ELSEVIER

Theoretical Computer Science 175 (1997) 15-27

Theoretical Computer Science

Program schemata vs. automata for decidability of program logics [1]

N.V. Shilov*

Russian Academy of Sciences, Institute of Informatics Systems, 6, Acad. Lavrentjev pr., 630090, Novosibirsk, Russia

Abstract

A new technique for decidability of program logics is introduced. This technique is applied to the most expressive propositional program logic – mu-calculus.

0. Introduction

We would like to present program scheme technique (PST) for decidability of program and polymodal propositional logics. This technique leads to one-exponential time upper bounds usually.

Second-order propositional dynamic logic (SO-PDL) of program schemata is a variant of propositional dynamic logic (PDL) [6, 8] with program schemata and second-order quantifiers. Unfortunately it is undecidable, but the validity in Herbrand models (HM) is decidable with a one-exponential upper bound. This is based on a polynomial reduction of the validity problem in HM to the same problem but for sentences in the special form. The original one-exponential algorithm solves the last problem. The syntax and semantics of SO-PDL together with some results on the expressive power and the undecidability of this logic are presented in Section 1 of this paper. The decidability of SO-PDL in HM is proved in Section 2.

We have to remark that some authors have introduced and investigated second-order variants of program logics – quantified propositional temporal logic (QPTL), which is propositional linear temporal logic equipped with quantification over propositions. In contrast to the results mentioned above, QPTL [15] is decidable with a non-elementary complexity and is as expressive as the monadic second-order logic S1S. The complete calculus for QPTL has been presented quite recently [9]. In fact, the completeness

* E-mail: shilov@isi.itfs.nsk.su.

[1] This work was supported by the Russian Basic Research Fund (Project 93-01-00986) and the International Association for the Promotion of Cooperation with Scientists from the Independent States of Former Soviet Union (INTAS contract 1010-CT93-0048).

proof is based on the reduction of a QPTL formula into a Büchi automaton, and performing equivalence transformations on these automata, formally justifying these transformations.

As an example of applicability of PST, we would like to consider Mu-calculus (MuC) [10, 12] – a propositional polymodal logic with the least (μ) and the greatest (ν) fixed points. In [16] a triple-exponential upper bound for MuC is proved on the basis of automata theoretical technique (ATT). We recall the syntax and semantics of MuC in Section 1 of this paper. In Section 3 the validity problem for MuC is reduced to the validity problem for SO-PDL in HM with a linear complexity. So, PST leads to decidability of MuC with a one-exponential upper bound and to a new proof of small model property (SMP) for MuC [16].

Another algorithmic problem for MuC is axiomatization. In the original paper [10] a very natural sound axiomatization for MuC was proposed, but the completeness of the axiomatization was proved for the fragment of MuC – for aconjunctive formulae only. A complete axiomatization of MuC and completeness of the axiom system from [10] were open problems during 10 years. The first problem was solved by Walukiewicz in 1993 [17] and the second one in 1994 by him too. As a “side effect” of the application of PST to MuC we get another sound axiomatization of MuC which is complete for two fragments of MuC: for the so-called diamond formulae and the so-called ν-free formulae. This axiomatization is presented in Section 3 too. The completeness of this axiomatization for MuC itself is an open question.

1. Syntax, semantics and expressive power of mu-calculus and second-order propositional dynamic logic

The propositional mu-calculus (MuC) is a propositional program – a polymodal logic with constructions for the least and the greatest fixed points [10, 12]. The syntax of MuC is constructed from a countable alphabet of (program) symbols and a countable alphabet of (propositional) variables as follows. Propositional variables are (elementary) formulae. Propositional combinations (negations, conjunctions and disjunctions, etc.) of formulae are formulae. Modalities in MuC are associated with program symbols: If a is a symbol then $[a]$ and $< a >$ is the pair of modalities associated with this symbol. Fixed point constructions are associated with propositional variables: If p is a variable then $\mu p.$ and $\nu p.$ is the pair of fixed point constructions associated with this variable and applicable to formulae with positive instances of the variable only. The semantics of MuC is defined in Kripke structures where program symbols are interpreted as binary relations and propositional variables – as unary predicates. Propositional operations have the usual semantics. For a program symbol a the semantics of the associated modalities $[a]$ and $< a >$ is the same as the usual K-modalities semantics but with respect to an interpretation of the symbol a. The semantics of fixed point constructions is straightforward from their names – the least and the greatest fixed points with respect to the inclusion as a partial order on interpretations of propositional variables. But this semantics can be defined constructively too in accordance with the Tarski–Knaster theorem for the least and the greatest fixed points of a monotonic function over subsets of a set.

The second-order propositional dynamic logic (SO-PDL) is an extension of propositional dynamic logic (PDL) [6, 8] with quantifiers over propositional variables. The syntax of SO-PDL is constructed from a countable alphabet of (program) symbols and a countable alphabet of (propositional) variables too. The syntax consists of (program) schemata and (logical) formulae which are defined by mutual induction.

Schemata: Let us use the natural numbers extended by ∞ as labels. The label 0 is called the start-label and the label ∞ is the exit-label. An assignment is an expression of the form l : *a goto* L, where l is a label, a is a program symbol and L is a finite set of labels. A test is an expression of the form l : *if* A *then* $L+$ *else* $L-$, where l is a label, A is a formula and $L+, L-$ are finite sets of labels. A (program) scheme is a finite set of assignments and tests.

Formulae: Propositional variables are (elementary) formulae. Propositional combinations (negations, conjunctions and disjunctions, etc.) of formulae are formulae. Some modalities are associated with program schemata (similarly to MuC), but others are not: If S is a program scheme then $[S]$ and $< S >$ is the pair of modalities associated with this scheme as well as $\Box$ and $\Diamond$ are modalities too. Strong and weak quantifiers over propositional variables are admissible too: If p is a propositional variable then $\forall p.$, $\forall^f p.$, $\exists p$ and $\exists^f p.$ are quantifier prefixes with this variable.

The semantics of SO-PDL is defined in Kripke structures where program symbols are interpreted as binary relations and propositional variables – as unary predicates. The semantics of a program scheme in a Kripke structure is its input–output (IO) binary relation which can be defined in the usual manner [8]. Propositional operations have the usual semantics. For a program scheme S the semantics of the associated modalities $[S]$ and $< S >$ is the same as the usual K-modalities semantics but with respect to an IO-relation of the scheme S. Modalities $\Box$ and $\Diamond$ are the usual S5-modalities, so their semantics is traditional. The semantics of quantifiers is straightforward from their names – for all/some (finite) interpretation of a propositional variable as a unary predicate.

A traditional algorithmic problem for logics is the decidability. For MuC and SO-PDL this problem can be formulated as follows: Is there an algorithm deciding for any formula its validity in each Kripke structure on each state. For MuC the answer (on principle) is "Yes" because MuC can be interpreted in Rabin' second-order logic of several monadic successor functions [13]. But the next question arises for MuC: what about lower and upper bounds for the decidability of MuC? A one-exponential low bound for propositional dynamic logic (PDL) [6, 8] is well known but a traditional reduction of PDL to MuC [16] is exponential. In [16] a triple-exponential time upper bound for MuC is proved on the basis of a polynomial reduction to a certain emptiness problem for finite automata on infinite trees. So an effective procedure for the emptiness problem mentioned above can improve the upper time bound for MuC too. It seems interesting and important to develop a self-contained technique oriented for decidability of a variety of program-polymodal logics in general and of MuC in particular. The program scheme technique (PST) introduced in [11] is a good candidate for these.

Lemma 1. *The expressive power of MuC is less than or equal to the expressive power of SO-PDL.*

Proof. Let us consider a combined logic MuC+SO-PDL – with united syntax and semantics. For any formulae A and B, for any program scheme S, for any program symbol a and for any propositional variable p the following formulae are valid:

$(A \rightarrow B) \leftrightarrow ((\neg A) \vee B)$,
$(\neg(\neg A)) \leftrightarrow A$,
$(\neg(A \wedge B)) \leftrightarrow ((\neg A) \vee (\neg B))$,
$(\neg(< S > A)) \leftrightarrow ([S](\neg A))$,
$(\neg(< a > A)) \leftrightarrow ([a](\neg A))$,
$(\neg(\diamond A)) \leftrightarrow (\square(\neg A))$,
$(\neg(\exists p.A)) \leftrightarrow (\forall p.(\neg A))$,
$(\neg(\exists^f p.A)) \leftrightarrow (\forall^f p.(\neg A))$,
$\neg \mu p.B(p) \leftrightarrow \nu p.\neg B(\neg p)$.

So it is sufficient to consider so-called normal formulae of MuC+SO-PDL only – i.e. those formulae without equivalences and implications in which negations are applied to the proposition variables only. Similarly to MuC those formulas are monotonous [16]. Hence for any normal formulae B and C, for any propositional variable p the following formulae are equivalent:

$$B(\mu p.C(p)) \text{ and } \forall p.(\square(C(p) \rightarrow p) \rightarrow B(p)).$$

Similarly, for any normal formulae B and C, for any propositional variable p (i.e. p is absent in B and C) the following formulae are equivalent:

$$B(\nu p.C(p)) \text{ and } \exists p.(\square(p \rightarrow C(p)) \wedge B(p)).$$

So all fixed points can be eliminated. □

For any program schemata $S1$ and $S2$ let $EQ(S1, S2)$ be a formula with the following semantics: $EQ(S1, S2)$ is valid in a state s of a Kripke structure iff $\{t \mid (s,t) \in IO(S1)\} = \{t \mid (s,t) \in IO(S2)\}$.

Lemma 2. *PDL extended by EQ is undecidable.*

Proof (*sketch*). *EQ* permits to check that in a Kripke structure the interpretations of some program symbols are commutative, some inverse. Those properties are sufficient for a simulation of counter-machines in terms of PDL, so PDL with *EQ* is undecidable. □

Since PDL is expressible in MuC [10, 16], we get as a consequence of Lemmas 1 and 2 the following theorems.

Theorem 1. *SO-PDL is undecidable.*

Proof. For any program schemata $S1$ and $S2$ the formula $EQ(S1,S2)$ is equivalent to $\forall p.(< S1 > p \leftrightarrow < S2 > p)$ where p is a new propositional variable (i.e. p is absent in $S1$ and $S2$). □

Theorem 2. *The expressive power of MuC is less then the expressive power of SO-PDL.*

Proof (*sketch*). Let us consider the following SO-PDL formula $\forall p.(< (a) > p \leftrightarrow < (b) > p)$, where p is a propositional variable, a and b are program symbols and (a) and (b) are notations for program schemata $\{0 : a\ goto\ \{\infty\}\}$ and $\{0 : a\ goto\ \{\infty\}\}$ respectively. If this formula is equivalent to a formula A of MuC then (since it is equivalent to $EQ((a),(b))$) for any schemata $S1$ and $S2$ the formula $EQ(S1,S2)$ is equivalent to the formula $A(S1/a,\ S2/b)$ which is expressible in MuC. (Here $A(S1/a,\ S2/b)$ is the result of the substitution of the program schemata $S1$ and $S2$ on places of the program symbols a and b respectively.) So the formula $\forall p.(< (a) > p \leftrightarrow < (b) > p)$ is not expressible in MuC. □

2. Program scheme technique and decidability

A Herbrand model (HM) for a formula or a scheme is a Kripke structure whose domain is the set T of all strings (including the empty string λ) over the alphabet of program symbols which occur in this formula or this scheme and the interpretation for those program symbols is the concatenation, i.e. for any program symbol a and any string s the following holds: $a(s) = as$.

A (halting) assertion is a formula of SO-PDL in the form $PREF < S > TRUE$, where $PREF$ is a quantifier prefix and S is a program scheme with simple tests only, i.e. all conditions are propositional variables.

Lemma 3. *In Herbrand models, any formula of SO-PDL is equivalent to a halting assertion which can be constructed in linear time.*

Proof. Let us present the following reduction algorithm which transforms any formula into a halting assertion. We would like to describe this algorithm in general in terms of global steps and give some remarks on a feature of each step.

The first step is an elimination of all complex tests as follows: for formulae B and C, for a new propositional variable p (i.e. which is absent in B and C) the formula B is equivalent to the formula $\exists p.(\Box(p \leftrightarrow C) \wedge B(p/C))$ where $B(p/C)$ means substitution of p on place of each instance of C.

The next step is a so-called normalization, i.e. the elimination of all equivalences and implications and the filtration of the negation upto propositional variables in accordance with the traditional equivalences mentioned in the proof of Lemma 1.

The third step is a replacement of all elementary subformulae (i.e. all instances of propositional variables which are not tests) by so-called halting formulae as follows: a formula B is equivalent to the halting formula $< B? > TRUE$, where $B?$ is the abbreviation for the program scheme $\{0 : \mathit{if}\ B\ \mathit{then}\ \{\infty\}\ \mathit{else}\ \emptyset\}$.

The last step of the reduction consists in an interpretation of propositional operations and box-modalities in terms of diamond-modalities and second-order quantification. Let us use the traditional structured operation; for sequential composition of schemata, *if–then–else–fi* for deterministic choice of schemata, *while–do–od* for deterministic loop of a scheme, $\cup$ for non-deterministic choice of schemata and $*$ for non-deterministic loop of a scheme. For a program symbol a we denote by (a) the program scheme $\{0 : \ a\ \mathit{goto}\ \{\infty\}\}$. Let $HALT$ and $LOOP$ be the following schemata: $\{0 : \mathit{if}\ TRUE\ \mathit{then}\ \{\infty\}\ \mathit{else}\ \{0\}\}$ and $\{0 : \mathit{if}\ TRUE\ \mathit{then}\ \{0\}\ \mathit{else}\ \{\infty\}\}$ respectively.

Then the elimination of propositional operations for $\vee$ and $\wedge$ can be done as follows. For any disjoint quantifier prefixes $PREF_1$ and $PREF_2$ (i.e. $PREF_1$ and $PREF_2$ have no common propositional variables), for any program schemata S_1 (disjoint with $PREF_2$) and S_2 (disjoint with $PREF_1$) the following formulae are equivalent in Herbrand models:

$$((PREF_1 < S_1 > TRUE) \vee (PREF_2 < S_2 > TRUE)) \text{ and}$$

$$PREF_1 PREF_2 \exists p.(< \mathit{if}\ p\ \mathit{then}\ S_1\ \mathit{else}\ S_2\ \mathit{fi} > TRUE),$$

$$((PREF_1 < S_1 > TRUE) \wedge (PREF_2 < S_2 > TRUE)) \text{ and}$$

$$PREF_1 PREF_2 \forall p.(< \mathit{if}\ p\ \mathit{then}\ S_1\ \mathit{else}\ S_2\ \mathit{fi} > TRUE),$$

where p is a new propositional variable.

The modalities $\Box$ and $\Diamond$ can be simulated in Herbrand models by modalities $[UNI]$ and $< UNI >$ respectively, where UNI is the program scheme $(\cup_{a \in ACT}(a))*$ and ACT is the set of all program symbols of the formula.

The elimination of modalities associated with program schemata can be done as follows. For any quantifier prefix $PREF$, for any program schemata S_1 and S_2 let dS_1 be a deterministic scheme which simulates non-deterministic constructions *goto* ... , *then* ... and *else*... of S_1 by deterministic choice with respect to values of a vector of new propositional variables P and aborts this simulation and halts in the case when all variables from P are falsified. For example, the non-deterministic *goto*-construction in the assignment $l\ :\ a\ \mathit{goto}\ \{l_1,\ l_2\}$ is simulated by the following fragment:

$$l : \ a\ \mathit{goto}\ \{l'\}$$

$$l' : \ \mathit{if}\ p_1\ \mathit{then}\ \{l_1\}\ \mathit{else}\ \{l''\}$$

$$l'' : \ \mathit{if}\ p_2\ \mathit{then}\ \{l_2\}\ \mathit{else}\ \{\infty\},$$

where l' and l'' are new labels and p_1 and p_2 are new propositional variables. Then the following formulae are equivalent in Herbrand models:

$(< S_1 > (PREF < S_2 > TRUE))$ and

$(\exists P.\ PREF\ (< dS_1\ ;\ if\ (\vee P)\ then\ S_2\ else\ LOOP\ fi > TRUE))$,

$([S_1](PREF < S_2 > TRUE))$ and

$(\forall^f P.\ PREF\ (< dS_1\ ;\ if\ (\vee P)\ then\ S_2\ else\ HALT\ fi > TRUE))$. $\square$

Lemma 4. *The validity in Herbrand models for halting assertions is decidable in one-exponential time.*

Proof. Let us choose and fix a halting assertion $Q_1 p_1 \ldots Q_n p_n (< S > TRUE)$ where $Q_1, \ldots, Q_n$ are quantifiers, $p_1, \ldots, p_n$ are different propositional variables and S is a program scheme. Without loss of generality we can suppose that

the start-label 0 marks an assignment in S,

each label marks the unique operator in S,

each propositional variable occurring in S occurs among $p_1, \ldots, p_n$.

Let $v_1, \ldots, v_n$ be different boolean variables. For any labels l_1 and l_2, for any evaluation of $v_1, \ldots, v_n$ (by boolean values) let us write $l_1 \rightsquigarrow (v_1 \ldots v_n) \rightsquigarrow l_2$ iff there exists a logical path (i.e. across tests) which is consistent with the evaluation. Let us define the notion of the type for a label as follows: if the label marks an assignment in S then its type is the program symbol from this assignment; the label ∞ has all possible types; otherwise the type of the label is undefined. Let us define *SPACE* as the set of all L where L is a set of labels of one and the same type. For any sets L_1 and L_2, for any evaluation of $v_1, \ldots, v_n$ let us write $L_1 \rightsquigarrow (v_1 \ldots v_n) \rightsquigarrow L_2$ iff for any label $l_2 \in L_2$ there exists a label $l_1 \in L_1$ such that $l_1 \rightsquigarrow (v_1 \ldots v_n) \rightsquigarrow l_2$.

The decision procedure for validity of the halting assertion $Q_1 p_1 \ldots Q_n p_n$ $(< S > TRUE)$ consists of the following two steps and the validation criterion.

The first step: For any i $(0 \leqslant i \leqslant n)$ let $(Q_i p_i)'$ be

the quantified boolean variable $Q_i v_i$ iff $Q_i \in \{\forall, \exists\}$,

or the evaluation of v_i by *FALSE* iff $Q_i \in \{\forall^f, \exists^f\}$.

Now we are going to define the sequence $D_0 \subseteq D_1 \subseteq \cdots \subseteq SPACE$ which may be infinite but stabilizes after an exponential number of steps. Let D_0 be $\{\{\infty\}\}$. For any $j \geqslant 0$ let D_{j+1} be $D_j \cup \{L_1 \in SPACE | (Q_1 v_1)' \ldots (Q_n v_n)'$: there exists $L_2 \in D_j$ such that $L_1 \rightsquigarrow (v_1 \ldots v_n) \rightsquigarrow L_2\}$. Let D_∞ be $\cup_{j \geqslant 0} D_j$.

The second step: For any i $(0 \leqslant i \leqslant n)$ let $(Q_i p_i)''$ be the quantified boolean variable

$\forall v_i$ iff $Q_i \in \{\forall, \forall_f\}$,

or $\exists v_i$ iff $Q_i \in \{\exists, \exists^f\}$.

Now we are going to define the sequence $E_0 \subseteq E_1 \subseteq \cdots \subseteq SPACE$ which may be infinite but stabilized after an exponential number of steps. Let E_0 be D_∞. For any $j \geqslant 0$ let

E_{j+1} be $E_j \cup \{L_1 \in SPACE | (Q_1 v_1)'' \dots (Q_n v_n)''$: there exists $L_2 \in E_j$ such that $L_1 \rightsquigarrow (v_1 \dots v_n) \rightsquigarrow L_2\}$. Let E_∞ be $\cup_{j \geqslant 0} E_j$.

Then the validation criterion is: the assertion $Q_1 p_1 \dots Q_n p_n (< S > TRUE)$ is valid iff $\{0\} \in E_\infty$ holds.

Indeed, let us consider the following subsets of the Herbrand domain T: for any $k \geqslant 0$ the set T_k consists of all strings of length less than k. So $T_0 = \emptyset$. For any k $(k \geqslant 0)$ and any i $(0 \leqslant i \leqslant n)$ let $(Q_i p_i)^k$ be the quantified interpretation for the proposition variable p_i:

$$Q_i p_i \subseteq T \text{ iff } Q_i \in \{\forall, \exists\},$$

$$\text{or } Q_i p_i \subseteq T_k \text{ iff } Q_i \in \{\forall^f, \exists^f\}.$$

Then for any $j \geqslant 0$ and for any $L \in SPACE$

$L \in D_j$ iff $(Q_1 p_1)^0 \dots (Q_n p_n)^0$: there exists a label $l \in L$ and a path from l to ∞ with j assignments at most which is consistent with an interpretation of $p_1, \dots, p_n$ by subsets of Herbrand domain chosen with respect to the quantifier prefix $(Q_1 p_1)^0 \cdots (Q_n p_n)^0$;

$L \in E_j$ iff $(Q_1 p_1)^j \cdots (Q_n p_n)^j$: there exists a label $l \in L$ and a path from l to ∞ which is consistent with an interpretation of $p_1, \dots, p_n$ by subsets of Herbrand domain chosen with respect to the quantifier prefix $(Q_1 p_1)^j \dots (Q_n p_n)^j$.

Both facts can be proved by induction. Since $T = \bigcup_{k \geqslant 0} T_k$ then the validation criterion follows from the above facts. □

As a consequence from Lemmas 3 and 4 we get

Theorem 3. *The validity problem for SO-PDL in Herbrand models is decidable in one-exponential time.*

3. Application of PST to MuC

Lemma 5. *The decidability problem for MuC is equivalent to the validity problem in countable Kripke structures.*

Proof (*sketch*). A formula B is said to be a subformula of a formula A iff the string B is a substring of A. Let us enumerate formulae of MuC with respect to the natural partial order for formulae, i.e. any formula A appears in the enumeration after all its subformulae: $A_0, \dots A_n, \dots$. Then the lemma follows from the proposition:

For any natural number n, for any model M with domain D, for any countable set of states $D' \subseteq D$ there exists a countable set D'' such that $D' \subseteq D'' \subseteq D$ and for all $0 \leqslant m \leqslant n$ the relation $N(A_m) \cap D' = M(A_m) \cap D'$ holds, where N is the restriction of M by D''.

(The proposition can be proved by induction on n.) □

Lemma 6. *The validity problem for MuC in countable Kripke structures is reducible to the validity problem for SO-PDL in Herbrand Models, and the time complexity of this reduction is linear.*

Proof (*sketch*). Let A be a formula of MuC. Let us consider a combined logic MuC+ SO-PDL – with united syntax and semantics and the formula A as a formula of this logic.

For any program symbol a which occurs in A let b_a, c_a and p_a be new program symbols and a new propositional variable. We would like to use the structured operations $*$, ; and ? again (see the proof of Lemma 3). Finally, let us denote by B_A the result of the replacement of each program symbol a in A by the program scheme $p_a?\,;\,(b_a)*\,;\,c_a$. Then the formula A is valid in all countable Kripke structures iff the formula B_A is valid in all countable Kripke structures where all program symbols are interpreted as graphics of general functions. This proposition is an analog of the reduction of PDL to deterministic PDL and is proved similarly [8]. At the same time a quantifier-free and S5-modalities-free ($\Box$ and $\Diamond$) formula of MuC+SO-PDL is valid in all countable Kripke structures where all program symbols are interpreted as graphics of general functions iff it is valid in all Herbrand models. This proposition is an analog of similar reductions from arbitrary models to Herbrand models in the classical theory of program schemata and can be proved with usage of the notion of the associated Herbrand model [7].

Since for any formulae B the following formulae

$$\neg\mu p.B(p) \quad \text{and} \quad \nu p.\neg B(\neg p)$$

are equivalent then we can suppose without loss of generality that the formula A is a normal formula. Since normal formulae of combined logic are monotonous (similarly to MuC [16]) then for any normal formulae B and C, for any propositional variable p the following formulae are equivalent:

$$B(\mu p.C(p)) \quad \text{and} \quad \forall p.(\Box(C(p) \to p) \to B(p))$$

$$B(\nu p.C(p)) \quad \text{and} \quad \exists p.(\Box(p \to C(p)) \wedge B(p)).$$

So, we can eliminate all fixed points in B_A and get a formula of SO-PDL which is valid in all Herbrand models iff the initial MuC formula is valid in all countable models. $\Box$

As a consequence from Theorem 4 and Lemmas 5 and 6 we get

Theorem 4. *MuC is decidable in one-exponential time.*

At the same time as a corollary of the proof of Lemma 4 we can get a new proof of the finite model property [16].

Theorem 5. *MuC has the finite model property: a formula is valid iff it is valid in all finite Kripke structures.*

Proof (*sketch*). Let A be a formula of MuC. Let B be the correspondent formula of SO-PDL which is constructed from A in accordance with the sketch of the proof of Lemma 6. Let $Q_1 p_1 \ldots Q_n p_n(<S> TRUE)$ be the halting assertion which is equivalent to B and is constructed in accordance with the proof of Lemma 3. As follows from the justification of the validation criterion from the proof of Lemma 4 $Q_1 p_1 \ldots Q_n p_n(<S> TRUE)$ is valid in all Herbrand models iff $(Q_1 p_1)^k \ldots (Q_n p_n)^k$: there exists a path from 0 to ∞ with $2k$ assignments at most which is consistent with the interpretation, where k is an exponential function of the size of S. So (without loss of the validity) it is possible to restrict the Herbrand domain T till T_{2*k} where $k = \exp|A|$ is an exponential function of the size of the formula A. □

In the original paper [10] a very natural sound axiomatization for MuC was proposed, but the completeness of the axiomatization was proved for the fragment of MuC – for aconjunctive formulae only. A complete axiomatization of MuC and completeness of the axiom system from [10] were open problems for 10 years. Both problems were solved by Walukiewicz in 1993 [17] and 1994, respectively.

We would like to present an alternative research approach to a complete axiomatization of MuC. The idea of this approach is similar to [14] and consists in the design of a sound axiomatic system and a deductive strategy which establish the deductive equivalence of PDL and some fragments of MuC.

Let us consider another combined logic PDL+MuC and accept as the start point a complete axiomatization of PDL [8]. Let us denote this axiomatization by AS. We would like to add to AS the new axiom scheme and two new inference rules. The axiom scheme is the equivalence mentioned above in the proof of Lemma 1 and in the sketch of the proof of Lemma 6.

$$AX\neg\mu p.A(p) \leftrightarrow \nu p.\neg A(\neg p).$$

The first inference rule is a PDL+MuC version of the equivalence

$$A(\mu p.C(p)) \text{ and } \forall p.(\Box(C(p) \to p) \to A(p))$$

mentioned in the proof of Lemma 1 and in the sketch of the proof of Lemma 6:

$$\text{IR1}\ \frac{[UNI](B(p) \to p) \to A(p)}{A(\mu p.B(p))}.$$

So the new axiom and the first inference rule are some steps of application of PST to MuC. The second new inference rule is not inspired by PST, but is some variant of induction

$$\text{IR2}\ \frac{A(true),\ [UNI](B(p) \to p) \to ([UNI]A(p) \to [UNI]A(B(p)))}{A(\nu p.B(p))}$$

In the inference rules *UNI* is the same program scheme as in the proof of Lemma 3, p is a propositional variable, $A(C)$ is a normal formula (i.e. negations may be applied to variables only) with instances of a subformula C, $A(D)$ is result of the substitution of a formula D for C in A.

Let us denote by AS1 the extension of the complete axiomatization AS for PDL by the inference rule IR1. Let us denote by AS2 the extension of AS1 by the inference rule IR2. Let us denote by μAS1 and μAS2 the result of the translation of all PDL-constructions of AS1 and AS2, respectively, in terms of MuC with respect to the standard procedure [10, 16]. Finally let us define

a μ-formula as a normal ν-free formula and

a diamond formula as a normal box-free formula.

Theorem 6. 1. *The axiom systems $AS1, AS2, \mu AS1$ and $\mu AS2$ are sound.*

2. *The axiom systems AS1 and μAS1 are complete for μ-formulae of PDL+MuC and MuC, respectively.*

3. *The axiom systems AS2 and μAS2 are complete for diamond formulae of PDL+MuC and MuC, respectively.*

Proof (*sketch*). Since the inference rule IR1 is a quantifier-free version of the equivalence then this rule is invertible. So the application of this rule

preserves the validity in both directions (up-down and down-up),

introduces instances of μ in up-down direction,

eliminates instances of μ in up-down direction.

So the strategy which consists in the application of IR1 in down-up direction is to nest the least fixed points of a μ-formula of PDL+MuC, this strategy leads from a μ-formula of PDL+MuC to a validity equivalent formula of PDL. Since the axiomatization AS is complete then AS1 = AS+IR1 is complete for μ-formulae of PDL+MuC. Since each formula of MuC can be considered as a formula of PDL+MuC and AS1 is complete for PDL+MuC then μAS1 – the translation of AS1 in terms of MuC – is complete for μ-formulae of MuC.

The last item of the theorem can be proved similarly based on the following fact: IR2 is invertible for diamond formulae of PDL+MuC. $\square$

Question: Are $AS2+AX$ and $\mu AS2+AX$ complete?

4. Conclusion

Decidability and axiomatization are not the only algorithmic problems for program logics. A new algorithmic problem for program logics is the model-checking problem, i.e. the evaluation of the validity set of a formula in a finite model. For MuC and SO-PDL this problem is decidable because the semantics of those logics can be defined constructively. But the next question arises: what about lower and upper bounds for this problem?

The model-checking problem as a mathematical problem originated as an approach to specification and verification of finite state systems. A stream of publications on applied model-checking is very wide now and can be a subject for a separate survey. We would like to point out [3] because of the importance of this paper for the verification practice. Furthermore, [3] demonstrates the following typical feature of applied computer-aided model-checking: MuC is an internal representation of external specifications and verification is done in terms of MuC' model-checking. The time bound for the direct model checking algorithm based on the constructive semantics of MuC is exponential on the length of a formula, and it turns out that the model-checking problem for MuC is an NP and co-NP problem [5], in contrast with a lot of program logics which are decidable with a one-exponential time bound as MuC itself but have a polynomial model-checking algorithm. So, the problem of finding an expressive fragment of MuC with a polynomial model-checking algorithm arises. In [5] one of such fragments is presented and generalized: this fragment consists of normal formulae of MuC such that in each conjunction only one subformula has instances of propositional variables. The class of such fragments introduced in [4] has the following restriction: the alternation of the fixed points have to be bounded in each class. The new fragment of such kind is presented and generalized in [2].

Another fragment of MuC with a polynomial model-checking algorithm is presented in [1]. This fragment consists of normal formulae such that all inner least fixed points are syntactical independent of all outer greatest fixed points. So, these formulae have no restrictions on the alternation of fixed points, neither on the discipline of modal operators or boolean connectives, but have the restriction on the dependence of fixed points. This fragment is more expressive than CTL [3] and PDL [6]. The correctness of this polynomial model-checking algorithm can be proved in terms of SO-PDL too.

Acknowledgements

PST was designed in collaboration with Prof. V.A. Nepomniaschy and the first presentation of PST was given in [11]. The author is very appreciative to Sergey Berezine and Dilian Gurov for a kind reading of the draft of the paper and comments, to Prof. Vladimir A. Zaharov for the information on second-order variants of program logics and to Prof. Larisa L. Maksimova for the kind attention to researches whose results are presented in the paper.

References

[1] S.A. Berezin and N.V. Shilov, An approach to effective model-checking of real-time finite-state machines in mu-calculus, *Internat. Symp. on Logical Foundation of Computer Science LFCS'94*, Lecture Notes in Computer Science, Vol. 813 (Springer, Berlin, 1994) 47–55.

[2] G. Bhat and R. Cleaveland, Efficient local model-checking for fragments of the modal-μ-calculus, *2nd Internat. Workshop TACAS'96*, Lecture Notes in Computer Science, Vol. 1055 (Springer, Berlin, 1996) 107–126.

[3] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill and L.J. Hwang, Symbolic model checking: 10^{20} states and beyond, *Inform. and Comput.* **98** (1992) 142–170.
[4] R. Cleaveland, M. Klain and B. Steffen, Faster model checking for the modal mu-calculus, *Internat. Conf. on Computer-Aided Verification CAV'92*, Lecture Notes in Computer Science, Vol. 663 (Springer, Berlin, 1993) 410–422.
[5] E.A. Emerson, C.S. Jutla and A.P. Sistla, On model-checking for fragments of mu-calculus, *Internat. Conf. on Computer-Aided Verification CAV'93*, Lecture Notes in Computer Science, Vol. 698 (Springer, Berlin, 1993) 385–396.
[6] M.J. Fisher and R.E. Ladner, Propositional dynamic logic of regular programs, *J. Comput. System Sci.* **18** (1979) 194–211.
[7] S.A. Greibach, *Theory of Program Structures: Schemes, Semantics, Verification*, Lecture Notes in Computer Science, Vol. 36 (Springer, Berlin, 1975).
[8] D. Harel, Dynamic logic, in: *Handbook of Philosophical Logic*, Vol. 2 (Reidel, Dordrecht, 1984).
[9] Y. Kosten and A. Pnueli, A complete proof system for QPTL (an extended abstract), *IEEE proc. LICS'95* (1995) 2–12.
[10] D. Kozen, Results on the propositional mu-calculus, *Theoret. Comput. Sci.* **27** (1983) 333–354.
[11] V.A. Nepomniaschy and N.V. Shilov, Non-deterministic program schemata and their relation to dynamic logic, *Internat. Conf. on Math. Logic and its Applications* (Plenum Press, New York, 1986).
[12] V.R. Pratt, A decidable mu-calculus: preliminary report, *22nd IEEE Symp. on Foundation of Computer Science* (1982) 421–427.
[13] M.O. Rabin, Decidability of second order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* **141** (1969) 1–35.
[14] N.V. Shilov, Proving halting in first-order dynamic logic, in: *Vichislitelnie systemi*, Novosibirsk, Institute of Mathematics, Vol. 124 (1988) 72–83.
[15] A.P. Sistla, M.Y. Vardy and P. Wolper, The complementation problem for Buchi automata with application to temporal logic, *Theoret. Comput. Sci.* **49** (1987) 217–237.
[16] R.S. Streett and E.A. Emerson, An automata theoretic decision procedure for the propositional mu-calculus, *Inform. and Comput.* **81** (1989) 249–264.
[17] I. Walukiewicz, A complete deduction system for the μ-calculus, Doctoral Thesis, Warsaw, 1993.