A New Proof of Exponential Decidability for the Propositional μ-Calculus with Program Converse*

K. Yi N.V. Shilov †

Research On Program Analysis System (ROPAS)
Department of Computer Science
Korean Advanced Institute of Science and Technology (KAIST)
Kusong-dong Yusong-gu 373-1
Taejon 305-701, Republic of Korea
{kwang, shilov}@ropas.kaist.ac.kr

Abstract

The propositional μ-Calculus (μC) is a powerful propositional program logic with fixpoints. μC decidability with exponential upper bound was sketched for the first time in 1988 by E. A. Emerson and Ch. S. Jutla on base of *automata-theoretic* technique, while a complete proof was published in 1999 only. Meanwhile M. Vardi sketched in 1998 an *automata-theoretic* proof of exponential decidability for the propositional μ-Calculus with program converse (μC^-). We believe that alternative, independent and automata-free proofs of exponential decidabilities for μC and for μC^- are important for validation of these upper bounds and due to a complexity of automata-theoretic proofs. Previously the author published in 1997 a proof of exponential upper bound for μC which exploited a so-called Program Schemata Technique (PST) for decidability of propositional program logics. This time an extended PST is applied to μC^- and yields exponential upper bound too.

Subject classification: 03B25 – Decidability of theories and sets of sentences, 03B70 – Logic in Computer Science, 68Q60 – Specification and verification.

*This work is supported by Creative Research Initiatives of the Korean Ministry of Science and Technology

†While on leave from A.P. Ershov Institute of Informatics Systems of Siberian Division of Russian Academy of Science, Novosibirsk, Russia

Key words: program schemata, program logics, fixpoints, second order quantifiers.

1 Introduction

A so-called Program Scheme Technique (PST) [13] is a powerful approach for proving decidability of program and polymodal propositional logics. It is completely self-contained, automata free technique yielding exponential upper time bounds. This time PST is applied to the propositional μ-Calculus [7] with program converse [14]. In principle, exponential decidability for this logic can be proved by means of an automata-theoretic technique [16] as well as decidability for the μ-Calculus itself [15, 2, 3]. μC exponential decidability was sketched for the first time in 1988 in a conference proceedings by E. A. Emerson and Ch. S. Jutla on base of *automata-theoretic* technique [2], while a complete proof was published in 1999 only [3]. Proof of E. A. Emerson and Ch. S. Jutla is based on improved upper time bounds for checking emptiness of *non-deterministic* automata on infinite trees, while M. Vardi exploits decidability of emptiness for *two-way alternating* automata on infinite trees. These impressive applications of the automata-theoretic technique and impossibility to apply classical decision techniques to these logics lead some part of program logic community to the opinion [17] that the automata-theoretic approach is the unique paradigm for proving decidability for complicated propositional program logics with second-order features. In spite of this opinion, the successful applications of PST were reported several times. Previously exponential upper time bounds were proved by means of PST for Propositional Dynamic Logic with loop [10] and for the μ-Calculus [12, 13].

PST is based on a reduction of the decidability of a propositional program logic to the validity problem for a variant of Second Order Propositional Dynamic Logic of program schemata (SOPDL) in a variant of Herbrand Models. SOPDL is a variant of Propositional Dynamic Logic (PDL) [4] with non-deterministic monadic program schemata [6] and with second order quantifiers (weak as well as strong). Herbrand Models without converse (HM) are models for Second Order Theory of Monadic Successors $\mathbf{S(n)S}$ [11]. Unfortunately SOPDL is undecidable, but the validity in HM is decidable with 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 of generalized halting assertions for non-deterministic schemata. A special exponential algorithm solves the last problem.

The paper presents how PST can be extended on SOPDL with program converse (SOPDL$^-$) in Herbrand Models with converse (HM$^-$) and how an extended PST yields exponential decidability for the propositional μ-Calculus with converse (μC$^-$). We believe that alternative, independent and automata-free proofs of exponential decidabilities for μC and for μC$^-$ are important for validation of these upper bounds and due to a complexity of automata-theoretic proofs. The rest of the paper is organized as follows. The syntax and semantics of μC^- and SOPDL$^-$ are presented in the section 2. PST is extended on SOPDL$^-$ in HM$^-$ in the next section 3. In the section 4 the validity problem for the μC^- is reduced to the validity problem in HM$^-$ for a special fragment of SOPDL$^-$. It leads to the decidability of the μC^- with exponential upper bound. PST was designed in collaboration with Prof. Nepomniaschy V.A. [8, 9, 10]. Author appreciate him for fruitful discussions of results presented in this paper.

2 Converse, μ-Calculus and SOPDL

The Propositional μ-Calculus (μC) is a propositional program logic with the least and the greatest fixpoints [7]. μC in brief is a polymodal variant of basic propositional modal logic **K** [1] with fixpoints. The syntax of μC is constructed from a finite alphabet of (program) symbols $\mathcal{A}$ and a finite alphabet of (propositional) variables $\mathcal{P}$ as follows. Propositional variables are (elementary) formulae. Propositional combinations (negations, conjunctions and disjunctions, etc.) of formulae are formulae. Modalities in μC are associated with program symbols: if a is a symbol then $[a]$ and $\langle a\rangle$ are modalities. Fixpoints are associated with propositional variables: if p is a variable then $\mu p.$ and $\nu p.$ are fixpoints applicable to formulae with positive instances of the variable only. The semantics of μC 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 $\langle a\rangle$ is the same as the usual semantics of **K**-modalities but with respect to an interpretation of the symbol a. The semantics of fixpoints is straightforward from their names – the least and the greatest fixed points with respect to $\subseteq$ as a partial order on interpretations of propositional variables.

The Second Order Propositional Dynamic Logic (SOPDL) [13] is an extension of Propositional Dynamic Logic (PDL) with monadic program schemata and quantifiers over propositional variables. The syntax of SOPDL is constructed from the above alphabets of (program) symbols and (propositional) variables too. The syntax consists of (non-deterministic monadic program) schemata and (logical) formulae which are defined by mutual induction.
Schemata. Let us use the Natural numbers extended by ∞ as labels (0 is call the start-label and ∞ - 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 boolean formula and $L+,\ L-$ are finite sets of labels. A 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. Almost all modalities are associated with program schemata (in contrast to μC): if α is a program scheme then $[\alpha]$ and $\langle\alpha\rangle$ are modalities. Two special modalities are $\Box$ and $\Diamond$. 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.

The semantics of SOPDL is defined in Kripke structures too. 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 [5, 6]. Let us also say that a scheme is *free* iff every path in the scheme is consistent with some Kripke structure and there is no *aborts* in the scheme (i.e. *goto* $\emptyset$, *then* $\emptyset$ or *else* $\emptyset$). Propositional operations have the usual semantics. For a program scheme α the semantics of the associated modalities $[\alpha]$ and $\langle\alpha\rangle$ is the same as the usual semantics of **K**-modalities but with respect to an IO-relation of the scheme α. Modalities $\Box$ and $\Diamond$ are the usual **S5**-modalities [1] "in all states" and "in some state". The semantics of quantifiers is straightforward from their names - "for all/some (finite) interpretation of a propositional variable". In a framework of SOPDL a special role belongs to *generalized halting assertions* i.e. formulae of the following form $(Qp)^*(\langle\alpha\rangle true)$ where $(Qp)^*$ is a quantifier prefix and α is a program schemata with *simple tests* only (i.e. propositional variables or their negations).

Both logics (μC and SOPDL) can be extended by converse [14]. Syntactically converse of a program symbol a is the expression $a-$. Semantically converse of a program symbol $a-$ (in Kripke structures) is the converse of semantics of the symbol a (i.e. a corresponding binary relation). μC and SOPDL with converse are denoted as μC$^-$ and SOPDL$^-$. In the framework of μC$^-$ the converse is *reasoning about the past* while in the framework of SOPDL$^-$ it is *backward computations.*

3 Program Scheme Technique for SOPDL$^-$

A traditional algorithmic problem for program logics is the decidability: is there an algorithm deciding for formulae their validity in all Kripke structures in all states. Exponential upper and low bounds for PDL [5] are well-known. It trivially implies the same low bound for μC$^-$. Unfortunately, SOPDL is undecidable [13]. The situation change in *Herbrand Models.* Let $\mathcal{A}$ be the alphabet of all program symbols. All Herbrand Models have fixed domain D_H and fixed interpretation R_H of program symbols, while interpretation P of propositional variables is variable. A definition of Herbrand Models without converse is quite simple [13]. The Herbrand Domain D_H in this case is $\mathcal{A}^*$ – a free semi-group generated by $\mathcal{A}$, i.e. a set of words $\mathcal{A}^*$ which can be presented as an infinite tree. The Herbrand Interpretation R_H of a program symbol a is a total function $R_H(a) : D_H \to D_H$ such that $w \mapsto aw$ for every $w \in \mathcal{A}^*$. In particular, Herbrand Models without converse (HM) are models for Rabin's Second Order

Logics of Monadic Successors **S(n)S** [11]. It is known that SOPDL is decidable in HM [13] with exponential upper bound. In the presence of the converse the definition of Herbrand Models is more complicated. The Herbrand Domain D_H in this case is $(\mathcal{A}^- \cup \mathcal{A})^* / \overset{\mathcal{A}}{\sim}$ – a free group generated by $\mathcal{A}$, i.e. a free semi-group $(\mathcal{A}^- \cup \mathcal{A})^*$ factored by equivalence $\overset{\mathcal{A}}{\sim}$ generated by equalities $aa^- = a^-a = \epsilon$ for all $a \in \mathcal{A}$. For every $w \in (\mathcal{A}^- \cup \mathcal{A})^*$ let us denote the equivalence class $w/\overset{\mathcal{A}}{\sim}$ as $\lfloor w \rceil_{\mathcal{A}}$. Let ε be the unity element $\lfloor \epsilon \rceil_{\mathcal{A}}$ of $(\mathcal{A}^- \cup \mathcal{A})^* / \overset{\mathcal{A}}{\sim}$. The Herbrand Interpretation R_H of a program symbol a is a total function $R_H(a) : D_H \to D_H$ such that $\lfloor w \rceil_{\mathcal{A}} \mapsto \lfloor aw \rceil_{\mathcal{A}}$ for every $w \in (\mathcal{A}^- \cup \mathcal{A})^*$. Let us denote Herbrand Models with converse as HM$^-$.

Theorem 1 *Formulae of SOPDL$^-$ in HM$^-$ are equivalent to generalized halting assertions which can be constructed in a linear time and which have free schemata if formulae are test-free.*

For proof – see the appendix.

Theorem 2 *The validity problem in HM$^-$ for SOPDL$^-$ generalized halting assertions with free schemata is decidable in exponential time.*

Sketch of proof. When a propositional variable is tested for its boolean value for every next time during an execution of a free scheme in some HM$^-$, it is always tested in a new state (i.e. a state where this variable was not tested yet) since every path in a free scheme is consistent with some model. Hence after every next assignment during an execution of a free scheme in Herbrand Models with converse we can select arbitrary boolean value for every propositional variable, since either we are in a new state, where the variable have not been checked, or we revisit a state where the variable have been checked but will not be checked again. Hence for free schemata there is no difference between computations in $(\mathcal{A}^- \cup \mathcal{A})^*$ and in $(\mathcal{A}^- \cup \mathcal{A})^* / \overset{\mathcal{A}}{\sim}$, they can "simulate each-other". Hence a SOPDL$^-$ generalized halting assertion with a free schema is valid in HM$^-$ generated by program symbols $\mathcal{A}$ iff it is valid as SOPDL assertion in HM generated by old program symbols $\mathcal{A}$ and new program symbols $\mathcal{A}^-$. But validity problem for generalized halting assertions in HM is decidable with exponentional upper bound [13]. A sketch of the proof is over.

4 Decidability Results for μC$^-$ and SOPDL$^-$

Theorem 3 *The μ-Calculus with converse is decidable in exponential time.*

Sketch of proof. First let us remark that the decidability problem for μC$^-$ is equivalent to the validity problem for μC$^-$ formulae in HM$^-$ and the complexity of this reduction is linear. This fact is closely related to a so-called tree model property [16] but can be proved independently – just similar to the corresponding theorem for μC in [13]. Really, for every program symbol a let f_a, g_a and p_a be new program symbols and a new propositional variable. Let also F_ϕ be a result of a substitution of schemata $\big((p_a?);(f_a)^*;g_a\big)$ and $\big(g_a^-;(f_a^-)^*;(p_a?)\big)$ instead of all instances of a and $a-$ in a formula ϕ of $\mu\mathrm{C}^-$ (here ; and * stand for sequential composition and non-deterministic iteration). The initial formula ϕ is valid iff the resulting formula F_ϕ is valid in all HM^-.

Next let us eliminate all instances of the converse and fixpoints in accordance with the following equivalences:

$\langle a-\rangle\psi \ \leftrightarrow\ \forall p.(\Box(\psi \to [a]p) \to p)$,
$[a-]\psi \ \leftrightarrow\ \exists p.(\Box(\langle a\rangle p \to \psi) \wedge p)$,
$\mu p.\psi(p) \ \leftrightarrow\ \forall p.(\Box(\psi(p) \to p) \to p)$,
$\nu p.\psi(p) \ \leftrightarrow\ \exists p.(\Box(p \to \psi(p)) \wedge p)$.

This elimination leads to converse- and test-free formulae of SOPDL^-. In accordance with theorems 1 and 2 validity problem in HM^- for formulae of this kind is decidable in exponential time.

Theorem 4 *The SOPDL$^-$ is decidable in HM$^-$ in exponential time.*

Sketch of proof. Every formula $\langle\alpha\rangle p$ and $[\alpha]p$ (where α is a scheme and p is a propositional variable) can be presented as a formula of μC. For example, if α is a scheme $\{(0:\ a\ goto\ \{0,1\}),\ (1:\ if\ q\ then\ \{1,\infty\}\ else\ \emptyset)\}$ (where a and q are program symbol and propositional variable) then $\langle\alpha\rangle p$ can be presented as $\mu\ at0.\Big(\langle a\rangle\Big(at0 \vee \mu\ at1.\big(q \wedge (at1 \vee p)\big)\Big)\Big)$ and $[\alpha]p$ can be presented as $\nu\ at0.\Big([a]\Big(at0 \wedge \nu\ at1.\big(\neg q \vee (at1 \wedge p)\big)\Big)\Big)$ where $at0$ and $at1$ are new propositional variables. Then all new fixpoints can be eliminated as is sketched in the proof of the theorem 3. Hence in this way we can reduce a formula of SOPDL^- to a formula without tests in linear time. In accordance with theorems 1 and 2 validity problem in HM^- for formulae of this kind is decidable in exponential time.

Concluding remarks. Program Schemata Technique for SOPDL^- presented in the paper leads to decidability of the propositional $\mu\mathrm{C}^-$ with exponentional upper bound. Another algorithmic problem for $\mu\mathrm{C}^-$ axiomatization in general and how to axiomatize $\mu\mathrm{C}^-$ on based on PST in particular. In this context we would like to remark that in the paper [7] a natural sound axiomatization for μC was proposed, but the completeness of the axiomatization was proved for a fragment of this logic only. The completeness problem for μ-Calculus was an open problem during 10 years. Finally it solved by I. Walukiewicz in 1993 [17, 18] on base of theory of automata on infinite trees.

References

[1] Bull R.A., Segerberg K. *Basic Modal Logic.* Handbook of Philosophical Logic, v.II, Reidel Publishing Company, 1984 (1-st ed.), Kluwer Academic Publishers, 1994 (2-nd ed.), p.1-88.

[2] Emerson E.A., Jutla C.S. *The Complexity of Tree Automata and Modal Logics of Programs.* Proc. of 29-th IEEE Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1988

[3] Emerson E.A., Jutla C.S. *The Complexity of Tree Automata and Logics of Programs.* SIAM J. Comput., v.29, n1, 1999, p.132-158.

[4] Fisher M.J. Ladner R.E. *Propositional dynamic logic of regular programs.* J. Comput. System Sci., v.18, n.2, 1979, p.194- 211.

[5] Harel D. *Dynamic Logic.* Handbook of Philosophical Logic, v.II, (1-st ed.), Kluwer Academic Publishers, 1994 (2-nd ed.) p.498-604.

[6] Harel D., Sherman R. *Propositional Dynamic Logic of Flowcharts.* Information and Control, v.64, 1985, p.119-135.

[7] Kozen D. *Results on the Propositional Mu-Calculus.* Theoretical Computer Science, v.27, n.3, 1983, p.333-354.

[8] Nepomniaschy V.A. Shilov N.V. *Non-deterministic Program Schemata and there relation to Dynamic Logic.* Int. Conf. on Math. Logic and its Applications, Plenum Press, 1986, p.137-149.

[9] Nepomniaschy V.A. Shilov N.V. *Non-deterministic Program Schemata and there relation to Dynamic Logic.* Cybernetics, n.3, 1988, p.12-19 (in Russian, English translation by Plenum Press)

[10] Nepomniaschy V.A. Shilov N.V. *Program Schemata Technique for decidability of Propositional Dynamic Logic variants.* Proc. of Int. Conf. COLOG'88, v.2, Tallinn 1988, p.193-205.

[11] Rabin M.O. *Decidability of second order theories and automata on infinite trees.* Trans. Amer. Math. Soc., v.141, 1969, p.1-35.

[12] Shilov N.V. *Propositional Dynamic Logic with Fixed Points: algorithmical tools for verification of finite state machines.* LNCS, v.620, 1992, p.452-458

[13] Shilov N.V. *Program schemata vs. automata for decidability of program logics.* Theoretical Computer Science, v.175, n.1, 1997, p.15-27

[14] Streett R.S *Propositional dynamic logic of looping and converse is elementary decidable.* Information and Control, v.54, n.1-2, 1982, p.121-141.

[15] Streett R.S. Emerson E.A. *An Automata Theoretic Decision Procedure for the Propositional Mu-Calculus.* Information and Computation, v.81, n.3, 1989, p.249-264.

[16] Vardi M.Y. *Reasoning about the past with two-way automata'.* Lecture Notes in Computer Science, v.1443, 1998, p.628-641.

[17] Walukiewicz I. *A Complete Deduction System for the μ - Calculus.* Doctoral Thesis, Warsaw, 1993.

[18] Walukiewicz I. *On completeness of the μ-calculus.* IEEE Computer Society Press, Proc. of 8-th Ann. IEEE Symposium on Logic in Computer Science, 1993, p.136-146.

5 Appendix: proof of theorem 1

Algorithm comprises two stages: *preprocessing* and *while loop*. It exploits a special schemata $(A?) \equiv \{(0 : \ if\ A\ then\ \{\infty\}\ else\ \{1\}), (1 : \ a\ goto\ \{1\})\}$ (where a is an program symbol, while A is a boolean formula) and two structured operations: a *sequential composition* $(\alpha;\beta)$ and a *deterministic choice* $(if\ A\ then\ \alpha\ else\ \beta)$ where α, β are schemata and A is a boolean formula. Let us omit (due to space limitations) syntactical details of representation of this operations, but illustrate only on fig. 1. Let us remark also that elementary schemata are free, sequential composition of free schemata with disjoint sets of propositional variable is a free scheme as well as deterministic choice between free schemata by means of a new propositional variable.

Figure 1: Structured operations

Preprocessing.
• *Elimination of complex conditions.* For all formulae ϕ and ψ, for a new propositional variable q the formula ϕ is equivalent to the formula $\exists q.(\Box(q \leftrightarrow \psi) \wedge \phi(q/\psi))$.
• *Normalization* i.e. filtration of negation up to propositional variables in accordance with the traditional modal logic equivalences:

$(\neg(\neg\phi)) \leftrightarrow \phi$,
$(\neg(\phi \wedge \psi)) \leftrightarrow ((\neg\phi) \vee (\neg\psi))$,
$(\neg(\langle\alpha\rangle\phi)) \leftrightarrow ([\alpha](\neg\phi))$,
$(\neg(\Diamond\phi)) \leftrightarrow (\Box(\neg\phi))$,

$(\neg(\forall^{(f)} p.\phi)) \leftrightarrow (\exists^{(f)} p.(\neg\phi))$.

• *Converse elimination* in accordance with the following equivalence:
$(\langle a-\rangle\psi) \leftrightarrow \forall p.(\Box(\psi \to [a]p) \to p)$.

• *Elimination of* **S5***-modalities.* Equivalence of formulae $\Box\phi$, $\Diamond\phi$ and formulae $([uni]\phi)$, $(\langle uni\rangle\phi)$ holds for every formula ϕ and a scheme *uni* which visits every element of the Herbrand domain with converse. We are especially interested in a scheme *uni* which visits every element only once: this scheme every time after an assignment $a \in \mathcal{A}$ can pass control nondeterministically to any assignment except $a-$ and vice versa, every time after an assignment $a-$ it can pass control nondeterministically to any assignment except a. An example of *uni* for $\mathcal{A}$ consisting of two program symbols f and g is presented on fig. 2.

$$
\begin{array}{ccccccc}
\infty & & & & & & \infty \\
 & \nwarrow & & & & \nearrow & \\
 & \Rightarrow f & \longleftrightarrow & & g \Leftarrow & & \\
 & \uparrow & \nwarrow & \nearrow & \uparrow & & \\
 & | & & 0 & | & & \\
 & \downarrow & \swarrow & \searrow & \downarrow & & \\
 & \Rightarrow g^{-} & \longleftrightarrow & & f^{-} \Leftarrow & & \\
 & \swarrow & & & & \searrow & \\
\infty & & & & & & \infty
\end{array}
$$

Figure 2: Program *uni* for $\mathcal{A} = \{f, g\}$

• *Initiation.* Every elementary subformula p or $\neg p$ is equivalent to a halting assertion $\langle(p?)\rangle true$ and $\langle((\neg p)?)\rangle true$ respectively.

Loop while the formula is not a generalized halting assertion:

• *Elimination of propositional operations* $\vee$ *and* $\wedge$. For all disjoint quantifier prefixes[1] $(Q'p')^*$ and $(Q''p'')^*$, for all program schemata α (disjoint[1] with $(Q''p'')^*$) and β (disjoint[1] with $(Q'p')^*$) the following formulae are equivalent in Herbrand Models with converse:

$$(((Q'p')^*\langle\alpha\rangle true) \vee ((Q''p'')^*\langle\beta\rangle true)) \text{ and } (Q'p')^*(Q''p'')^*\exists q.(\langle if\ q\ then\ \alpha\ else\ \beta\rangle true),$$
$$(((Q'p')^*\langle\alpha\rangle true) \wedge ((Q''p'')^*\langle\beta\rangle true)) \text{ and } (Q'p')^*(Q''p'')^*\forall q.(\langle if\ q\ then\ \alpha\ else\ \beta\rangle true),$$

where q is a new propositional variable.

• *Elimination of nesting modalities.* For every scheme α let $d^e_P(\alpha)$ be a deterministic scheme which simulates non-deterministic *goto* , *then* and *else* of α by deterministic tests with respect to values of a vector of new propositional variables P and pass control to a new *emergency* label e in the case when all new variables are falsified. For example, the non-deterministic *goto* in the assign-

[1] i.e. without common propositional variables

ment $l:\ a\ goto\ \{l_1\ ,\ l_2\}$ is simulated by a fragment presented on fig. 3 where p_1 and p_2 are new propositional variables P, e is the emergency label. Then for all quantifier prefix $(Qp)^*$ and program schemata α and β the following formulae are equivalent in Herbrand Models with converse:

$(\langle\alpha\rangle((Qp)^*\langle\beta\rangle true))$ and $(\exists P.\ (Qp)^*\ (\langle d^e_P(\alpha)\ ;\ \beta_{e-loop}\rangle true))$,

$([\alpha]((Qp)^*\langle\beta\rangle true))$ and $(\forall^f P.\ (Qp)^*\ (\langle d^e_P(\alpha)\ ;\ \beta_{e-halt}\rangle true))$,

where $\beta_{e-loop} \equiv (\beta\ \cup\{(e:\ \ goto\ \{e\})\})$ is an extension of β which loops in emergency, and $\beta_{e-halt} \equiv (\beta\ \cup\{(e:\ goto\ \{infty\})\})$ is an extension of β which halts in emergency.

$$
\begin{array}{ccccccc}
l:a & \longrightarrow & p_1 & \stackrel{-}{\longrightarrow} & p_2 & \stackrel{-}{\longrightarrow} & e \\
 & & \downarrow + & & \downarrow + & & \\
 & & l_1 & & l_2 & &
\end{array}
$$

Figure 3: Fragment $d^e_{p_1 p_2}(l:a\ goto\ \{l_1\ ,\ l_2\})$

Reduction is over. Let us remark that it always terminates since each iteration of the loop decreases amount of propositional operations and nesting modalities. Let us also remark that if an initial SOPDL$^-$ formula is test-free then all schemata in it are free as well as all imposed determinations $d^e_P(uni)$. Hence applications of sequential composition and deterministic choice to these schemata lead to free schemata again.