS4} \vdash F^o$.

Proof. This is a straightforward induction on a derivation in **LPi**.
◄

Now we will establish the converse: **LPi** suffices to realize any **IS4** theorem. By an **LPi**-*realization* of a modal formula F we mean an assignment of proof polynomials to all occurrences of the modality in F, Let F^r be 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.

4.3 Theorem. *If* $\mathbf{IS4} \vdash F$*, then* $\mathbf{LPi} \vdash F^r$ *for some normal realization* r.

Proof. This proof is a straightforward adaptation of the corresponding fact for the classical modal logic **S4** and the classical logic of proofs **LP** (cf. [1], [2]. Consider a cut-free sequent formulation of **IS4** ([4]), with sequents of the form

$\Gamma \Rightarrow F$. Axioms are sequents of the form $S \Rightarrow S$, where S is a propositional letter. Along with the usual structural rules and rules introducing boolean connectives there are two proper modal rules:

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

$(\Box\{A_1, \ldots, A_n\} = \{\Box A_1, \ldots, \Box A_n\})$.

If $\mathbf{IS4} \vdash F$, then there exists a cut-free derivation $\mathcal{T}$ of a sequent $\Rightarrow F$. It suffices now to construct a normal realization r such that $\mathbf{LPi} \vdash \bigwedge \Gamma^r \to F^r$ for any sequent $\Gamma \Rightarrow F$ in $\mathcal{T}$. We will also speak about a sequent $\Gamma \Rightarrow F$ being derivable in **LPi** meaning $\mathbf{LPi} \vdash \bigwedge \Gamma \to F$, or, equivalently, $\Gamma \vdash_{\mathbf{LPi}} F$. Note that in a cut-free derivation $\mathcal{T}$ the rules respect polarities, all occurrences of $\Box$ introduced by $(\Rightarrow \Box)$ are positive, and all negative occurrences are introduced by $(\Box \Rightarrow)$ or by weakening. Occurrences of $\Box$ are *related* if they occur in related formulas of premises and conclusions of rules; we extend this relationship by transitivity. All occurrences of $\Box$ in $\mathcal{T}$ are naturally split into disjoint *families* of related ones. We call a family *essential* if it contains at least one case of the $(\Rightarrow \Box)$ rule.

Now the desired r will be constructed by steps 1 – 3 described below. We reserve a large enough supply of proof variables as *provisional variables.*

Step 1. For every negative family and non essential positive family we replace all occurrences of $\Box$ by "x:" for a fresh proof variable x.

Step 2. Pick an essential family f, enumerate all the occurrences of rules $(\Rightarrow \Box)$, which introduce boxes of this family. Let n_f be the total number of such rules for the family f. Replace all boxes of the family f by the term

$$(v_1 + \ldots + v_{n_f}),$$

where v_i's are fresh provisional variables. The resulting tree $\mathcal{T}_0$ is labelled by **LPi** formulas, since all occurrences of the kind $\Box X$ in $\mathcal{T}$ are replaced by $t\!:\!X$ for the corresponding t.

Step 3. Replace the provisional variables by proof polynomials as follows. Proceed from the leaves of the tree to its root. By induction on the depth of a node in $\mathcal{T}_0$ we establish that after the process passes a node, a sequent assigned to this node becomes derivable in **LPi**. The axioms $S \Rightarrow S$ are derivable in **LPi**. For every rule other than $(\Rightarrow \Box)$ we do not change the realization of formulas, and just establish that the concluding sequent is provable in **LPi** given that the premises are. Moreover, every move down in the tree $\mathcal{T}_0$ other than $(\Rightarrow \Box)$ is a rule of the Gentzen style system for **LPi**, therefore, the induction steps corresponding to these moves are routine.

Let an occurrence of the rule $(\Rightarrow \Box)$ have number i in the numbering of all rules $(\Rightarrow \Box)$ from a given family f. This rule already has a form

$$\frac{y_1\!:\!Y_1,\ldots,y_k\!:\!Y_k \Rightarrow Y}{y_1\!:\!Y_1,\ldots,y_k\!:\!Y_k \Rightarrow (u_1+\ldots+u_{n_f})\!:\!Y}\ ,$$

where $y_1,\ldots,y_k$ are proof variables, $u_1,\ldots,u_{n_f}$ are proof polynomials, and u_i is a provisional variable. By the induction hypothesis, the premise sequent $y_1\!:\!Y_1,\ldots,y_k\!:\!Y_k \Rightarrow Y$ is derivable in **LPi**, which yields a derivation of

$$y_1\!:\!Y_1,\ldots,y_k\!:\!Y_k \Rightarrow Y.$$

By Lifting Lemma 2.4, construct a proof polynomial $t(y_1,\ldots,y_n)$ such that

$$y_1\!:\!Y_1,\ldots,y_k\!:\!Y_k \Rightarrow t(y_1,\ldots,y_n)\!:\!Y$$

is derivable in **LPi**. Since

$$\mathbf{LPi} \vdash t\!:\!Y \rightarrow (u_1+\ldots+u_{i-1}+t+u_{i+1}+\ldots+u_{n_f})\!:\!Y$$

we have

$$\mathbf{LPi} \vdash y_1\!:\!Y_1,\ldots,y_k\!:\!Y_k \Rightarrow (u_1+\ldots+u_{i-1}+t+u_{i+1}+\ldots+u_{n_f})\!:\!Y.$$

Now substitute $t(y_1,\ldots,y_n)$ for u_i everywhere in the tree $\mathcal{T}_0$. Note, that $t(\vec{y})$ has no provisional variables, there is one provisional variable (namely u_i) less in the entire $\mathcal{T}_0$. All sequents derivable in **LPi** remain derivable, the conclusion of the given rule $(\Rightarrow \Box)$ became derivable, and the induction step is complete.

Eventually, we substitute terms of non-provisional variables for all provisional variables in $\mathcal{T}_0$ and establish that the corresponding root sequent of $\mathcal{T}_0$ is derivable in **LPi**. Note that the realization of $\Box$'s built by this procedure is normal.

◀

5 Defined λ-abstraction in LPi

The defined abstraction operator λ^*x on proof polynomials below is a natural extension of the defined λ-abstraction operator λ^*x in combinatory logic (cf. [9]).

5.1 Definition. Under *ground* $(\Rightarrow !)$ rule we mean the rule $(\Rightarrow !)$ where the principal proof polynomial t contains no variables. An **LPGi**-derivation $\mathcal{D}$ is *pure* if it uses no rules other than $(\Rightarrow \cdot)$, $(\Rightarrow c)$, and ground $(\Rightarrow !)$. It is clear that every pure derivation is normal since it has no cuts.

5.2 Theorem. (Defined abstraction) *Let $\mathcal{D}$ be a pure* **LPGi**-*derivation of a sequent*

$$\vec{p}\!:\!\Gamma, x\!:\!A \Rightarrow t(x)\!:\!B$$

such that x does not occur in $\vec{p}:\Gamma, A, B$. Then one may construct a proof polynomial $\lambda^* x.t(x)$ and a pure **LPGi**-*derivation $\mathcal{D}'$ of the sequent*

$$\vec{p}\!:\!\Gamma \Rightarrow \lambda^* x.t(x)\!:\!(A\!\rightarrow\! B).$$

Proof. The base case is the depth of $\mathcal{D}$ equals one. There are two possibilities.

1. $\mathcal{D}$ is an axiom sequent $\vec{p}:\Gamma, x:A \Rightarrow t(x):B$ and $t(x)$ contains an occurrence of x. Then $t(x):B = x:A$. Let $\mathcal{D}'$ be the pure derivation of the sequent $\Rightarrow (a\cdot b\cdot c)\!:\!(A\!\rightarrow\! A)$ where a,b,c are proof constants specified by the conditions (cf.[9], section 1.3.6.)

$a\!:\!([A\!\rightarrow\!((A\!\rightarrow\! A)\!\rightarrow\! A)]\!\rightarrow\![(A\!\rightarrow\!(A\!\rightarrow\! A))\!\rightarrow\!(A\!\rightarrow\! A)])$
$b\!:\![A\!\rightarrow\!((A\!\rightarrow\! A)\!\rightarrow\! A)]$
$c\!:\![A\!\rightarrow\!(A\!\rightarrow\! A)]$.

Let $\lambda^* x.x = (a\cdot b\cdot c)$. In fact this case coincides with the presentation of $\lambda^* x^A.x$ as $\mathbf{s}^{A,A\rightarrow A,A}\mathbf{k}^{A,A\rightarrow A}\mathbf{k}^{A,A}$ in combinatory logic (cf. [9]).

2. $\mathcal{D}$ is an axiom sequent $\vec{p}\!:\!\Gamma, x:A \Rightarrow t(x):B$ and t does not contain an occurrence of x. Then $t\!:\!B \in \vec{p}\!:\!\Gamma$ and $\vec{p}\!:\!\Gamma \Rightarrow t\!:\!B$ is again an axiom sequent. Let $\mathcal{D}'$ be

$$\dfrac{\dfrac{}{\vec{p}\!:\!\Gamma \Rightarrow b\!:\!(B\!\rightarrow\!(A\!\rightarrow\! B))}(\Rightarrow c) \qquad \vec{p}\!:\!\Gamma \Rightarrow t\!:\!B}{\vec{p}\!:\!\Gamma \Rightarrow (b\cdot t)\!:\!(A\!\rightarrow\! B)}(\Rightarrow\cdot)\ .$$

Let $\lambda^* x.t = b\cdot t$. This is the well known equality $\lambda^* x^A.t^B = \mathbf{k}^{B,A}t^B$ of combinatory logic.

The induction step corresponding to the ground $(\Rightarrow !)$ rule is treated similarly to case 2.

Consider the case $(\Rightarrow\cdot)$. Let a derivation $\mathcal{D}$ end with

$$\dfrac{\vec{p}\!:\!\Gamma, x\!:\!A \Rightarrow s\!:\!(Y\!\rightarrow\! B) \qquad \vec{p}\!:\!\Gamma, x\!:\!A \Rightarrow t\!:\!Y}{\vec{p}\!:\!\Gamma, x\!:\!A \Rightarrow (s\cdot t)\!:\!B}\ .$$

By the induction hypothesis, we have already built pure derivations of $\vec{p}:\Gamma \Rightarrow \lambda^* x.s : (A\rightarrow (Y\rightarrow B))$ and $\vec{p}:\Gamma \Rightarrow \lambda^* x.t : (A\rightarrow Y)$. From them we construct pure derivations

$$\dfrac{\overline{\vec{p}\!:\!\Gamma \Rightarrow c\!:\!((A\!\to\!(Y\!\to\!B))\!\to\!((A\!\to\!Y)\!\to\!(A\!\to\!B)))} \qquad \vec{p}\!:\!\Gamma\!\Rightarrow\!\lambda^* x.s\!:\!(A\!\to\!(Y\!\to\!B))}{\vec{p}\!:\!\Gamma \Rightarrow (c\cdot\lambda^* x.s)\!:\!((A\!\to\!Y)\!\to\!(A\!\to\!B))}$$

and

$$\dfrac{\vec{p}\!:\!\Gamma \Rightarrow (c\cdot\lambda^* x.s)\!:\!((A\!\to\!Y)\!\to\!(A\!\to\!B)) \qquad \vec{p}\!:\!\Gamma \Rightarrow \lambda^* x.t\!:\!(A\!\to\!Y)}{\vec{p}\!:\!\Gamma \Rightarrow (c\cdot\lambda^* x.s\cdot\lambda^* x.t)\!:\!(A\!\to\!B)}\,.$$

Let $\lambda^* x.(s\cdot t) = (c\cdot\lambda^* x.s\cdot\lambda^* x.t)$. In combinatory logic notation

$$\lambda^* x^A.s^{Y\to B}t^Y \;=\mathbf{s}^{A,Y,B}\lambda^* x.s\lambda^* x.t$$

◀

5.3 Comment. In **LPGi**λ-abstraction is decoded by a proof polynomials depending on a context (e.g. an **LPGi**-derivation). In this respect the realization of λ-abstraction by proof polynomials is similar the realization of **IS4**-modality which is decomposed in Section 4 into a set of proof polynomials depending on a context (an **IS4**-derivation).

5.4 Comment. In fact λ^* cannot be easily extended from pure to more general derivations without sacrificing some desired properties. We need to keep the format $\vec{p}:\Gamma, x:A \Rightarrow t(x):B$ throughout the entire derivation $\mathcal{D}$ in order to preserve the inductive character of the definition. The restriction "x does not occur in $\vec{p}\!:\!\Gamma, A, B$" is needed to guarantee the correctness of β-conversion (below) for λ^*-abstraction, though it rules out $(\Rightarrow !)$. Note that the rule $(\Rightarrow !)$ does not admit abstraction anyway. Indeed, in **LPGi** we may derive

$$\dfrac{x\!:\!A \Rightarrow x\!:\!A}{x\!:\!A \Rightarrow !x\!:\!x\!:\!A}\,,$$

but for no proof polynomial p does **LPGi** derive

$$\Rightarrow p\!:\!(A\to x\!:\!A),$$

since $A\to x\!:\!A$ is not provable in **LP**.

The dual operation to λ-abstraction i.e. *β-conversion*

$$(\lambda x^A.t^B)s^A \quad\longrightarrow_\beta\quad t^B[x^A/s^A]$$

is naturally presented as the following transformation of pure derivations in **LPGi**:

$$\dfrac{\dfrac{\vec{p}{:}\Gamma, x{:}A \Rightarrow t(x){:}B}{\vec{p}{:}\Gamma \Rightarrow \lambda^* x t(x){:}(A \to B)} \qquad \vec{p}{:}\Gamma \Rightarrow s{:}A}{\vec{p}{:}\Gamma \Rightarrow (\lambda^* x t(x) \cdot s){:}B}$$

transforms into

$$\dfrac{\vec{p}{:}\Gamma \Rightarrow s{:}A \qquad \vec{p}{:}\Gamma, s{:}A \Rightarrow t(s){:}B}{\vec{p}{:}\Gamma \Rightarrow t(s){:}B}\,.$$

The rule of η*-conversion*

$$(\lambda x^A.t^B)s^A \quad \longrightarrow_\eta \quad t \qquad \textit{if } x \textit{ is not free in } t$$

is treated in the same way. Finally, α*-conversion* corresponds to an obviously valid rule of renaming bounded variables in **LPGi**-derivations with abstraction.

All other standard λ-term constructors for can also be realized as admissible rules in **LPGi** (cf.[2],[3]).

Since both modal logic **IS4** and all standard λ-term constructors can be emulated by proof polynomials, **LPi** can also emulate modal λ-calculi. As it was shown in [2], [3] **LPGi** naturally realizes the modal λ-calculus for **IS4** ([4], [6], [7], cf. also [5]).

6 Deep realization of modalities by combinatory (λ-) terms

Realization algorithm from Section 4 recovers combinatory terms for every occurrence of modalities in any **IS4** derivation. Natural fragments of **S4** may be be now regarded as implicit description of the corresponding subsystems of **LPi**: the realization algorithm gives the "solution" of such an implicit description in the form of the explicit λ-terms recovered from the derivation in **IS4**.

In particular, the traditional Curry-Howard isomorphism of intuitionistic derivations and simply typed λ-terms may be regarded as a special case of the realization algorithm applied to the derivations in **IS4** containing no modal rules except a single use of the necessitation rule

$$\dfrac{\Gamma \Rightarrow F}{\Box\Gamma \Rightarrow \Box F}$$

as the last rule of the derivation.

The modal λ-calculi realizes the outermost boxes in the necessitation rule above in every **IS4** derivation that ends with this rule. The inner modalities remain undecoded.

The whole **LPi** corresponds to the derivations in the whole **IS4**. The latter is nothing but an implicit specification of some system of λ-terms (proof polynomials) whereas the former is the explicit description of this system.

Therefore **LPi** can recover explicit terms in some modal derivations where the usual modal λ-calculi does not work. Here is a typical example. Consider the sequent $\Box F \Rightarrow \Box\Box F$ derivable in **IS4**. There is no **IS4**-derivation of this sequent that ends with the necessitation rule

$$\frac{F \Rightarrow \Box F}{\Box F \Rightarrow \Box\Box F}\ ,$$

since $F \Rightarrow \Box F$ is not derivable in **IS4**. Hence there is no modal λ-calculus realization of $\Box F \Rightarrow \Box\Box F$ in the sense of [4], [6], [7], i.e. there is no modal λ-term $t(x)$ such that the modal λ-calculus derives $x : F \Rightarrow t(x) : \Box F$. On the other hand, the formula $\Box F \Rightarrow \Box\Box F$ admits a relization in **LPi**, namely $x{:}F \Rightarrow\, !x{:}x{:}F$ where $!x$ is the proof checker polynomial.

7 Standard provability interpretation of LPi

Within this section under Δ_1 and Σ_1 we mean the corresponding classes of arithmetical predicates. If n is a natural number, then $\overline{n}$ will denote a numeral corresponding to n, i.e. a standard arithmetical term $0'''\cdots$ where $'$ is a successor functional symbol and the number of $'$'s equals n. We will use the simplified notation n for a numeral $\overline{n}$ when it is safe.

7.1 Definition. We assume that **HA** contains terms for all primitive recursive functions (cf. [8]), called *primitive recursive terms.* Formulas of the form $f(\vec{x}) = 0$ where $f(\vec{x})$ is a primitive recursive term are *standard primitive recursive formulas. A standard* Σ_1 *formula* is a formula $\exists x\varphi(x,\vec{y})$ where $\varphi(x,\vec{y})$ is a standard primitive recursive formula. An arithmetical formula φ is *provably* Σ_1 if it is provably equivalent in **HA** to a standard Σ_1 formula; φ is *provably* Δ_1 iff both φ and $\neg\varphi$ are provably Σ_1.

7.2 Definition. A *proof predicate* is a provably Δ_1-formula $Prf(x,y)$ such that for every arithmetical sentence φ

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

A proof predicate *Prf(x,y)* is *normal* if the following conditions are fulfilled:

1) (*finiteness of proofs*) For every proof k the set $T(k) = \{l \mid Prf(k,l)\}$ is finite. The function from k to the canonical number of $T(k)$ is computable.

2) (*conjoinability of proofs*) For any natural numbers k and l there is a natural number n such that

$$T(k) \cup T(l) \subseteq T(n).$$

7.3 Comment. Every multi-conclusion normal proof predicate can be transformed into a single conclusion one by changing from

$$\text{“}p \text{ proves } F_1, \ldots, F_n\text{”} \qquad \text{to} \qquad \text{“}(p,i) \text{ proves } F_i,\ i = 1, \ldots n\text{”}.$$

In turn, every single conclusion proof predicate may be regarded as normal multi-conclusion by reading

$$\text{“}p \text{ proves } F_1 \wedge \ldots \wedge F_n\text{”} \qquad \text{as} \qquad \text{“}p \text{ proves each of } F_i,\ i = 1, \ldots n\text{”}.$$

7.4 Proposition. *For every normal proof predicate Prf there are computable functions $\mathbf{m}(x,y)$, $\mathbf{a}(x,y)$, $\mathbf{c}(x)$ such that for all arithmetical formulas φ, ψ and all natural numbers k, n the following formulas are provable in* $\mathbf{HA}$*:*

$$Prf(k, \ulcorner\varphi \rightarrow \psi\urcorner) \wedge Prf(n, \ulcorner\varphi\urcorner) \rightarrow Prf(\mathbf{m}(k,n), \ulcorner\psi\urcorner)$$

$$Prf(k, \ulcorner\varphi\urcorner) \rightarrow Prf(\mathbf{a}(k,n), \ulcorner\varphi\urcorner), \qquad Prf(n, \ulcorner\varphi\urcorner) \rightarrow Prf(\mathbf{a}(k,n), \ulcorner\varphi\urcorner)$$

$$Prf(k, \ulcorner\varphi\urcorner) \rightarrow Prf(\mathbf{c}(k), \ulcorner Prf(k, \ulcorner\varphi\urcorner)\urcorner).$$

Proof. The following function can be taken as $\mathbf{m}$:

> *Given k, n put $\mathbf{m}(k,n) = \mu z$“$Prf(z, \ulcorner\psi\urcorner)$ for all ψ such that there are $\ulcorner\varphi \rightarrow \psi\urcorner \in T(k)$ and $\ulcorner\varphi\urcorner \in T(n)$” .*

Likewise, for $\mathbf{a}$ one could take

> *Given k, n put $\mathbf{a}(k,n) = \mu z$ “$T(k) \cup T(n) \subseteq T(z)$”.*

Finally, $\mathbf{c}$ may be put

> *Given k put $\mathbf{c}(k) = \mu z$“$Prf(z, \ulcorner Prf(k, \ulcorner\varphi\urcorner)\urcorner)$ for all $\ulcorner\varphi\urcorner \in T(k)$”. Such z always exists. Indeed, $Prf(k, \ulcorner\varphi\urcorner)$ are true Δ_1 formulas for every $\ulcorner\varphi\urcorner \in T(k)$, therefore they all are provable in $\mathcal{PA}$. Use conjoinability to find a uniform proof of all of them.*

◀

Note, that the natural arithmetical proof predicate *PROOF(x,y)*

"*x is the code of a derivation in* **HA** *containing a formula with the code y*".

is an example of a normal proof predicate.

7.5 Definition. An arithmetical *interpretation* $*$ of the **LPi**-language has the following parameters:

- a normal proof predicate *Prf* with the functions $\mathbf{m}(x,y)$, $\mathbf{a}(x,y)$, $\mathbf{c}(x)$ as in Proposition 7.4,
- an evaluation of propositional letters by sentences of **HA**, and
- an evaluation of proof variables and proof constants by natural numbers.

Let $*$ commute with boolean connectives,

$$(t\cdot s)^* = \mathbf{m}(t^*, s^*), \quad (t+s)^* = \mathbf{a}(t^*, s^*), \quad (!t)^* = \mathbf{c}(t^*),$$

$$(t\!:\!F)^* = Prf(t^*, \ulcorner F^* \urcorner).$$

Under an interpretation $*$ a proof polynomial t becomes a natural number t^*, an **LPi**-formula F becomes an arithmetical sentence F^*. A formula $(t\!:\!F)^*$ is always provably Δ_1. Note, that **HA** is able to derive any true Δ_1 formula, and thus to derive a negation of any false Δ_1 formula. For a set Γ of **LP**-formulas under Γ^* we mean the set of all F^*'s such that $F \in \Gamma$.

7.6 Definition. Given a constant specification $\mathcal{CS}$, an arithmetical interpretation $*$ is a $\mathcal{CS}$-*interpretation* if all formulas from $\mathcal{CS}^*$ are true (equivalently, are provable in **HA**).

7.7 Proposition. (Arithmetical soundness of **LPi**)

If **LPi** $\vdash F$ *with a constant specification* $\mathcal{CS}$ *then* **HA** $\vdash F^*$ *for any* $\mathcal{CS}$-*interpretation* $*$.

8 Acknowledgements

The research described in this paper was supported in part by ARO under the MURI program "Integrated Approach to Intelligent Systems", grant DAAH04-96-1-0341 and by DARPA under program LPE, project 34145.

References

[1] S. Artemov, "Operational Modal Logic," *Technical Report MSI 95-29*, Cornell University, 1995.
(http://www.cs.cornell.edu/home/artemov/MSI95-29.ps)

[2] S. Artemov, "Explicit provability: the intended semantics for intuitionistic and modal logic," Technical Report CFIS 98-10, Cornell University, September 1998.
(http://www.cs.cornell.edu/home/artemov/CFIS98-10.ps)

[3] S. Artemov, "Unified semantics for modality and λ-terms via proof polynomials," *Technical Report CFIS 99-03*, Cornell University, 1999. To appear in *Logic, Language and Computation, CSLI Publications*, Stanford University, 1999.
(http://www.cs.cornell.edu/home/artemov/CSLI.ps)

[4] G. Bierman and V. de Paiva, "Intuitionistic necessity revisited", *Proceedings of the Logic at Work Conference,* Amsterdam (December 1992), Second revision, June 1996.
(http://theory.doc.ic.ac.uk/tfm/papers.html).

[5] V. A. J. Borghuis, *Coming to Terms with Modal Logic: On the interpretation of modalities in typed λ-calculus*, Ph.D. Thesis, Technische Universiteit Eindhoven, 1994

[6] S. Martini and A. Masini,"A computational interpretation of modal proofs", in Wansing, ed., *Proof Theory of Modal Logics*, (Workshop proceedings), Kluwer, 1994.

[7] F. Pfenning and H.C. Wong, "On a modal lambda-calculus for S4", *Electronic Notes in Computer Science* 1, 1995.

[8] G. Takeuti, *Proof Theory*, North-Holland, 1975

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