Functionality in the Basic Logic of Proofs

Sergei Artëmov*
Steklov Mathematical Institute,
Vavilov str. 42,
117966 Moscow, Russia.
e-mail: art@log.mian.su

Tyko Straßen†
University of Berne, IAM,
Länggassstr. 51,
CH- 3012 Berne.
e-mail: strassen@iam.unibe.ch

Abstract

This report describes the elimination of the injectivity restriction for functional arithmetical interpretations as used in the systems $\mathcal{PF}$ and $\mathcal{PFM}$ in the Basic Logic of Proofs. An appropriate axiom system $\mathcal{PU}$ in a language with operators "x is a proof of y" is defined and proved to be sound and complete with respect to all arithmetical interpretations based on functional proof predicates. Unification plays a major role in the formulation of the new axioms.

1 Introduction

The Basic Logic of Proofs, which has been presented in [1] and [2], describes the behaviour of the arithmetical operator "x is a proof of y" by means of modal logic.

Our modal language contains two sorts of variables, $p_0, p_1, \ldots$ (called *proof variables*) and $S_0, S_1, \ldots$ (called *sentence variables*), the symbol $\rightarrow$ for the classical implication, the truth value $\bot$ (for absurdity), and for each proof variable p_i the unary modal operator $\Box_{p_i}(\cdot)$ (labeled modality). The usual boolean connectives and the truth value $\top$ (for truth) are defined as abbreviations. Small letters $p, q, r, \ldots$ are used for proof variables, capital letters $S, T, \ldots$ for sentence variables, and $A, B, C, \ldots$ for modal formulas.

The semantics of the Basic Logic of Proofs is based on arithmetical interpretations:

1.1 Definition An arithmetical formula $Prf(\cdot,\cdot)$ is called a *functional proof predicate* in **PA** iff

- $Prf(x,y)$ is (provably-in-**PA** equivalent to) a recursive formula in x and y.
- $\mathbf{PA} \vdash \varphi \iff \exists n \in \mathbb{N}: \; Prf(n, \ulcorner \varphi \urcorner)$ for all arithmetical formulas φ.
- If $Prf(n,k_1)$ and $Prf(n,k_2)$ then $k_1 = k_2$.

*Supported by the Swiss Nationalfonds (projects 21-27878.89 and 20-32705.91) during two stays at the University of Berne in January 1992 and January 1993.

†Supported by the Union Bank of Switzerland (**UBS/SBG**) and by the Swiss Nationalfonds (projects 21-27878.89 and 20-32705.91).

1.2 Definition Let $Prf(\cdot,\cdot)$ be a functional proof predicate in $\mathtt{PA}$, and let ϕ be a function that assigns to each proof variable p_i some $n \in \mathbb{N}$ and to each sentence variable S_i a sentence of $\mathtt{PA}$. A *functional arithmetical interpretation* $(\cdot)^*$ is a pair $(Prf(\cdot,\cdot),\phi)$ of such $Prf(\cdot,\cdot)$ and ϕ. The arithmetical interpretation $(A)^*$ (A^* for short) of a modal formula A is the extension of ϕ to all modal formulas by:

- $\top^* := (0=0) \quad \bot^* := (0=1) \quad p_i{}^* := \phi(p_i) \quad S_i{}^* := \phi(S_i)$
- $(A \to B)^* := A^* \to B^*$
- $(\Box_p A)^* := Prf(p^*, \ulcorner A^* \urcorner)$

The Basic Logic of Proofs describes the class of all those formulas which are true for every functional arithmetical interpretation. In [1] and [2] the function ϕ of definition 1.2 was required to be injective, which means that $A^* \equiv B^*$ implies $A \equiv B$ (where $\equiv$ denotes the syntactical identity). This report describes how the requirement of injectivity can be avoided. The Functionality Axiom (letter $\mathcal{F}$ in $\mathcal{PF}$ and $\mathcal{PFM}$) is replaced by a new Unification Axiom (letter $\mathcal{U}$), resulting in the axiom systems $\mathcal{PU}$ and $\mathcal{PUM}$. The theories $\mathcal{P}$ and $\mathcal{PM}$ remain unchanged. In this report only $\mathcal{PU}$ is considered, definitions and proofs for $\mathcal{PUM}$ are analogous.

So the goal of this report is to show that $\mathcal{PU}$ is sound and complete with respect to functional arithmetical interpretations which are not necessarily injective. In the sequel, $(\cdot)^*$ denotes such an arbitrary functional arithmetical interpretation. Functional interpretations as they are used in [1] and [2] for $\mathcal{PF}$ and $\mathcal{PFM}$ will explicitly be called *injective* interpretations.

2 Unification

In this section we make some adaptation of the common unification technique (cf.[3]) to our language of labeled modalities. Those readers who are familiar with substitutions, most general unifiers, etc., can go directly to definition 2.5 .

2.1 Definition A *substitution* θ is a finite set of the form $\{T_1 \leftarrow A_1, \ldots, T_n \leftarrow A_n, q_1 \leftarrow r_1, \ldots, q_m \leftarrow r_m\}$, where the T_i are distinct sentence variables, the q_j are distinct proof variables, and each A_i is a modal formula different from T_i, and each r_j is a proof variable different from q_j. In the following an *expression* is a modal formula or a proof variable. If E is an expression, then we write $E\theta$ for the result of simultaneously replacing each occurrence of T_i in E by A_i and each occurrence of q_j in E by r_j for $i \leq n$ and $j \leq m$, respectively. If θ and θ' are substitutions, then obviously $\theta = \theta'$ iff $E\theta \equiv E\theta'$ for every expression E. The *composition* of substitutions $\theta = \{T_1 \leftarrow A_1, \ldots, q_1 \leftarrow r_1, \ldots\}$ and $\theta' = \{T_1' \leftarrow A_1', \ldots, q_1' \leftarrow r_1', \ldots\}$, denoted by $\theta \circ \theta'$ ($\theta\theta'$ for short) is the substitution obtained by removing elements of the form $T_i' \leftarrow A_i'$ where $T_i' \in \{T_1, \ldots\}$ and $q_j' \leftarrow r_j'$ where $q_j' \in \{q_1, \ldots\}$, and those of the form $T_i \leftarrow T_i$ and $q_j \leftarrow q_j$ from the set $\{T_1 \leftarrow A_1\theta', \ldots, q_1 \leftarrow r_1\theta', \ldots\} \cup \theta'$. Notice that composition is defined in such a way that $(E\theta)\sigma \equiv E(\theta\sigma)$ and $(\mu\theta)\sigma = \mu(\theta\sigma)$ for any expression E and substitutions μ, θ and σ.

2.2 Definition An *equation set* S is a (possibly empty) set $\{E_1 = E'_1, \ldots, E_n = E'_n\}$ of equations of expressions. A substitution θ is called a *unifier* of S iff $E_1\theta \equiv E'_1\theta, \ldots, E_n\theta \equiv E'_n\theta$. Two modal formulas A and B are called *unifiable* iff the equation set $\{A = B\}$ has a unifier. An equation set is in *solved form* if it is of the form $\{T_1 = A_1, \ldots, T_n = A_n, q_1 = r_1, \ldots, q_m = r_m\}$ where $A_1, \ldots, A_n$ are modal formulas, $r_1, \ldots, r_m$ are proof variables, and the $T_1, \ldots, T_n$ and $q_1, \ldots, q_m$ are distinct sentence and proof variables, respectively, which do not occur on the right hand side of any equation. Such an equation set in solved form determines the substitution $\{T_1 \leftarrow A_1, \ldots, T_n \leftarrow A_n, q_1 \leftarrow r_1, \ldots, q_m \leftarrow r_m\}$, which clearly is a unifier of the set. Equation sets are called *equivalent* if they have the same unifiers.

The following algorithm transforms any unifiable equation set S in an equivalent equation set, which is in solved form. For any equation set S, which is not unifiable, the algorithm halts with failure.

2.3 Unification Algorithm Non-deterministically choose an equation from the equation set to which a numbered step applies:

1. $(A_1 \to A_2) = (B_1 \to B_2)$: replace by the equations $A_1 = B_1$ and $A_2 = B_2$.
2. $\Box_p A = \Box_q B$: replace by the equations $p = q$ and $A = B$.
3. $\bot = \bot$ or $p_i = p_i$ or $S_i = S_i$: delete the equation.
4. $A = B$ where A and B are both one of $\bot$, $C \to D$ or $\Box_p C$ but have different principal connectives: halt with failure.
5. $A = S_i$ where A is not a sentence variable: replace by the equation $S_i = A$.
6. $S_i = A$ where A is not the sentence variable S_i and S_i has another occurrence in the set of equations: if S_i appears in A then halt with failure (occurs check) otherwise replace S_i by A in every other equation.
7. $p_i = p_j$ where $i \neq j$ and p_i has another occurrence in the set of equations: replace p_i by p_j in every other equation.

The algorithm terminates when no step can be applied or when failure has been returned.

2.4 Lemma

(i) If the Unification Algorithm terminates by failure, then the equation set S has no unifier,

(ii) If the Unification Algorithm does not terminate by failure, then it terminates with an equivalent equation set S' in solved form, and if θ is the substitution determined by S' then

- θ is a unifier of S,
- θ is idempotent, i.e. $\theta\theta = \theta$,
- θ is the *most general unifier (mgu)* unifier of S, i.e. if μ is another unifier of S then there exists a substitution λ such that $\mu = \theta\lambda$.

Proof (cf. [3])
■

For convenience we consider some deterministic variant of the Unification Algorithm by fixing an order of the equations for this algorithm to choose. Let $\tau_{A,B}$ be the mgu of A, B obtained by this deterministic algorithm starting with the equation set $\{A = B\}$.

2.5 Definition Let A, B, C, D be modal formulas. Then:

$$C = D \pmod{A = B} \quad :\Longleftrightarrow \quad \forall\theta : (A\theta \equiv B\theta \rightarrow C\theta \equiv D\theta)$$

Note that, if A, B are not unifiable then $C = D \pmod{A = B}$ holds for all C and B.

2.6 Lemma Let A, B, C, D be modal formulas such that A and B are unifiable. Then:

$$C = D \pmod{A = B} \quad \Longleftrightarrow \quad C\tau_{AB} \equiv D\tau_{AB}$$

Proof Let $C = D \pmod{A = B}$. Then $C\tau_{A,B} \equiv D\tau_{A,B}$ since $\tau_{A,B}$ unifies A and B. Conversely if $C\tau_{A,B} \equiv D\tau_{A,B}$ and θ is an arbitrary unifier A and B then for some substitution λ, $C\theta \equiv C\tau_{A,B}\lambda \equiv D\tau_{A,B}\lambda \equiv D\theta$.
■

2.7 Corollary The relation $C = D \pmod{A = B}$ is decidable.

■

2.8 Definition If $(\cdot)^* = (Prf(\cdot,\cdot), \phi)$ is an arithmetical interpretation and if θ is a substitution, then their composition $((\cdot)\theta)^*$ is defined to be the arithmetical interpretation $(Prf(\cdot,\cdot), \phi')$ where $\phi'(\cdot) := \phi((\cdot)\theta)$.

Apparently, if $(\cdot)^{**} := ((\cdot)\theta)^*$ then $D^{**} \equiv (D\theta)^*$ for all modal formulas D.

3 The system $\mathcal{PU}$

3.1 Hilbert style system $\mathcal{PU}$

$\mathcal{PU}$ is the modal system with axioms and rules of inference as follows:

(A1) All (boolean) tautologies

(A2) $\Box_p A \longrightarrow A$

(R1) $\dfrac{A \qquad A \rightarrow B}{B}$

(A3) $\Box_p A \wedge \Box_p B \wedge F \longrightarrow G \qquad (F = G \pmod{A = B})$

The axiom (A3) is called *Unification Axiom*. Due to lemma 2.6 it can also be replaced by

(A3$'_1$) $\neg(\Box_p A \wedge \Box_p B) \qquad (A, B$ not unifiable$)$

(A3$'_2$) $\Box_p A \wedge \Box_p B \wedge F \longrightarrow G \qquad (A, B$ unifiable and $F\tau_{A,B} \equiv G\tau_{A,B})$

which gives us an easy procedure to test whether a formula is an instance of (A3). For technical reasons we will often use $(A3'_1)$+$(A3'_2)$ instead of (A3).

The goal of this report is to show that $\mathcal{PU}$ is sound and complete with respect to functional arithmetical interpretations:

3.2 Main Theorem Let A be a modal formula. Then:

$$\mathcal{PU} \vdash A \qquad \Longleftrightarrow \qquad \forall^* : \ \mathtt{PA} \vdash A^*$$

3.3 Example In the following some theorems of $\mathcal{PU}$:

$$\Box_p S_0 \wedge \Box_p S_1 \wedge S_0 \longrightarrow S_1$$
$$\Box_p S_0 \wedge \Box_p S_1 \longrightarrow (\Box_q S_0 \leftrightarrow \Box_q S_1)$$
$$\neg(\Box_p S_0 \wedge \Box_p \Box_q S_0)$$
$$\Box_p S_0 \wedge \Box_p S_1 \wedge \Box_q S_0 \longrightarrow \neg\Box_q(S_1 \vee \bot)$$

3.4 Theorem Let A be a modal formula. Then

$$\mathcal{PU} \vdash A \qquad \Longrightarrow \qquad \mathcal{PF} \vdash A$$

Proof It suffices to show that $\mathcal{PF}$ proves the axioms $(A3'_1)$ and $(A3'_2)$:

(A$3'_1$) If A and B are not unifiable, then they are not syntactically identical, hence $\mathcal{PF} \vdash \neg(\Box_p A \wedge \Box_p B)$.

(A$3'_2$) Let A and B be unifiable.
1^{st} case: $A \equiv B$, thus $\tau_{A,B} = \emptyset$ and so $F \equiv G$.
Trivially $\mathcal{PF} \vdash \Box_p A \wedge \Box_p A \wedge F \to F$.
2^{nd} case: $A \not\equiv B$, then $\mathcal{PF} \vdash \neg(\Box_p A \wedge \Box_p B)$ and so $\mathcal{PF} \vdash \Box_p A \wedge \Box_p B \wedge F \to G$.

Of course, after the soundness result for $\mathcal{PU}$ is established, this theorem is a direct consequence of the fact that the theorems of $\mathcal{PF}$ are all those formulas which are true under *injective* functional interpretations, and the theorems of $\mathcal{PU}$ are the formulas which are true under *all* functional interpretations.
■

4 Soundness

The goal of this section is to investigate the close relationship between arithmetical interpretations and substitutions, and to prove soundness of $\mathcal{PU}$ with respect to arithmetical interpretations.

4.1 Lemma Let A, B be modal formulas and let $(\cdot)^*$ be an arithmetical interpretation. If $A^* \equiv B^*$ then

(a) A and B are unifiable,

(b) $(\cdot)^* = ((\cdot)\tau_{A,B})^*$.

Proof Run the Unification Algorithm 2.3 starting with the equation set $\{A = B\}$ and observe that

- if the equation set S is reduced to S' in one step then

$$\bigwedge_{(E_1=E_2)\in S} {E_1}^* \equiv {E_2}^* \quad \text{iff} \quad \bigwedge_{(E_1=E_2)\in S'} {E_1}^* \equiv {E_2}^*,$$

  (where E_1 and E_2 are expressions)

- if the algorithm terminates by failure with the equation set S then

$$\bigwedge_{(E_1=E_2)\in S} {E_1}^* \equiv {E_2}^* \quad \text{is false.}$$

It follows by induction that

(i) if the algorithm terminates by failure then $A^* \not\equiv B^*$,

(ii) if the algorithm terminates successfully then $A^* \equiv B^*$.

As (i) can not occur it follows that A and B are unifiable and for all sentence variables S_i and proof variables p_j:

$$\begin{array}{rcl} {S_i}^* & \equiv & (S_i\tau_{A,B})^* \\ {p_j}^* & \equiv & (p_j\tau_{A,B})^* \end{array}$$

Finally it follows for arbitrary expressions E by induction that $E^* \equiv (E\tau_{A,B})^*$.
■

This lemma discloses a major difference between "injective functional interpretations and syntactical identity" as used in [1] and [2] where we have

$$\forall^* : \quad A^* \equiv B^* \quad \Longleftrightarrow \quad A, B \text{ are identical}$$

and "functional interpretations and unifiability" as used in this report where we have

$$\exists^* : \quad A^* \equiv B^* \quad \Longleftrightarrow \quad A, B \text{ are unifiable}$$

4.2 Soundness of $\mathcal{PU}$ Let A be a modal formula. Then

$$\mathcal{PU} \vdash A \quad \Longrightarrow \quad \begin{array}{ll} \forall^* : & \mathtt{PA} \vdash A^* \\ \forall^* : & A^* \text{ is true} \end{array} \quad \text{and therefore}$$

Proof It remains to show that $(\mathrm{A3}_1')$ and $(\mathrm{A3}_1')$ are sound, so let $(\cdot)^*$ be an arithmetical interpretation.

$(\mathbf{A3}_1')$ As A and B are not unifiable it follows by lemma 4.1 that $A^* \not\equiv B^*$, so by the functionality of $Prf(\cdot,\cdot)$, $Prf(p^*, \ulcorner A^* \urcorner) \wedge Prf(p^*, \ulcorner B^* \urcorner)$ is false, hence as this formula is recursive, $\neg(Prf(p^*, \ulcorner A^* \urcorner) \wedge Prf(p^*, \ulcorner B^* \urcorner))$ is provable in $\mathtt{PA}$.

$(\mathbf{A3}_2')$ 1st case: $\mathtt{PA} \vdash Prf(p^*, \ulcorner A^* \urcorner) \wedge Prf(p^*, \ulcorner B^* \urcorner)$. By the functionality $A^* \equiv B^*$, thus by lemma 4.1 A and B are unifiable and $F^* \equiv (F\tau_{A,B})^* \equiv (G\tau_{A,B})^* \equiv G^*$. So $\mathtt{PA} \vdash F^* \leftrightarrow G^*$, and finally $\mathtt{PA} \vdash Prf(p^*, \ulcorner A^* \urcorner) \wedge Prf(p^*, \ulcorner B^* \urcorner) \wedge F^* \rightarrow G^*$.
2nd case: $\mathtt{PA} \not\vdash Prf(p^*, \ulcorner A^* \urcorner) \wedge Prf(p^*, \ulcorner B^* \urcorner)$.
As $Prf(\cdot,\cdot)$ is recursive, $\mathtt{PA} \vdash \neg(Prf(p^*, \ulcorner A^* \urcorner) \wedge Prf(p^*, \ulcorner B^* \urcorner))$,
hence $\mathtt{PA} \vdash Prf(p^*, \ulcorner A^* \urcorner) \wedge Prf(p^*, \ulcorner B^* \urcorner) \wedge F^* \rightarrow G^*$.

■

5 Gentzen style system for $\mathcal{PU}$

The completeness proof for $\mathcal{PU}$ has the same outline as the completeness proof for $\mathcal{P}$ (cf. [1]), i.e. we show using a Gentzen style version $\mathcal{PU}_{\mathcal{G}}$ of $\mathcal{PU}$:

$$\begin{array}{ccccc} \forall^*: \ (\bigwedge\Gamma \to \bigvee\Delta)^* \text{ is true} & & \Longleftarrow & & \forall^*: \ \mathbb{PA} \vdash (\bigwedge\Gamma \to \bigvee\Delta)^* \\ \Downarrow & & & & \Uparrow \\ \mathcal{PU}_{\mathcal{G}}^- \vdash \Gamma \supset \Delta & \Longrightarrow & \mathcal{PU}_{\mathcal{G}} \vdash \Gamma \supset \Delta & \Longrightarrow & \mathcal{PU} \vdash \bigwedge\Gamma \to \bigvee\Delta \end{array}$$

In this section $\mathcal{PU}_{\mathcal{G}}$ is defined and it is shown to be sound with respect to $\mathcal{PU}$.

5.1 Definition $\mathcal{PU}_{\mathcal{G}}$ is the sequent calculus with axioms and rules of inference as follows:

- Sequent calculus for classical propositional logic including the cut-rule.
- $\dfrac{A, \Gamma \supset \Delta}{\Box_p A, \Gamma \supset \Delta}^{\text{refl}}$
- $\Box_p A, \Box_p B \supset \qquad (A, B \text{ not unifiable})$
- $\dfrac{(\Box_p A, \Gamma \supset \Delta)\tau_{A,B}}{\Box_p A, \Box_p B, \Gamma \supset \Delta}^{\text{unify}} \qquad (A, B \text{ unifiable})$

$\mathcal{PU}_{\mathcal{G}}^-$ is the system $\mathcal{PU}_{\mathcal{G}}$ without the cut rule.

5.2 Soundness of $\mathcal{PU}_{\mathcal{G}}$ w.r.t. $\mathcal{PU}$ For each sequent $\Gamma \supset \Delta$:

$$\mathcal{PU}_{\mathcal{G}} \vdash \Gamma \supset \Delta \qquad \Longrightarrow \qquad \mathcal{PU} \vdash \bigwedge\Gamma \to \bigvee\Delta$$

Proof Essentially it remains to show that the unify-rule is sound with respect to $\mathcal{PU}$: Let $\mathcal{PU} \vdash (\Box_p A \wedge \bigwedge\Gamma \to \bigvee\Delta)\tau_{A,B}$. As $\tau_{A,B}$ is idempotent (lemma 2.4)

$$(\Box_p A \wedge \Box_p B \wedge \bigwedge\Gamma \to \bigvee\Delta)\tau_{A,B}\tau_{A,B} \quad \text{and} \quad (\Box_p A \wedge \Box_p B \wedge \bigwedge\Gamma \to \bigvee\Delta)\tau_{A,B}$$

are identical, so

$$\Box_p A \wedge \Box_p B \wedge (\Box_p A \wedge \Box_p B \wedge \bigwedge\Gamma \to \bigvee\Delta)\tau_{A,B} \longrightarrow (\Box_p A \wedge \Box_p B \wedge \bigwedge\Gamma \to \bigvee\Delta)$$

can be derived from $(\mathrm{A3}_2')$. As a consequence,

$$\Box_p A \wedge \Box_p B \longrightarrow (\Box_p A \wedge \Box_p B \wedge \bigwedge\Gamma \to \bigvee\Delta)$$

which is equivalent to

$$\Box_p A \wedge \Box_p B \wedge \bigwedge\Gamma \to \bigvee\Delta$$

■

6 Completeness

The goal of this section is to prove the last gap in the diagram shown at the beginning of the previous section, i.e. to prove that if $(\bigwedge \Gamma \to \bigvee \Delta)^*$ is true for every functional interpretation $(\cdot)^*$ then $\Gamma \supset \Delta$ is $\mathcal{PU}_{\mathcal{G}}^-$-provable. The proof is an extension of the completeness proof for $\mathcal{PF}$.

The next definition and the next lemma are cited as a reminder from [1].

6.1 Definition A sequent $\Gamma \supset \Delta$ is called *functionally saturated*, if the following statements hold for all modal formulas A, B and for all proof variables p:

1. If $A \to B \in \Delta$ then $A \in \Gamma$ and $B \in \Delta$,
2. If $A \to B \in \Gamma$ then $B \in \Gamma$ or $A \in \Delta$,
3. If $\Box_p A \in \Gamma$ then $A \in \Gamma$,
4. If $\Box_p A, \Box_p B \in \Gamma$ then $A \equiv B$.

6.2 Main Lemma for $\mathcal{PF}$ Let $\Gamma' \supset \Delta'$ be a functionally saturated sequent which is not $\mathcal{PF}_{\mathcal{G}}^-$-provable. Then there exists an injective functional interpretation $(\cdot)^*$ which makes all formulas in Γ' true and all formulas in Δ' false.

6.3 Saturation Lemma for $\mathcal{PU}_{\mathcal{G}}$ Let $\Gamma \supset \Delta$ be a sequent such that $\mathcal{PU}_{\mathcal{G}}^- \not\vdash \Gamma \supset \Delta$. Then there exists a functionally saturated sequent $\Gamma' \supset \Delta'$ and a substitution σ, such that

(i) $\Gamma\sigma \subset \Gamma'$, $\Delta\sigma \subset \Delta'$,

(ii) $\mathcal{PF}_{\mathcal{G}}^- \not\vdash \Gamma' \supset \Delta'$.

Furthermore $\Gamma' \supset \Delta'$ is effectively computable from $\Gamma \supset \Delta$.

Proof The Saturation Algorithm for $\mathcal{PU}_{\mathcal{G}}$ works similarly to that for $\mathcal{PF}_{\mathcal{G}}$:

- if $S = A \to B \in \Delta$ and $A \notin \Gamma$ or $B \notin \Delta$ then $\Gamma := \Gamma \cup \{A\}$ and $\Delta := \Delta \cup \{B\}$.
- if $S = A \to B \in \Gamma$ and $B \notin \Gamma$ and $A \notin \Delta$ then either $\Gamma := \Gamma \cup \{B\}$ and branch, or $\Delta := \Delta \cup \{A\}$ and branch.
- if $S = \Box_p A \in \Gamma$ and $A \notin \Gamma$ then $\Gamma := \Gamma \cup \{A\}$.
- if $\bot \in \Gamma$ or $\top \in \Delta$ or $\Gamma \cap \Delta \neq \emptyset$ then backtrack.
- if $\Box_p A, \Box_p B \in \Gamma$ and A, B are not unifiable then backtrack.
- if $\Box_p A, \Box_p B \in \Gamma$ and A, B are unifiable then apply $\tau_{A,B}$ to $\Gamma \supset \Delta$.

Properties of the Saturation Algorithm:

- Termination: Each substitution τ_{A_i,B_i} replaces at least one proof or sentence variable, therefore there are only finitely many substitutions possible during the algorithm. After each substitution there are only finitely many other saturation steps possible, too.
- If the algorithm fails, then each branch in the computation contains an axiom, and so one can readily construct a $\mathcal{PU}_{\mathcal{G}}^{-}$-proof of $\Gamma \supset \Delta$.
- If the algorithm succeeds, then the resulting sequent $\Gamma' \supset \Delta'$ is saturated and not $\mathcal{PU}_{\mathcal{G}}^{-}$-provable. Otherwise, assume that $\Gamma' \supset \Delta'$ is $\mathcal{PU}_{\mathcal{G}}^{-}$-provable and thus, as it is saturated, an axiom. Starting with $\Gamma' \supset \Delta'$, and according to the saturation process, construct a $\mathcal{PU}_{\mathcal{G}}^{-}$-proof of $\Gamma \supset \Delta$.
- If $\tau_{A_1,B_1}, \ldots, \tau_{A_k,B_k}$ are the substitutions applied during the saturation algorithm (in that order), then let $\sigma := \tau_{A_1,B_1} \circ \tau_{A_2,B_2} \circ \cdots \circ \tau_{A_k,B_k}$. As each substitution τ_{A_i,B_i} is applied to the whole sequent, clearly $\Gamma\sigma \subset \Gamma'$ and $\Delta\sigma \subset \Delta'$.
- $\mathcal{PF}_{\mathcal{G}}^{-} \not\vdash \Gamma' \supset \Delta'$: Apply the Saturation Algorithm for $\mathcal{PF}_{\mathcal{G}}$ to $\Gamma' \supset \Delta'$. The algorithm succeeds immediately since $\Gamma' \supset \Delta'$ is already functionally saturated: It is not possible that $\Box_p A, \Box_p B \in \Gamma'$ with $A \not\equiv B$ as both cases (A, B unifiable resp. not unifiable) are treated by the Saturation Algorithm for $\mathcal{PU}_{\mathcal{G}}$. Hence $\Gamma' \supset \Delta'$ is not $\mathcal{PF}_{\mathcal{G}}^{-}$-provable.

■

6.4 Main Lemma for $\mathcal{PU}$ Let $\Gamma \supset \Delta$ be a sequent. Then

$$\left[\forall^* : \ (\bigwedge \Gamma \to \bigvee \Delta)^* \ \text{ is true } \right] \quad \Longrightarrow \quad \mathcal{PU}_{\mathcal{G}}^{-} \vdash \Gamma \supset \Delta$$

Proof Let $\mathcal{PU}_{\mathcal{G}}^{-} \not\vdash \Gamma \supset \Delta$. According to the Saturation Lemma for $\mathcal{PU}_{\mathcal{G}}$ there exists a functionally saturated sequent $\Gamma' \supset \Delta'$ which is not $\mathcal{PF}_{\mathcal{G}}^{-}$-provable and a substitution σ such that $\Gamma\sigma \subset \Gamma'$, $\Delta\sigma \subset \Delta'$ (i.e. if $D \in \Gamma$ then $D\sigma \in \Gamma'$, and if $D \in \Delta$ then $D\sigma \in \Delta'$).

From the Main Lemma for $\mathcal{PF}$ it follows that there exists an injective interpretation $(\cdot)^*$ such that:

$$\begin{array}{rcl} D \in \Gamma' & \Longrightarrow & D^* \text{ is true} \\ D \in \Delta' & \Longrightarrow & D^* \text{ is false} \end{array}$$

Let $(\cdot)^{**} := ((\cdot)\sigma)^*$. Notice that $(\cdot)^{**}$ is not necessarily injective. Now if $D \in \Gamma$ then $D^{**} \equiv (D\sigma)^*$ is true since $D\sigma \in \Gamma'$, and if $D \in \Delta$ then $D^{**} \equiv (D\sigma)^*$ is false since $D\sigma \in \Delta'$. Therefore $(\bigwedge \Gamma \to \bigvee \Delta)^{**}$ is false, and again it is clear that even $\mathtt{PA} \vdash \neg(\bigwedge \Gamma \to \bigvee \Delta)^{**}$.

■

The following theorem is another consequence of the Completeness Theorem and the Saturation Lemma for $\mathcal{PU}_{\mathcal{G}}$:

6.5 Theorem Let A be a modal formula, then:

$$\mathcal{PU} \vdash A \quad \Longleftrightarrow \quad \forall\sigma : \ \mathcal{PF} \vdash A\sigma$$

Proof Let $\mathcal{PU} \vdash A$ and let σ be a substitution. Consider $(\cdot)^{**} := ((\cdot)\sigma)^*$ where $(\cdot)^*$ is an arbitrary arithmetical interpretation. From $\mathcal{PU} \vdash A$ it follows that A^{**} thus $(A\sigma)^*$ is true for each interpretation $(\cdot)^*$. As $\mathcal{PU}$ is arithmetically complete it follows $\mathcal{PU} \vdash A\sigma$, and so by theorem 3.4, $\mathcal{PF} \vdash A\sigma$. Conversely, if $\mathcal{PU} \nvdash A$, then $\mathcal{PU}_{\mathcal{G}}^{-} \nvdash \supset A$, and therefore there exists a sequent $\Gamma' \supset \Delta'$ with $\mathcal{PF}_{\mathcal{G}}^{-} \nvdash \Gamma' \supset \Delta'$ and a substitution σ such that $A\sigma \in \Delta'$, by the Saturation Lemma 6.3 for $\mathcal{PU}_{\mathcal{G}}$. By weakening $\mathcal{PF}_{\mathcal{G}}^{-} \nvdash \supset A\sigma$, hence $\mathcal{PF} \nvdash A\sigma$.
■

Notice that $\forall\sigma : \mathcal{PF} \vdash A\sigma$ is not equivalent to $\mathcal{PF} \vdash A$ since $\mathcal{PF}$ is not closed under substitutions:

$$\mathcal{PF} \vdash \neg(\Box_p S_0 \wedge \Box_p S_1)$$

but

$$\mathcal{PF} \nvdash \neg(\Box_p \top \wedge \Box_p \top)$$

So the theorems of $\mathcal{PU}$ are exactly the theorems of $\mathcal{PF}$ for which the substitution property holds.

References

[1] S. Artëmov and T. Straßen, "The Basic Logic of Proofs," in *Computer Science Logic* (E. Börger, G. Jäger, H. Kleine Büning, and M. Richter, eds.), vol. 702 of *Lecture Notes in Computer Science*, pp. 14–28, Proceedings of the 6th Workshop, CSL'92, San Miniato, Italy, October 1992, Springer-Verlag, 1993.

[2] S. Artëmov and T. Straßen, "The Logic of the Gödel Proof Predicate," in *Computational Logic and Proof Theory* (G. Gottlob, A. Leitsch, and D. Mundici, eds.), vol. 713 of *Lecture Notes in Computer Science*, Springer-Verlag, 1993.

[3] J. Lassez, M. Maher, and K. Marriott, "Unification revisited," in *Foundations of Deductive Databases and Logic Programming* (J. Minker, ed.), ch. 15, pp. 587–625, Morgan Kaufmann Publishers, Inc., 1987.