Referential data structures

Sergei Artëmov*
University of Berne, IAM,
Länggassstr. 51,
CH- 3012 Berne.
e-mail:artemov@iam.unibe.ch

Vladimir Krupski†
Department of Mathematics
Moscow State University
Moscow 119899, RUSSIA
email:krupski@sci.math.msu.su

March, 1994

Abstract

We introduce *reference structures* – a basic logical model of a computer memory organization capable to store and utilize information about its addresses. The corresponding labeled modal logics are axiomatized and supplied with the completeness and decidability theorems. A labeled modal formula can be regarded as a description of a certain reference structure; the satisfiability algorithm gives a method of building and optimizing reference structures satisfying a given formula.

1 Introduction

The information about memory blocks addresses are built-in into some data structures, which are arranged in order to provide more fast, more direct access to the memory blocks.

For example, a structure *list* of $A_1, A_2, \ldots$ is a set of records each of which contains a proper data A_i together with the address p_{i+1} of the record containing A_{i+1}:

$$p_1 \text{ stores } (A_1, p_2), \quad p_2 \text{ stores } (A_2, p_3), \quad \ldots.$$

A structure *tree* is organized as a set of records containing a proper data together with the addresses of all successors of a given node. These are simple examples of reference structures. In the example *list* the search algorithm considers a component "p_2" from the cell p_1 as a special short sign saying that "A_2 is stored in p_2". Note that this sign is of a size of the address of p_2, not of A_2.

*Supported by the Swiss Nationalfonds (project 20-32705.91) during the stay at the University of Berne in January of 1994 and by the grant # 93-011-16015 of the Russian Foundation for Fundamental Research

†Partially supported by the grant # 2.1.21 of the Foundation "Universities of Russia" – Fundamental problems of mathematics and mechanics.

We intend to use a language of labeled modalities $\Box_p(\cdot)$, where $\Box_p A$ stands for "block p stores A", to describe reference structures. It requires that a structure stores *sentences*, not numbers, terms, names etc. However, it does not lead to a loss of generality for our purposes. Suppose we have to arrange a certain ordered storage of proper data $A_1, A_2, \ldots$. If A_i for example should originally represent a number N, we assume that A_i is the sentence "this is a number N"; the same treatment may be given to other sorts of proper information: terms, names, substitutions, etc. Thus, we assume that the proper information is represented in a reference structure as a uniform set of "atomic sentences" $A_1, A_2, \ldots$. This sentence-style description makes all stored atomic sentences true. Note that the labeled modal approach presented here is concerned about *referential behavior of the memory organization* and ignores proper information stored; the proper information is hidden in a uniform set of "atomic sentences".

Moreover, we may assume that *all stored sentences are true*; indeed, each sentence stored is either an atomic one or a true statement about the contents of certain cells, about the location of files, etc. This gives us a condition which all the reference structures satisfy:

$$\Box_p A \to A.$$

Another condition appears when we try to make the meaning of $\Box_p A$ precise: it is assumed that p stores not more than one sentence, otherwise we put A to be a conjunction of all sentences stored in p. In fact, if we admit that p can store several sentences, then without extending our basic labeled modal language we loose the control of a cell content; there will be no way to say that p stores exactly the sentences $B_1, \ldots, B_k$, and no others. This imposes another condition of the reference structures:

$$\Box_p A \text{ and } \Box_p B \text{ implies that } A \text{ and } B \text{ coincide.}$$

Below, in a particular labeled modal language with variables over abstract addresses and over data sentences this condition will be expressed explicitly via unification of corresponding formulas.

Reference structures become more realistic model of referential data organizations after we incorporate procedures of reading the contents of memory blocks into them. With a reading procedure it becomes possible to represent direct references to the contents and to the address of a cell by short formulas . This will allow to build optimal reference structures, where a system of aliases is used and no one formula is stored twice. This can give an exponential gain of memory compare to the usual reference structures.

Concerning possible applications of logics of reference structures one may expect them to appear in situations, when we have to arrange a large amount of data within a memory with costly access to it and/or with frequent queries.

2 Reference structures

Definition 2.1 *Let M and X be nonempty sets. Elements of M will be called* memory

cells*; elements of* X - atomic statements. *We treat each* $A \in X$ *as a sentence of some formal language with given semantics, i.e. we suppose that every such* A *is a sentence which is either true or false. The set of reference formulas* $L(M,X)$ *is defined as follows:*

- $\{\top\} \cup X \subseteq L(M,X)$;
- *if* $A, B \in L(M,X)$, *then* $(A \wedge B)$, $\neg A \in L(M,X)$;
- *if* $A \in L(M,X)$ *and* $p \in M$, *then* $\Box_p A \in L(M,X)$.

Here $\wedge$ *and* $\neg$ *denote classical conjunction and negation,* $\top$ *is the boolean constant for truth. Other boolean connectives will be used in* $L(M,X)$ *as suitable abbreviations. Labeled modalities* $\Box_p$ *correspond to cells* $p \in M$. *Let* $\Re = (M, X, R \models)$, *where* $R \subseteq M \times L(M,X)$ *and* $\models$ *is a validity relation on* X. *We define the validity relation* $\Re \models A$ *for* $A \in L(M,X)$ *as follows:*

- $\Re \models x \quad \Leftrightarrow \quad \models x \qquad$ *for* $x \in X$;
- $\Re \models \top$;
- $\Re \models (A \wedge B) \quad \Leftrightarrow \quad \Re \models A$ *and* $\Re \models B$;
- $\Re \models \neg A \quad \Leftrightarrow \quad \Re \not\models A$;
- $\Re \models (\Box_p A) \quad \Leftrightarrow \quad (p, A) \in R$.

$\Re = (M, X, R, \models)$ *is called* reference structure *(*r.s.*), if the relation* R *is functional and the condition*

$$(p, A) \in R \quad \Rightarrow \quad \Re \models A$$

holds for each $p \in M$ *and* $A \in L(M,X)$.

Example 2.2 *A* list *of* $A_1, A_2, ..., A_n$ *can be regarded as r.s.* $(M, X, R, \models)$, *where* $M = \{p_1, p_2, ..., p_n, q\}$ *is the set of cells,* $X = \{A_1, A_2, ..., A_n\}$ *is the set of atomic sentences,* $\models A_i$ *for all* i *and*

$$R = \{(p_1, \varphi_1), ..., (p_n, \varphi_n), (q, \top)\}$$

for $\varphi_1 = A_1 \wedge \Box_{p_2}\varphi_2$, $\varphi_2 = A_2 \wedge \Box_{p_3}\varphi_3, \ldots,$ $\varphi_n = A_n \wedge \Box_q \top$; *here* $\top$ *works as a marker of the end node. The entire list can be represented by the formula* $\Box_{p_1}\varphi_1$.

It does not mean, however that we intend to store the entire list in the cell p_1. After introducing reading procedures below we will use a short formula $\widetilde{p_2}$ of the size of p_2 instead of $\Box_{p_2}\varphi_2$; the meaning of $\widetilde{p_2}$ as "p_2 stores φ_2" is in fact built in the search algorithm. Thus a correspondence between a reference structures and real data bases widely uses the default strategy of the search algorithms.

Definition 2.3 *A* reading procedure $\theta : M \longmapsto L(M,X)$ *for reference structure* $\Re = (M, X, R, \models)$ *is a total functional extension of storage relation* R.

If a cell $p \in M$ contains a sentence from $L(M,X)$, then the reading procedure returns the contents of p; otherwise is returns some sentence of $L(M,X)$, which may be regarded as an error message depending on p.

Definition 2.4 *A* reference structure with reading (r.s.r.) *is a pair where $\Re$ is r.s. and θ is a reading procedure on $\Re$. We omit θ in $(\Re,\theta)$, if the storage relation R in $\Re$ is total.*

There is no natural way to represent functions in a propositional language; it requires some technical maneuvers to incorporate a reading procedure into $L(M,X)$.

Definition 2.5 The language $\widehat{L}(M,X)$ *extends $L(M,X)$ by allowing sentence constant $\widehat{p}$ for each cell $p \in M$.*

Every r.s.r. determines the translation from $\widehat{L}(M,X)$ into $L(M,X)$: for $A \in \widehat{L}(M,X)$ its translation $A\theta$ is the result of substitution of the variables $\widehat{p}$ by the values $\theta(p)$ everywhere in A. We define

$$(\Re,\theta) \models A \quad \Leftrightarrow \quad \Re \models A\theta.$$

So $\widehat{p}$ denotes the contents of p. If a cell p is empty, then $\Box_p\widehat{q}$ is false for any q. Thus $\Box_p\widehat{p}$ is true iff the cell p is not empty. We define a formula $\widetilde{p}$ as $\Box_p\widehat{p}$ and will use it as a natural pointer. Note that the length of $\Box_p\widehat{p}$ can be easily made of the order of the size of p.

Definition 2.6 *The formula $A \in \widehat{L}(M,X)$ is an alias for the formula $B \in L(M,X)$ with respect to $(\Re,\theta)$ if $A\theta = B$.*

Definition 2.7 *Let r.s. $\Re = (M,X,R,\models)$ with reading procedure θ be fixed. The relation $R_1 \subseteq M \times \widehat{L}(M,X)$ is called a* reduced storage relation *of $(\Re,\theta)$, if*

1. *R_1 is functional and has the same domain as R.*
2. *$R = \{(p, A\theta) \mid (p,A) \in R_1\}$.*

If R_1 is a reduced storage relation for $(\Re,\theta)$, then the quadruple $\Re_1 = (M,X,R_1,\models)$ is a reduction *of $(\Re,\theta)$. The validity relation $\Re_1 \models A$ for any $A \in \widehat{L}(M,X)$ is naturally defined by induction on A (we list only item additional to definition 2.1 of the reference structure):*

- $\Re_1 \models \widehat{p} \quad \Leftrightarrow \quad (\Re,\theta) \models \widehat{p}$

Clearly for any $A \in \widehat{L}(M,X)$

$$\Re_1 \models A \quad \Leftrightarrow \quad (\Re,\theta) \models A.$$

Example 2.8 *Now instead of the reference structure "list" from example 2.2 one may consider its reduced storage relation*

$$R_1 = \{(p_1, A_1 \wedge \widetilde{p_2}), (p_2, A_2 \wedge \widetilde{p_3}), \ldots, (p_n, A_n \wedge \widetilde{q}), (q, \top)\}.$$

The corresponding reduction of the r.s. from 2.2 can now be represented by the formula

$$\Box_{p_1}(A_1 \wedge \widetilde{p_2}) \wedge \ldots \wedge \Box_{p_n}(A_n \wedge \widetilde{q}) \wedge \Box_q \top.$$

The language $\widehat{L}(M,X)$ may give the exponential gain in the length of descriptions compare to $L(M,X)$.

Example 2.9 *The formula*

$$\varphi = \Box_{p_0} A \wedge \Box_{p_1}(\widehat{p}_0 \wedge \widehat{p}_0) \wedge \ldots \wedge \Box_{p_n}(\widehat{p}_{n-1} \wedge \widehat{p}_{n-1})$$

is valid iff the corresponding fragment of r.s. looks like this:

$$\begin{array}{lll} p_0 & \textit{stores} & A, \\ p_1 & \textit{stores} & A \wedge A, \\ p_2 & \textit{stores} & (A \wedge A) \wedge (A \wedge A), \\ \vdots & \vdots & \vdots \\ p_n & \textit{stores} & (\ldots(A \wedge A) \wedge \ldots) \wedge (\ldots \wedge (A \wedge A)\ldots). \end{array}$$

Any $L(M,X)$-formula describing the same storage relation is exponentially longer than φ.

3 Logic of reference structures

We introduce now a logical calculus $\widehat{\mathcal{PU}}$ which is a conservative extension of $\mathcal{PU}$ from [1]. Reference structures will be natural models of $\widehat{\mathcal{PU}}$; moreover $\widehat{\mathcal{PU}}$ will be proven to be complete with respect to the class of all (finite) reference structures with reading procedures.

Consider the abstract labeled modal language $\widehat{L}$ which contains memory cell variables $b_0, b_1, b_2, \ldots$, two sorts of sentence variables $S_0, S_1, S_2, \ldots$ and $\widehat{b}_0, \widehat{b}_1, \widehat{b}_2 \ldots$, constant $\top$, and is closed under boolean connectives and unary operators $\Box_{b_i}, i = 0, 1, 2, \ldots$.

Definition 3.1 *An* interpretation *of $\widehat{L}$ is r.s. $(M, X, R, \models)$ with reading procedure θ, and a mapping $*$ of $b_0, b_1, b_2, \ldots$ into M and $\widehat{L}$ into $L(M,X)$ such that $*$ commutes with the boolean connectives, $(\widehat{b}_i)^* = \theta(b_i^*)$ and $(\Box_p A)^*$ is $\Box_{p^*} A^*$. We say that an interpretation $*$ of $\widehat{L}$ in r.s.r. $(\Re, \theta)$ is a* model of $\Gamma \in \widehat{L}$ *if A^* is valid in $\Re$ for each $A \in \Gamma$.*

The language $\widehat{L}$ may now be regarded as a programming language for designing reference structures and reductions of reference structures. A program here is a formula $A \in \widehat{L}$ describing the properties of a pair $(\Re, \theta)$. Satisfiability of A means the existence of a desired r.s.r. $(\Re, \theta)$. The satisfiability algorithm for the language $\widehat{L}$ naturally arises from the completeness proof of the calculus $\widehat{\mathcal{PU}}$ (below).

We assume a reader to be familiar with substitutions and common unification technique (cf.[2]). For a convenience we consider some deterministic variant of the Unification Algorithm by fixing an order of the equations for this algorithm to choose. Usually the Unification Algorithm deals with systems of "unconditional" equalities of the form $A\sigma = B\sigma$. We are interested in the "conditional" equalities of the form $b_i\sigma = b_j\sigma \Longrightarrow \widehat{b}_i\sigma = \widehat{b}_j\sigma$ too. The suitable modification **U** of the Unification Algorithm is as follows: using the standard Unification Algorithm solve the unconditional part of the system, then check the conditions; if

the conditions fail, then we are done. If the conditions are fulfilled, add succedent equations to the unconditional part and solve the system again, etc.. The process terminates when the checking procedure fails to add new equalities or the Unification Algorithm fails to solve the unconditional part of the system. The standard argument proves that this modification gives the most general solution (m.g.u.) of the system with "conditional" equations.

Definition 3.2 *The* standard *m.g.u.* $\sigma_{A,B,p}$ *of the set of equations*

$$\begin{array}{c} \widehat{p}\sigma = A\sigma = B\sigma, \\ b_i\sigma = b_j\sigma \Longrightarrow \widehat{b}_i\sigma = \widehat{b}_j\sigma. \end{array} \tag{1}$$

is the m.g.u. obtained by $\mathbf{U}$*. Note that* σ *acts on the variables of all sorts,* $b_i\sigma$ *is a cell variable,* $\widehat{b}_i\sigma$ *and* $S_i\sigma$ *are formulas from* $\widehat{L}$*.*

Lemma 3.3 *(Cf.[2])* $Dom(\sigma_{A,B,p}) \cap Val(\sigma_{A,B,p}) = \emptyset$ *and* $\sigma_{A,B,p}$ *is idempotent, i.e.* $\sigma_{A,B,p} \circ \sigma_{A,B,p} = \sigma_{A,B,p}$*, and for every solution* σ *of (1) there exists a substitution* λ *s.t.* $\sigma = \sigma_{A,B,p} \circ \lambda$ *.*

Definition 3.4 *We define* $C = D \ (mod\ \widehat{p} = A = B)$ *to be an abbreviation for*

"$C\sigma = D\sigma$ *for every solution* σ *of (1)".*

Apparently, if the system (1) has no solution, then $C = D \pmod{\widehat{p} = A = B}$ holds for all C and D. If the system (1) has a solution then

$$C = D \pmod{\widehat{p} = A = B} \iff C\sigma_{A,B,p} = D\sigma_{A,B,p}.$$

So, the relation $C = D \pmod{\widehat{p} = A = B}$ is decidable.

Definition 3.5 *Axioms of* $\widehat{\mathcal{PU}}$*:*

(A1) The classical propositional calculus.

(A2) $\Box_p A \to A$*.*

(A3) $\Box_p A \wedge \Box_p B \to (C \to D)$ *if* $C = D \ (mod\ \widehat{p} = A = B)$*.*

Axiom (A3) is similar to the unification axiom from [1] and the functionality axiom from [3].

Lemma 3.6 *If* $A \in \widehat{L}$ *and* $\vdash A$*, then* A^* *is valid for all interpretations* $*$*.*

Proof. A straightforward induction on the proof of A. ■

Lemma 3.7 *The following is provable in* $\widehat{\mathcal{PU}}$*:*

$(A3^{'})$ $\neg(\Box_{p_1}A_1 \wedge \ldots \Box_{p_n}A_n)$ *if a substitution* σ *with the property*

$$\begin{array}{c} \widehat{p}_k\sigma = A_k\sigma, \quad k = 1, \ldots, n; \\ b_i\sigma = b_j\sigma \Longrightarrow \widehat{b}_i\sigma = \widehat{b}_j\sigma \end{array} \tag{2}$$

does not exist.

$(A4^{'})$ $\Box_{p_1}A_1 \wedge \ldots \Box_{p_n}A_n \rightarrow (B \leftrightarrow C)$ *if* $B\sigma = C\sigma$ *for the most general unifier* σ *satisfying the condition (2) and obtained as a result of the standard unification algorithm* $\boldsymbol{U}$.

Proof. Induction on n. If $n = 1$ then $(A3^{'})$,$(A4^{'})$ are equivalent to the variants of $(A3)$ where $A = B$. Let $n > 1$, $\varphi = \bigwedge_{i=1}^{i=n-1} \Box_{pi}A$.

Case $(A4^{'})$. Note that the standard m.g.u. is an idempotent. By induction

$$\begin{array}{l} \vdash \varphi \rightarrow (B \leftrightarrow B\sigma_1), \\ \vdash \varphi \rightarrow (C \leftrightarrow C\sigma_1), \\ \vdash \varphi \rightarrow (\Box_{p_n}\widehat{p}_n \leftrightarrow (\Box_{p_n}\widehat{p}_n)\sigma_1), \\ \vdash \varphi \rightarrow (\Box_{p_n}A_n \leftrightarrow (\Box_{p_n}A_n)\sigma_1), \end{array} \tag{3}$$

where σ_1 is the standard most general unifier for the system:

$$\begin{array}{c} \widehat{p}_k\sigma_1 = A_k\sigma_1, \quad k = 1, \ldots, n-1; \\ b_i\sigma_1 = b_j\sigma_1 \Longrightarrow \widehat{b}_i\sigma_1 = \widehat{b}_j\sigma_1. \end{array} \tag{4}$$

Consider the system

$$\begin{array}{c} \widehat{p}_n\sigma_2 = \widehat{p}_n\sigma_1\sigma_2, \\ \widehat{p}_n\sigma_2 = A_n\sigma_1\sigma_2, \\ b_i\sigma_1 = b_j\sigma_1 \Longrightarrow \widehat{b}_i\sigma_1 = \widehat{b}_j\sigma_1 \end{array} \tag{5}$$

and the standard most general unifier σ_2 for it. System (5) is a variant of (1), the consistency of (4) and (5) follows from the consistency of (2). Then $\sigma^{'} = \sigma_1\sigma_2$ is a solution of (2) and $B\sigma_1\sigma_2 = C\sigma_1\sigma_2$ so

$$(\Box_{p_n}\widehat{p}_n)\sigma_1 \wedge (\Box_{p_n}A_n)\sigma_1 \rightarrow (B\sigma_1 \leftrightarrow C\sigma_1) \tag{6}$$

is a variant of $(A3)$. The formula

$$\Box_{p_n}A_n \rightarrow \Box_{p_n}\widehat{p}_n \tag{7}$$

is also provable (from $(A1), (A3)$). So $(A4^{'})$ is derivable from (3),(6),(7) in classical propositional calculus.

Case $(A3^{'})$. If the system (4) has no solution then, by induction, $\neg\varphi$ is provable and so is $(A3^{'})$. Else, following the previous case we prove that (2) is inconsistent iff (5) is. So

$$\neg((\Box_{p_n}\widehat{p}_n)\sigma_1 \wedge (\Box_{p_n}A_n)\sigma_1)$$

is equivalent to a variant of $(A3)$. Similarly

$$\vdash \Box_{p_1}A_1 \wedge \ldots \wedge \Box_{p_n}A_n \rightarrow (\Box_{p_n}\widehat{p}_n)\sigma_1 \wedge (\Box_{p_n}A_n)\sigma_1,$$

that proves $(A4^{'})$. ■

4 The Completeness theorem

Theorem 4.1 *For any formula* $A \in \widehat{L}$

$$\widehat{\mathcal{PU}} \vdash A \quad \textit{iff} \quad \begin{array}{l} A \textit{ is valid under all interpretations} \\ \textit{in reference structures with reading.} \end{array}$$

Proof. The correctness part of the theorem follows from Lemma 3.6. The completeness proof is based on the Gentzen style formulation of $\widehat{\mathcal{PU}}$ and saturation procedure.

In the following a *sequent* is a formal expression of the form $\Gamma \supset \Delta$, where Γ and Δ are finite subsets of $\widehat{L}$.

Definition 4.2 $\widehat{\mathcal{PU}}_G$ *is the sequent calculus with axioms and rules of inference as follows:*

Axioms:

- $\Gamma \supset \Delta$ *such that* $\Gamma \cap \Delta \neq \emptyset$ *or* $\top \in \Delta$.
- $\Xi \supset$, *where* $\Xi = \{\Box_{pi} A_i \mid i = 1, \ldots, n\}$ *and a substitution* σ *with the property (2) for* Ξ *does not exist.*

Rules:

- *Classical rules for* $\wedge, \neg$ *and structural rules together with the cut-rule.*
- $$\frac{A, \Gamma \supset \Delta}{\Box_p A, \Gamma \supset \Delta}$$
- $$\frac{\Xi, B\sigma, \Gamma \supset \Delta}{\Xi, B, \Gamma \supset \Delta} \ , \quad \frac{\Xi, \Gamma \supset B\sigma, \Delta}{\Xi, \Gamma \supset B, \Delta} \ ,$$ *where* $\Xi = \{\Box_{pi} A_i \mid i = 1, \ldots, n\}$ *and* σ *is the most general unifier satisfying the condition (2) for* Ξ *and obtained as a result of the standard unification algorithm* $\boldsymbol{U}$. $\widehat{\mathcal{PU}_G}^-$ *is the system* $\widehat{\mathcal{PU}_G}$ *without the cut rule.*

Lemma 4.3 *(Soundness of* $\widehat{\mathcal{PU}}_G$ *w.r.t.* $\widehat{\mathcal{PU}}$*) If* $\widehat{\mathcal{PU}}_G \vdash \Gamma \supset \Delta$ *then* $\Gamma \vdash \bigvee \Delta$. [1]

Proof. Straightforward induction on the complexity of the proof of $\Gamma \supset \Delta$ in $\widehat{\mathcal{PU}}_G$. ■

Definition 4.4 Saturation process *is the nondeterministic procedure constructing a labeled* saturation tree *as follows:*

[1] $\bigvee \Delta$ denotes $A_1 \vee \ldots \vee A_n$ for $\Delta = A_1, ..., A_n$

Given the sequent $\Gamma_0 \supset \Delta_0$ label the root with it and try repeatedly to apply the saturation rules while they permit to add to the tree some node with the label different from the label of its parent. The rules can be applied to an arbitrary leaf of the current part of the tree if its label $\Gamma \supset \Delta$ is not an axiom of $\widehat{\mathcal{PU}_G}$. In the formulations of the rules we suppose that such a leaf (a current node) is already chosen.

Saturation rules:

Rule 1. If $A \wedge B \in \Gamma$, then $\Gamma_1 := \Gamma \cup \{A, B\}$ and $\Delta_1 := \Delta$. Add to the tree a son of the current node labeled with the sequent $\Gamma_1 \supset \Delta_1$.

Rule 2. If $A \wedge B \in \Delta$, then $\Gamma_1 := \Gamma, \Delta_1 := \Delta \cup \{A\}$ and $\Gamma_2 := \Gamma, \Delta_2 := \Delta \cup \{B\}$. Add to the tree two sons of the current node labeled with the sequents $\Gamma_1 \supset \Delta_1$ and $\Gamma_2 \supset \Delta_2$.

Rule 3. If $\neg A \in \Gamma$ ($\neg A \in \Delta$), then $\Gamma_1 := \Gamma, \Delta_1 := \Delta \cup \{A\}$ ($\Gamma_1 := \Gamma \cup \{A\}$ and $\Delta_1 := \Delta$). Add to the tree a son of the current node labeled with the sequent $\Gamma_1 \supset \Delta_1$.

Rule 4. If $\Box_p A \in \Gamma$, then $\Gamma_1 := \Gamma \cup \{A\}$ and $\Delta_1 := \Delta$. Add to the tree a son of the current node labeled with the sequent $\Gamma_1 \supset \Delta_1$.

Rule 5. Call the unification algorithm **U** to get the most general solution σ of the system (2) where $\Box_{pi} A_i, i = 1, ..., n$ is the list of all formulas of the form $\Box_p A$ from Γ. Let $\Gamma_1 := \Gamma \cup \{B\sigma | B \in \Gamma\}$ and $\Delta_1 := \Delta \cup \{B\sigma | B \in \Delta\}$. Add to the tree a son of the current node labeled with the sequent $\Gamma_1 \supset \Delta_1$.

Remark on termination. Rules 1-4 do not change the subformulas of the sequent so they can not be applied infinitely many times. In rule 5 the algorithm **U** calls the standard Unification Algorithm. The last one being applied to the 'unconditional' part of the system (2) returns a substitution σ with the property $\sigma^2 = \sigma$ that acts only on the variables from $\Gamma_0 \cup \Delta_0$. Let V be the set of all such variables and $V_0 = \{v \in V | v\sigma = v\}$. The set V_0 depends on the order in which the equations of the system (2) are given to the Unification Algorithm. If we add to the end of the system (2) some new equations without violation of the consistency of the system then the Unification Algorithm returns a substitution with corresponding set $V_0' \subseteq V_0$. So it is possible (and we suppose it is done) to implement the calls to the unification algorithm **U** in such a way that along every path in the saturation tree the sequence of corresponding sets V_0 decreases monotonically. That provides a bound on the set of subformulas. Thus a saturation tree for every sequent is finite (but not necessarily unique). Therefore the saturation process always terminates and computes some saturation tree of a given sequent. We say that the saturation process *succeeds* if it produces a saturation tree with all leafs labeled with axioms; otherwise it *fails*.

Lemma 4.5 *If the saturation process on a given sequent succeeds, then the sequent is provable in* $\widehat{\mathcal{PU}_G}^-$.

Proof. The saturation tree with all leafs labeled by axioms is in fact the tree-like derivation in $\widetilde{\mathcal{PU}_G}^-$ of the sequent labeling the root. ■

Lemma 4.6 *If the saturation process fails on a sequent $\Gamma_0 \supset \Delta_0$, then the interpretation $*$ with the property*

$$\Re \models (\bigwedge \Gamma_0)^*, \qquad \Re \not\models (\bigvee \Delta_0)^* \tag{8}$$

can be constructed from the saturation tree.

Proof. The saturation process provides the following *saturation properties* of the sequent $\Gamma \supset \Delta$ which is not an axiom and labels some leaf of the saturation tree:

- $\Gamma_0 \subseteq \Gamma, \Delta_0 \subseteq \Delta; \Gamma \cap \Delta = \emptyset; \ \top \notin \Delta;$
- if $(A \wedge B) \in \Gamma$, then $A \in \Gamma$ and $B \in \Gamma$;
- if $(A \wedge B) \in \Delta$, then $A \in \Delta$ or $B \in \Delta$;
- if $\neg A \in \Gamma$, then $A \in \Delta$; if $\neg A \in \Delta$, then $A \in \Gamma$
- if $\Box_p A \in \Gamma$, then $A \in \Gamma$;
- if $\Box_{pi} A_i, \ i = 1, ..., n$ is a list of all formulas of the form $\Box_p A$ from Γ, then there exist a substitution σ with the property (2);
- if σ is the standard m.g.u. of (2) and $A \in \Gamma$ ($A \in \Delta$), then $A\sigma \in \Gamma$ ($A\sigma \in \Delta$).

Let V_0 be the set of all variables from $\Gamma_0 \cup \Delta_0$ with the property $v\sigma = v$. For each variable $v \in V_0$ we choose a new constant of the corresponding type (cell or sentence) denoted by x_v. Let

$$\begin{array}{l} M = \{x_v \mid v \in V_0, \ v \text{ is a cell variable}\} \\ X = \{x_v \mid v \in V_0, \ v \text{ is a sentence variable}\} \\ v\sigma_1 = x_v, \ (v \in V_0), \\ R = \{(x_p, A\sigma_1) \mid p \in V_0, \ (\Box_p A) \in \Gamma\sigma.\} \end{array}$$

We also assume that $v\sigma_1 \in X$ for $v \notin V_0$. The validity relation on X is defined as follows:

$$\models x_v \quad \text{iff} \quad v \in \Gamma\sigma.$$

We claim that

$$\begin{array}{lll} A \in \Gamma & \Longrightarrow & \Re \models A^*, \\ A \in \Delta & \Longrightarrow & \Re \not\models A^*. \end{array}$$

Indeed, note that $\Gamma\sigma \subseteq \Gamma, \Delta\sigma \subseteq \Delta$ and it is sufficient to prove the claim for $A \in \Gamma\sigma \cup \Delta\sigma$ because $A\sigma$ has the same interpretation as A. Induction on the complexity of $A \in \Gamma\sigma \cup \Delta\sigma$:

- Case $A = \top$ follows from the saturation properties.

- Case $A = S_i$ follows from the definition of $\models$.
- Case $A = \widehat{p}$. Then $\widehat{p} \in V_0$ and

$$\widehat{p}^* = \theta(p^*) = \theta(x_{p1}) = \widehat{(p_1)}\sigma\sigma_1$$

  where $p_1 = p\sigma \in V_0$. So $p\sigma = p_1\sigma$ and $\widehat{p}\sigma = \widehat{p_1}\sigma = \widehat{p}$. We have

$$\widehat{p}^* = \widehat{p}\sigma_1 = x_{\widehat{p}}.$$

  By definition of validity relation $\Re \models \widehat{p}^*$ iff $\widehat{p} \in \Gamma\sigma$.
- Case $A = (B \wedge C)$ follows from the saturation properties and the induction hypothesis.
- Case $A = \Box_p B$ follows from the definition of R.

We are now to prove that $\Re := (M, X, R, \models)$ is a r.s., $\theta(x_p) = \widehat{p}\sigma\sigma_1$, $(x_p \in M)$ is a reading procedure and the condition (8) holds for $* = \sigma\sigma_1$.

$\Re$ is a r.s.. The functionality of R follows from saturation properties.

$$\begin{array}{rcll} (x_p, A\sigma_1) \in R & \Longrightarrow & \Box_p A \in \Gamma\sigma & \\ & \Longrightarrow & A \in \Gamma\sigma \subseteq \Gamma & \text{(saturation property)} \\ & \Longrightarrow & \Re \models A^* & \text{(by Claim above)} \\ & \Longrightarrow & \Re \models A\sigma_1 & (A\sigma_1 = A^*). \end{array}$$

θ is a reading procedure:

$$\begin{array}{rcll} (x_p, A\sigma_1) \in R & \Longrightarrow & \Box_p A \in \Gamma\sigma \subseteq \Gamma & \\ & \Longrightarrow & \widehat{p}\sigma = A\sigma = A & (\sigma^2 = \sigma) \\ & \Longrightarrow & \theta(x_p) = \widehat{p}\sigma\sigma_1 = A\sigma_1. & \end{array}$$

The conditions (8) follow from the claim and saturation property $\Gamma_0 \subseteq \Gamma$, $\Delta_0 \subseteq \Delta$. ■

The completeness part of Theorem 4.1 follows now from Lemmas 4.3, 4.5, 4.6. ■

Corollary 4.7 *For finite sets Γ, Δ of $\widehat{L}$-formulas, and any $\widehat{L}$-formula A*

$$\begin{array}{rcl} \Gamma \vdash A & \Longleftrightarrow & \widehat{\mathcal{PU}}_G \vdash \Gamma \supset A, \\ \widehat{\mathcal{PU}}_G \vdash \Gamma \supset \Delta & \Longleftrightarrow & \widehat{\mathcal{PU}}_G{}^{-} \vdash \Gamma \supset \Delta. \end{array}$$

Corollary 4.8 *$\widehat{\mathcal{PU}}$ is decidable.*

Proof. By Corollary 4.7 the decidability of $\widehat{\mathcal{PU}}$ follows from the decidability of $\widehat{\mathcal{PU}}_G$. The decision algorithm for $\widehat{\mathcal{PU}}_G$ is given by the saturation process. The sequent is provable in $\widehat{\mathcal{PF}}_G$ iff the Saturation succeeds. ■

5 Reference structures building and optimization.

The language $\widehat{L}$ can now be considered as a programming language for designing reference structures with reading procedures and their reductions. A program here is a formula $\varphi \in \widehat{L}$ describing the properties of some (reduction of) r.s.r. $(\Re, \theta)$. The satisfiability algorithm extracted from the proof of the Completeness theorem checks whether φ is satisfiable and constructs the finite model for φ, which is the desired r.s.r. $(\Re, \theta)$.

The first stage is saturation. We reduce the problem of constructing a model for φ to the problem of constructing a countermodel for the sequent $\varphi \supset$. The saturation algorithm checks whether this sequent is provable and transforms it into the sequent $\Gamma \supset \Delta$ with saturation properties. If saturation succeeds, then $\widehat{\mathcal{PU}} \vdash \neg\varphi$, and thus there is no r.s.r. satisfying the condition φ. If saturation fails, we go to the second stage, which is the construction of r.s.r. $(\Re, \theta)$ and interpretation $*$ satisfying conditions (8); it is described in the proof of Lemma 4.6. The interpretation is a model of φ because $\varphi \in \Gamma$. Note that according to Lemma 4.6 the resulting interpretation $*$ maps sentence variables into atomic sentences of r.s.r. $(\Re, \theta)$. It means, that in programming of reference structures by means of $\widehat{L}$-formulas sentence variables stand for atomic sentences of reference structures. The default third stage is a transition from the reference structure $(\Re, \theta)$ to a real memory organization; in particular, we have to restore a proper data, hidden under uniform set of atomic sentences of the reference structure. We may assume that if $(p, A) \in R$ (where p is a memory cell, A is an $L(M, X)$-formula, R is the storage relation of $(\Re, \theta)$), and A is $x \wedge A'$, where $x \in X$ is an atomic sentence of $(\Re, \theta)$, then we restore in the cell p the proper data corresponding to the atomic sentence x. The occurrences of atomic sentences from X inside the scope of labeled modalities in A play the role of *names* of the relevant pieces of proper data, which are much shorter, then the proper data itself; these names are not supposed to be replaced. So, we finally get a memory organization with cells containing proper information together with (much shorter) referential components, describing the structure of storage relation.

There are many natural ways to use the logic $\widehat{\mathcal{PU}}$ and the satisfiability algorithm to construct the desired reductions of r.s.r.'s. For example, suppose the formula φ describing a reference structure has the form

$$\Box_{p_1} A_1 \wedge \Box_{p_2} A_2 \wedge \ldots \Box_{p_n} A_n \wedge \varphi'.$$

Then the interpretation $*$ from Lemma 4.6 determines not only r.s.r $(\Re, \theta)$, but also a reduction of $(\Re, \theta)$ with the reduced storage relation R_1 such that $(p_i^*, A_i^*) \in R_1$ for all $i = 1, \ldots, n$.

In order to construct reductions of r.s.r.'s which use only one cell instead of many containing the same record, i.e. to construct r.s. with functional relation R^{-1} or, moreover, with invertible reading procedure θ, we introduce the logics $\widehat{\mathcal{PU}}_1$ and $\widehat{\mathcal{PU}}_{1-1}$. The logic $\widehat{\mathcal{PU}}_1$ is $\widehat{\mathcal{PU}}$ + (A5) where (A5) is the following axiom scheme:

(A5) $\qquad \Box_p A \wedge \Box_{p'} A \ \rightarrow (B \leftrightarrow B[p'/p])$.

$\widehat{\mathcal{PU}}_{1-1}$ is the modification of $\widehat{\mathcal{PU}}$ where the "conditional" equality $b_i\sigma = b_j\sigma \Longrightarrow \widehat{b}_i\sigma = \widehat{b}_j\sigma$ is replaced by

$$b_i\sigma = b_j\sigma \Longleftrightarrow \widehat{b}_i\sigma = \widehat{b}_j\sigma.$$

Theorem 5.1 *For any formula $A \in \widehat{L}$*

1. *$\widehat{\mathcal{PU}}_1 \vdash A$ iff A^* is valid for all interpretations $*$ in r.s.r. with functional relation R^{-1};*
2. *$\widehat{\mathcal{PU}}_{1-1} \vdash A$ iff A^* is valid for all interpretations $*$ in r.s.r. with invertible reading procedure θ.*

Proof. Similar to the proof of Theorem 4.1. ■

The logics $\widehat{\mathcal{PU}}_1$ and $\widehat{\mathcal{PU}}_{1-1}$ are also decidable. The satisfiability algorithms from the completeness proofs for these logics can be used in the same way as that for $\widehat{\mathcal{PU}}$ to construct r.s.r. without double stored sentences.

References

[1] S. Artëmov and T. Strassen. Functionality in the Basic Logic of Proofs. Technical report IAM 93-004, Universität Bern, January, 1993.

[2] J.-L.Lasser, M.J.Maher and K.Marriot. Unification revisited. In: "Foundations of Deductive Databases and Logic Programming" J.Minker (Ed.), Morgan Kauffman, pp.587-626, 1987.

[3] S. Artëmov. Logic of proofs. *Annals of Pure and Applied Logic*, v.67, 1994 (to appear).