On the equivalence-checking problem for polysemantic models of sequential programs

Ivan M. Zakharyaschev, Vladimir A. Zakharov

Abstract. We introduce a new propositional model of computation for sequential computer programs. A distinctive feature of this model is that program runs and the results of computations are defined by means of two independent operational semantics. One of them can be regarded as an internal semantics that is used for routing runs in the control-flow graph of a program. The other one can be viewed as an observational semantics which is used for interpreting the results of a program execution. We show that some conventional models of sequential and recursive programs can be embedded into our model. We consider the equivalence-checking problem for the presented model and develop a uniform approach to the design of efficient equivalence-checking algorithms.

1. Introduction

The investigation of the equivalence-checking problem is of great importance in computer programs optimization, maintenance and understanding. Although it is hardly possible to formalize completely the term "to understand the meaning of a program", nevertheless we can estimate the extent of our understanding using the following criterion: *the meaning of a given program is understood if we can distinguish it from programs with different meaning*. By defining an equivalence relation on programs in such a way that programs with the same meaning are equivalent, we face the need to consider the *equivalence-checking problem*. Usually, the meaning of a program is specified by its observable behavior. In the case of sequential programs, with every program π we can associate the input-output relation R_π computed by the program and consider this relation as the observable behavior of the program. Thus, the equivalence-checking problem is to verify whether two given sequential programs compute the same input-output relation.

As follows from the well-known Rice-Uspensky Theorem [21, 26], the equivalence-checking problem defined above is undecidable for any programming system Σ satisfying the following conditions:

1) any recursive input-output relation R is computable by some program π

from Σ, i.e., $R = R_\pi$;

2) there exists a program $\mathcal{U}$ which can simulate any program from Σ, i.e., $R_{\mathcal{U}}(\pi, in, out) = R_\pi(in, out)$ for every $\pi \in \Sigma$;

3) any programming system Σ' satisfying 1) and 2) above can be effectively translated into Σ, i.e., there exists a recursive function $f : \Sigma' \to \Sigma$ such that $R_{\pi'} = R_{f(\pi')}$ for every $\pi' \in \Sigma'$.

It is worth noting that all programming systems used in practice comply with these three conditions. To obtain positive results (effective criteria, semi-decision procedures, etc.) for the equivalence-checking problem, one can be guided by the following approach. Given a programming system Σ, consider a model of computation $\widehat{\Sigma}$ which has the same syntax but a simpler semantics. Based on this semantics, define an equivalence relation $\sim$ on programs. We say that $\widehat{\Sigma}$ *approximates* Σ if $\pi_1 \sim \pi_2$ implies $R_{\pi_1} = R_{\pi_2}$ for every pair π_1, π_2 of programs. If $\widehat{\Sigma}$ approximates Σ, it suffices to check $\pi_1 \sim \pi_2$ to certain that $R_{\pi_1} = R_{\pi_2}$. If the equivalence-checking problem "$\pi_1 \sim \pi_2$?" is decidable in $\widehat{\Sigma}$ then a decision procedure in $\widehat{\Sigma}$ can be used for checking program equivalence in Σ.

This approach goes back to the seminal papers by Lyapunov and Yanov [9, 27]. It was further developed and studied in details in [4, 8, 13]. A whole spectrum of computational models that can be used for approximating the behavioral equivalence was introduced for sequential programs [10, 13, 19], functional programs [1, 3, 5], parallel and distributed programs [7]. Lattice-theoretic properties of the approximation relation on such models were studied in [18]. The utility of this approach is strengthened by the design of efficient equivalence-checking algorithms for many approximating models [1, 7, 11, 15, 20, 22, 23, 24, 25, 28, 29, 30].

Usually the semantics of approximating models of computation is defined in terms of transition systems (Kripke models) $M = \langle S, R, \rho \rangle$ (see [6]), where S is a state space whose elements are associated with data states of programs, R is a transition relation which interprets program statements, and ρ is an evaluation of predicates in branching statements. A run of a program π on M is defined as a double route (w_π, w_S), where w_π is a route in the control flow graph of π, and w_S is the corresponding route in the state space S. A program builds both routes in the framework of a single operational semantics. If a run terminates then the final data state s reached by w_S is accepted as the result of the run.

However, there are cases when such models are not suitable for capturing some features of program computations. For example, we may assume that the program execution allows more than one output along its run. Then the result of computation is specified not only by its final data state, but also by some intermediate data states traversed by this run in S. Moreover, in functional programming (see [2, 17]), an execution of a program may consist of the interleaving of explicit data transformations (numerical computation steps) and transformations of function terms (symbolic or "lazy" computation steps). The latter requires an alternative operational semantics which is defined in terms of rewriting rules. Thus, in studying the equivalence-checking problem by means of approximating models there are cases when it is suitable to deal with models of programs supplied with different semantics.

In this paper we introduce a new propositional model of computation for sequential computer programs. A distinctive feature of this model is that program runs and the results of computations are defined by means of two independent operational semantics. One of them can be regarded as an internal semantics that is used for routing runs in the control-flow graph of a program. The other one can be viewed as an observational semantics which is used for interpreting the results of a program execution. We show that some conventional models of sequential and recursive programs (Yanov schemes with input-output statements, linear recursive monadic schemes) can be embedded into our model. We consider the equivalence-checking problem for the presented model and develop a uniform approach to the design of efficient equivalence-checking algorithms.

The paper is organized as follows. In Section 2 we introduce the basic concepts of our model, the syntax and the semantics of generalized propositional sequential programs (GPSP). Both syntax and semantics of GPSPs are defined in terms of transition systems. In Section 3 we show that some known models of computation used in studying the equivalence-checking problem can be uniformly embedded into GPSP models. Finally, in Section 4 we present a uniform approach to the equivalence-checking problem for GPSP. This approach extends the criteria system techniques used in [28, 29] for designing efficient equivalence-checking algorithms.

2. Preliminaries

We begin by defining the syntax and the semantics of generalized propositional sequential programs (GPSP).

2.1. Syntax of GPSPs

Fix two finite alphabets $\mathcal{A} = \{a^1, \dots, a^N\}$, $\mathcal{B} = \{p^1, \dots, p^K\}$ and an infinite alphabet $\mathcal{P} = \{F_1, F_2, \dots\}$.

The elements of $\mathcal{A}$ are called *basic actions*. Intuitively, basic actions stand for

elementary built-in procedures. A finite sequence of basic actions is called a *basic term*. The set of all basic terms is denoted by $\mathcal{A}^*$. We write λ for the empty sequence of actions and call it the *empty term*. As usual, we denote by $|t|$ the length of a term t, and by t_1t_2 the concatenation of t_1 and t_2. We also have a basic action **stop** not included in $\mathcal{A}$ — it corresponds to the statement that terminates each computation of a program.

The elements of $\mathcal{B}$ are called *basic predicates*. Each basic predicate stands for an elementary built-in relation on program data. A tuple $\langle \sigma_1, \dots, \sigma_k \rangle$ of truth-values of basic predicates is called a *condition*. The set of all conditions is denoted by $\mathcal{C}$. We write $c_1, c_2, \dots$ for generic elements from $\mathcal{C}$. Since the set of primitive relations used in programs is finite and fixed, the internal structure of conditions is of no importance.

The elements of $\mathcal{P}$ are called *procedures*. Depending on the type of programming system (imperative or functional) whose programs are approximated by GPSP, procedures may be thought of either as points in control flow graphs of imperative programs, or as names of procedures and functions defined in recursive programs.

Definition 1. *By a* (deterministic) generalized propositional sequential program (GPSP*, for short*) *over sets* $\mathcal{A}$, $\mathcal{C}$, $\mathcal{P}$ *we mean a finite labeled transition system* $\pi = \langle \mathcal{P}_\pi, \boldsymbol{entry}, \boldsymbol{exit}, T, B \rangle$, *where*

- $\mathcal{P}_\pi$ *denotes the set of procedures used in* π*;*
- $\boldsymbol{entry}$ *is the* initial point *of the program;*
- $\boldsymbol{exit}$ *is the* terminal point *of the program;*
- $T\colon (\mathcal{P}_\pi \cup \{\boldsymbol{entry}\}) \times \mathcal{C} \to (\mathcal{P}_\pi \cup \{\boldsymbol{exit}\})$ *is a* transition function*;*
- $B\colon (\mathcal{P}_\pi \cup \{\boldsymbol{entry}\}) \times \mathcal{C} \to \mathcal{A}^*$ *is a* binding function.

A transition function represents the control flow of a program, whereas a binding function associates with each transition a block of basic actions. Given a sequence of conditions $c_1, c_2, \dots, c_{n-1}$, we say that a sequence of procedures $F_1, F_2, \dots, F_n$ is a $\langle c_1, c_2, \dots, c_{n-1} \rangle$*-trace from* F_1 *to* F_n in a program π if $F_1 \in \mathcal{P}_\pi \cup \{\textbf{entry}\}$ and $F_{i+1} = T(F_i, c_i)$, for every i, $1 \le i < n$. This means that $F_1, F_2, \dots, F_n$ is a trace routed by conditions $c_1, c_2, \dots, c_{n-1}$ in the transition system. If $F_1 = \textbf{entry}$ and $F_n = \textbf{exit}$ then the trace is called *complete*. We extend the binding function to the traces of a GPSP π by assuming that $B(F_1, c_1, c_2, \dots, c_{n-1}) = B(F_1, c_1)B(F_2, c_2)\dots B(F_{n-1}, c_{n-1})$. By the size $|\pi|$ of π we mean the number $|\mathcal{P}_\pi| + \sum_{P \in \mathcal{P}_\pi} \sum_{c \in \mathcal{C}} |B(P, c)|$.

Given a GPSP π and two procedures $F' \in \mathcal{P}_\pi \cup \{\mathbf{entry}\}$, $F'' \in \mathcal{P}_\pi \cup \{\mathbf{exit}\}$, we say that F' *refers to* F'' if there exists a trace from F' to F''. A procedure F in $\mathcal{P}_\pi$ is called

- *self-referenced* if F refers to itself;
- *marginal* if F does not refer to any self-referenced procedure in π;
- *pre-marginal* if F is non-marginal, but there exists a condition c such that $T(F,c)$ is marginal procedure;
- *terminated* if F refers to **exit**.

2.2. Dynamic frames and models

The semantics of programs is defined by means of dynamic Kripke structures (frames and models) (see [6]).

Definition 2. *A* dynamic deterministic frame *(or simply a* frame*) over the set of basic actions $\mathcal{A}$ is a triple of the form $\mathcal{F} = \langle S, s_0, Q\rangle$, where*

- *S is a non-empty set of* data states*,*
- *s_0 is the* initial state*, $s_0 \in S$,*
- *$Q\colon\ S \times \mathcal{A} \cup \{\mathbf{stop}\} \ \to\ S$ is an* updating function.

For all $a \in \mathcal{A}$, $s \in S$, the state $Q(s,a)$ is interpreted as the result of application of the action a to the data state s. The updating function Q can be naturally extended to the set $\mathcal{A}^*$ of basic terms: $Q^*(s,\lambda) = s$, $Q^*(s,ta) = Q(Q^*(s,t),a)$.
A state s'' is said to be *reachable* from s' if $s'' = Q^*(s',t)$ for some $t \in \mathcal{A}^*$ (notation: $s' \sqsubseteq s''$). We also write $s' \sqsubset s''$ if $s'' = Q^*(s',t)$ for some $t \in \mathcal{A}^*, t \neq \lambda$. If $\sqsubseteq$ is a partial order on S, then the frame $\mathcal{F}$ is called *ordered*.
Denote by $[t]_\mathcal{F}$ the state $s = Q^*(s_0,t)$ reachable from the initial state by means of a basic term t. As usual, the subscript $\mathcal{F}$ will be omitted when the frame is understood. Since we will deal only with data states reachable from the initial state, it is assumed that every state $s \in S$ is reachable from the initial state s_0, i.e., $S = \{[t] :\ t \in \mathcal{A}^*\}$.
A frame $\mathcal{F}_s = \langle S', s, Q'\rangle$ is said to be a *subframe* of $\mathcal{F} = \langle S, s_0, Q\rangle$ induced by a state $s \in S$ if $S' = \{Q^*(s,t) :\ t \in \mathcal{A}^*\}$ and Q' is the restriction of Q to S'. A frame $\mathcal{F}$ is called

- a *semigroup* if $\mathcal{F}$ can be mapped homomorphically onto every subframe $\mathcal{F}_s$,

- *universal* if $[t^1] = [t^2]$ implies $t^1 = t^2$ for every pair $t^1, t^2 \in \mathcal{A}^*$.

Taking the initial state $s_0 = [\lambda]$ for the unit, one may regard a semigroup frame $\mathcal{F}$ as a finitely generated monoid $\langle S, *\rangle$ such that $[t_1] * [t_2] = [t_1 t_2]$. In what follows we will say that the frame $\mathcal{F}$ *is associated* with this monoid. Clearly, the universal frame $\mathcal{U}$ is associated with the free monoid on $\mathcal{A}$. If $\mathcal{F}$ is an ordered semigroup frame then the unit element $[\lambda]$ is irreducible, e.g., $[\lambda] = [t_1 t_2]$ implies $t_1 = t_2 = \lambda$.

Definition 3. *A* dynamic deterministic model for GPSP *(or simply a* GPSP-model*) over the sets of basic actions $\mathcal{A}$ and conditions $\mathcal{C}$ is a triple $M_G = \langle \mathcal{F}, \mathcal{E}, \xi\rangle$ such that*

- *$\mathcal{F} = \langle S, s_0, Q\rangle$ and $\mathcal{E} = \langle R, r_0, P\rangle$ are frames over $\mathcal{A}$,*
- *$\xi\colon\ S \ \to\ \mathcal{C}$ is a* valuation function *indicating for every data state $s \in S$ a condition $c \in \mathcal{C}$ which is satisfied at s.*

2.3. Equivalence-checking problem for GPSPs

Definition 4. *Let $\pi = \langle \mathcal{P}_\pi, \boldsymbol{entry}, \boldsymbol{exit}, T, B\rangle$ be some GPSP over the sets of basic actions $\mathcal{A}$ and conditions $\mathcal{C}$, and $M_G = \langle \mathcal{F}, \mathcal{E}, \xi\rangle$ be a GPSP-model based on frames $\mathcal{F} = \langle S, s_0, Q\rangle$ and $\mathcal{E} = \langle R, r_0, P\rangle$. Then a finite or infinite sequence of quadruples*

$$\rho = (F_1, c_1, s_1, r_1), (F_2, c_2, s_2, r_2), \dots, (F_i, c_i, s_i, r_i), \dots \quad , \tag{1}$$

where for every i, $i \geq 1$, $F_i \in \mathcal{P}_\pi \cup \{\boldsymbol{entry}\}$, $c_i \in \mathcal{C}$, $s_i \in S$, $r_i \in R$, is called a run of π on M_G *if*

1. *$F_1 = \boldsymbol{entry}$, $s_1 = [\lambda]_\mathcal{F}$, $r_1 = [\lambda]_\mathcal{E}$, $c_1 = \xi(s_1)$;*
2. *for every i, $i \geq 2$, one of the following alternatives holds:*
   - *either $F_i = \boldsymbol{exit}$ and (F_i, c_i, s_i, r_i) is the last quadruple in (1),*
   - *or $F_i \neq \boldsymbol{exit}$ and*

$$\begin{aligned} &F_{i+1} = T(F_i, c_i),\\ &s_{i+1} = [B(F_1, c_1, c_2, \dots, c_i)]_\mathcal{F},\ \ r_{i+1} = [B(F_1, c_1, c_2, \dots, c_i)]_\mathcal{E},\\ &c_{i+1} = \xi(s_{i+1}).\end{aligned}$$

If ρ is finite and (F_n, c_n, s_n, r_n) is its last element, we say that ρ *terminates* having the state $r = P(r_n, \mathbf{stop}) \in R$ as the *result* of the run ρ. If ρ is an infinite sequence, we say that ρ *loops* and has no results. Since GPSPs and frames under consideration are deterministic, every program π has the unique run $\rho(\pi, M_G)$ on a given model M_G. We denote the result of $\rho(\pi, M_G)$ by $[\rho(\pi, M_G)]$, assuming that $[\rho(\pi, M_G)]$ is undefined if $\rho(\pi, M_G)$ loops.

Let π' and π'' be some GPSPs, M a GPSP-model, and $\mathcal{F}$, $\mathcal{E}$ be frames. Then π' and π'' are called

- *equivalent on* M_G ($\pi' \sim_{M_G} \pi''$, in symbols) if $[\rho(\pi', M_G)] = [\rho(\pi'', M_G)]$, i.e., either both runs $\rho(\pi', M_G)$ and $\rho(\pi'', M_G)$ loop (and hence have no results) or both of them terminate with the same state r as their results;
- *equivalent on* $\mathcal{F}$, $\mathcal{E}$ ($\pi' \sim_{\mathcal{F},\mathcal{E}} \pi''$, in symbols) if $\pi' \sim_{M_G} \pi''$ for every model $M = \langle \mathcal{F}, \mathcal{E}, \xi \rangle$ based on $\mathcal{F}$ and $\mathcal{E}$.

The *equivalence-checking problem* w.r.t. frames $\mathcal{F}$, $\mathcal{E}$ is to check, given an arbitrary pair π', π'' of GPSPs, whether $\pi' \sim_{\mathcal{F},\mathcal{E}} \pi''$ holds. When the decidability and complexity aspects of the equivalence problem are concerned, the frames $\mathcal{F}$, $\mathcal{E}$ under consideration are assumed to be effectively characterized in logic or algebraic terms.

3. *Embedding sequential and recursive programs into GPSPs*

In this section we show that the computational model of generalized propositional sequential programs is sufficiently expressive for presenting uniformly the equivalence-checking problem for various models of computer programs. We consider two models of computer programs—sequential imperative programs with multiple outputs and linear recursive programs—and demonstrate the embedding of these models into GPSPs.

3.1. Sequential programs with multiple outputs

As was observed in Section 1, if more than one output statements are executed along a run of a sequential program, the result of computation is specified not by the final data state when the program terminates, but by the sequence of data states reached by the program after successive execution of output statements. Formally, the equivalence-checking problem for this class of programs can be defined as follows.

Suppose that the set of basic actions is split into disjoint sets $\mathcal{A}_1$ and $\mathcal{A}_2$. The actions from $\mathcal{A}_1$ are used just to output current intermediate results, whereas those from $\mathcal{A}_2$ are conventional actions whose execution is invisible to the external observer.

Definition 5. *A deterministic propositional sequential program (PSP) over sets $\mathcal{A}$, $\mathcal{C}$, $\mathcal{P}$ is a finite labeled transition system $\pi = \langle \mathcal{P}_\pi, \boldsymbol{entry}, \boldsymbol{exit}, T, B \rangle$, where*

- *$\mathcal{P}_\pi$ denotes the set of program points in π;*
- ***entry** is the* initial point *of the program;*
- ***exit** is the* terminal point *of the program;*
- *$T\colon (\mathcal{P}_\pi \cup \{\boldsymbol{entry}\}) \times \mathcal{C} \to (\mathcal{P}_\pi \cup \{\boldsymbol{exit}\})$ is a* transition function*;*
- *$B\colon \mathcal{P}_\pi \cup \{\boldsymbol{entry}\} \to \mathcal{A}$ is a* binding function.

Let $\mathcal{F} = \langle S, s_0, Q \rangle$ be a frame and ξ a valuation function on $\mathcal{F}$. Then the run of PSP π on the PSP-model $M = \langle \mathcal{F}, \xi \rangle$ is a finite or infinite sequence of triples

$$\rho = (F_1, c_1, s_1), (F_2, c_2, s_2), \dots, (F_i, c_i, s_i), \dots \quad , \tag{2}$$

such that

1. $F_i \in \mathcal{P}_\pi \cup \{\mathbf{entry}\},\ c_i \in \mathcal{C},\ s_i \in S$ for every $i,\ i \geq 1$,
2. $F_1 = \mathbf{entry}$, $c_1 = \xi(s_0)$, $s_1 = [\lambda]_{\mathcal{F}}$;
3. for every $i,\ i \geq 2$, one of the following alternatives holds:
   - either $F_i = \mathbf{exit}$ and (F_i, c_i, s_i) is the last triple in (2),
   - or $T(F_i, c_i) \neq \mathbf{exit}$ and
     $$F_{i+1} = T(F_i, c_i),\ s_{i+1} = Q(s_i, B(F_i)),\ c_{i+1} = \xi(s_{i+1}),$$

If ρ is finite and (F_n, c_n, s_n) is its last element, we say that ρ *terminates*. The result $r(\pi, M)$ of a terminating run ρ of a PSP π on a model M is defined as follows. Let $i_1, i_2, \dots, i_k$ be the sequence of all indices such that $B(F_{i_j}) \in \mathcal{A}_1$. Then $r(\pi, M) = \langle s_{i_1}, s_{i_2} \dots, s_{i_k} \rangle$. If ρ does not terminate then the result $r(\pi, M)$ is undefined.

The equivalence of PSPs π' and π'' on models and frames is defined analogously to that of GPSPs.

Now we show that the equivalence-checking problem for PSPs with multiple outputs can be reduced to the equivalence-checking problem for GPSPs.

With every PSP $\pi = \langle \mathcal{P}_\pi, \mathbf{entry}, \mathbf{exit}, T, B \rangle$ we associate a GPSP $\widehat{\pi} = \langle \mathcal{P}_\pi, \mathbf{entry}, \mathbf{exit}, T, \widehat{B} \rangle$ such that $\widehat{B}(F, c) = B(F)$ for every procedure $F \in \mathcal{P}_\pi$ and every condition $c \in \mathcal{C}$.

Given a PSP-model $M = \langle \mathcal{F}, \xi \rangle$ based on the frame $\mathcal{F} = \langle S, s_0, Q \rangle$, we consider a GPSP-model $M_G = \langle \mathcal{F}, \mathcal{E}, \xi \rangle$, where the frame $\mathcal{E} = \langle R, r_0, P \rangle$ is defined as follows:

1. R is the set of all finite sequences $\langle s_1, s_2, \ldots, s_k \rangle$ of data states from $\mathcal{F}$;
2. the initial state r_0 is the sequence $\langle s_0 \rangle$;
3. for every state $r = \langle s_1, \ldots, s_k \rangle$ in R and every action a in $\mathcal{A} \cup \{\mathbf{stop}\}$
   (a) $P(r, a) = \langle s_1, s_2, \ldots, s_{k-1}, Q(s_k, a), Q(s_k, a) \rangle$ if $a \in \mathcal{A}_1$;
   (b) $P(r, a) = \langle s_1, s_2, \ldots, s_{k-1}, Q(s_k, a) \rangle$ if $a \in \mathcal{A}_2$;
   (c) $P(r, \mathbf{stop}) = \langle s_1, s_2, \ldots, s_{k-1}, s_0 \rangle$.

The following theorem shows that PSPs with multiple outputs can be embedded into GPSPs:

Theorem 1. *Let π_1 and π_2 be PSPs and $M = \langle \mathcal{F}, \xi \rangle$ a PSP model. Consider the GPSPs $\widehat{\pi}_1, \widehat{\pi}_2$ and the GPSP-model $M_G = \langle \mathcal{F}, \mathcal{E}, \xi \rangle$ as defined above. Then*

1. $\pi_1 \sim_M \pi_2 \iff \widehat{\pi}_1 \sim_{M_G} \widehat{\pi}_2$;

2. $\pi_1 \sim_{\mathcal{F}} \pi_2 \iff \widehat{\pi}_1 \sim_{\mathcal{F},\mathcal{E}} \widehat{\pi}_2$.

Moreover, if $\mathcal{F}$ is a semigroup frame then $\mathcal{E}$ is a semigroup frame as well.

Thus, the equivalence-checking problem for sequential programs with multiple outputs can be reduced to the equivalence-checking problem for GPSPs without loss of specific algebraic features of semantics.

3.2. Linear recursive programs

Let $\mathcal{A}$ and $\mathcal{P}$ be the sets of basic actions and procedures, respectively. By a *term* we mean any finite sequence of basic actions and procedures. A term t is called *linear* if at most one procedure occurs in t and the rightmost element of t is a basic action. The set of all linear terms over $\mathcal{A} \cup \mathcal{P}$ is denoted by $LinTerm$. We write $F \in t$ to indicate that a procedure F occurs in t. If $t = a_1 a_2 \ldots a_n$ then the term $a_n \ldots a_2 a_1$ is called the *reverse* of t and denoted by t^{-1}.

A *definition* of a procedure F is an expression D of the form

$$\begin{array}{lll} F & = & \mathbf{if}\ c^1\ \mathbf{then}\ t_1\ \mathbf{else} \\ & & \quad \mathbf{if}\ c^2\ \mathbf{then}\ t_2\ \mathbf{else} \\ & & \quad \ldots \\ & & \qquad \mathbf{if}\ c^{I-1}\ \mathbf{then}\ t_{I-1}\ \mathbf{else}\ t_I \end{array}$$

where $t_i \in LinTerm$, $c^i \in \mathcal{C} = \{c^1, c^2, \ldots, c^I\}$, $1 \le i \le I$. The definition above will be also written as

$$F\colon (c^1, t_1), (c^2, t_2), \ldots, (c^I, t_I) \ . \qquad (3)$$

The first occurrence of F in D is called the *head* of D, and the list of pairs $(c^1, t_1), (c^2, t_2), \ldots, (c^I, t_I)$ is the *body* of D. For every pair (c^i, t_i) in the body of D, the term t_i is called a *c^i-variant* of the definition D.

Definition 6. *A* (deterministic) linear recursive program (LRP) *over the sets $\mathcal{A}$, $\mathcal{C}$, $\mathcal{P}$ is a tuple $\pi = \langle G, D_1, D_2, \ldots, D_n \rangle$, where*

- *$G \in LinTerm$ is the* goal *of the program;*
- *$D_1, D_2, \ldots, D_n$ are definitions of pairwise different procedures $F_1, \ldots, F_n$.*

The set of procedures $\{F_1, \ldots, F_n\}$ defined in an LRP π is denoted by $\mathcal{P}_\pi$. Given a procedure F in $\mathcal{P}_\pi$, we write $D_\pi(F)$ for the definition of F in π, and $D_\pi(F, c)$ for the c-variant of $D_\pi(F)$. If a program is understood, the subscript π will be omitted. It is also assumed that every procedure occurring in π is defined in π.

The semantics of LRPs is defined by means of dynamic frames and models.

Definition 7. *Let $\pi = \langle G, D_1, D_2, \ldots, D_n \rangle$ be some LRP and $M = \langle \mathcal{F}, \xi \rangle$ a model based on a frame $\mathcal{F} = \langle S, s_0, Q \rangle$. A finite or infinite sequence of triples*

$$\rho = (t_1, s_1, c_1), (t_2, s_2, c_2) \ldots, (t_i, s_i, c_i), \ldots \ , \qquad (4)$$

where $t_i \in LinTerm$, $s_i \in S$, $c_i \in \mathcal{C}$, $i \ge 1$, is called a run of π on M *if $t_1 = G$ and for every i, $i \ge 1$, one of the following conditions holds:*

1. if t_i is a basic term then $s_i = Q^*(s_{i-1}, t_i^{-1})$, $c_i = \xi(s_i)$, and the triple (t_i, s_i, c_i) is the last element of (2);

2. if t_i is a non-basic term of the form $t_i = TFt$, where $F \in \mathcal{P}_\pi$, $t \in \mathcal{A}^*$, then $s_i = Q^*(s_{i-1}, t^{-1})$, $c_i = \xi(s_i)$, $t_{i+1} = TD(F, c_i)$.

If ρ is finite and the triple (s_m, c_m, t_m) is its last element, we say that ρ *terminates* with *result* s_m. If ρ is an infinite sequence, we say that ρ *loops*. Since LRPs and frames under consideration are deterministic, every program π has a unique run $\rho(\pi, M)$ on a given model M. We denote by $[\rho(\pi, M)]$ the result of $\rho(\pi, M)$, assuming that $[\rho(\pi, M)]$ is undefined if $\rho(\pi, M)$ loops.

The equivalence of LRPs π' and π'' on models and frames is defined similarly to that of GPSPs.

Now we show that the equivalence-checking problem for LRPs can be reduced to the equivalence-checking problem for GPSPs.

First, we define a translation from LRPs into GPSPs. Given the set $\mathcal{A}$ of basic actions for LRPs, we introduce a set $\overline{\mathcal{A}}$ of basic actions for GPSPs by taking $\overline{\mathcal{A}} = \{\langle\lambda, a\rangle : a \in \mathcal{A}\} \cup \{\langle a, \lambda\rangle : a \in \mathcal{A}\}$. For any pair of basic terms $t_1 = a_1, \dots, a_k$ and $t_2 = a'_1, \dots, a'_m$ over $\mathcal{A}$ we denote by $\langle t_2, t_1\rangle$ the term $\langle\lambda, a_1\rangle \dots \langle\lambda, a_k\rangle\langle a'_1, \lambda\rangle \dots \langle a'_m, \lambda\rangle$.

Let $\pi = \langle G, D_1, D_2, \dots, D_n\rangle$ be an LRP over $\mathcal{A}$, $\mathcal{C}$, and $\mathcal{P}$. The corresponding GPSP $\overline{\pi} = \langle \mathcal{P}_{\overline{\pi}}, \mathbf{entry}, \mathbf{exit}, T, B\rangle$ is defined as follows:

1. $\mathcal{P}_{\overline{\pi}} = \mathcal{P}_\pi, \quad \mathbf{entry} = G$;
2. for every procedure $f \in \mathcal{P}_\pi$ and every condition $c \in \mathcal{C}$,
   (a) if $D_\pi(F, c) = t$, where t is a basic term in $\mathcal{A}^*$, then $T(F, c) = \mathbf{exit}$ and $B(F, c) = \langle\lambda, t\rangle$;
   (b) if $D_\pi(F, c) = t'F't$, where t, t' are basic terms in $\mathcal{A}^*$ and F' is a procedure in $\mathcal{P}_\pi$, then $T(F, c) = F'$ and $B(F, c) = \langle t', t\rangle$.

Next we relate dynamic models for LRP with GPSP-models. Let $M = \langle \mathcal{F}, \xi\rangle$ be a dynamic model over the set of basic actions $\mathcal{A}$ and conditions $\mathcal{C}$. Then the corresponding GPSP-model $M_G = \langle \mathcal{F}, \mathcal{E}, \xi\rangle$ is obtained from M by adopting the updating function Q to the basic actions from $\overline{\mathcal{A}}$

$$Q(s, \langle t', t\rangle) = Q(s, t)$$

and by adding to M a frame $\mathcal{E} = \langle R, r_0, P\rangle$ such that

1. $R = (\mathcal{A} \cup \{\mathbf{stop}\}) \times S$;
2. $r_0 = \langle\lambda, s_0\rangle$ is the initial state;
3. the updating function $P\colon R \times (\overline{\mathcal{A}} \cup \{\mathbf{stop}\}) \to R$ is defined for each $r = \langle t, s\rangle$ in R by the following equalities:
   (a) $P(r, \mathbf{stop}) = \langle \mathbf{stop}, Q^*(s, t)\rangle$;
   (b) $P(r, \langle a, \lambda\rangle) = \langle at, s\rangle$;
   (c) $P(r, \langle\lambda, a\rangle) = \langle t, Q(s, a)\rangle$.

The following theorem shows that LRP can be embedded into GPSPs:

Theorem 2. *Let π_1 and π_2 be LRPs, and $M = \langle \mathcal{F}, \xi\rangle$ a PSP model. Consider the GPSPs $\overline{\pi}_1, \overline{\pi}_2$ and the GPSP model $M_G = \langle \mathcal{F}, \mathcal{E}, \xi\rangle$ as defined above. Then*

1. $\pi_1 \sim_M \pi_2 \iff \overline{\pi}_1 \sim_{M_G} \overline{\pi}_2$;

2. $\pi_1 \sim_{\mathcal{F}} \pi_2 \iff \overline{\pi}_1 \sim_{\mathcal{F},\mathcal{E}} \overline{\pi}_2$.

Moreover, if $\mathcal{F}$ is a semigroup frame then $\mathcal{E}$ is a semigroup frame as well.

Thus, the equivalence-checking problem for linear recursive programs can be reduced to the equivalence-checking problem for GPSPs without loss of specific semantic features.

4. How to design a polynomial time equivalence-checking algorithms for GPSPs

In this section we present an approach to the design of efficient equivalence-checking algorithms for GPSPs w.r.t. some ordered semigroup frames. Its key idea is as follows. Given frames $\mathcal{F}$, $\mathcal{E}$ and a pair of programs π^1, π^2, we first choose some specific semigroups U and V to encode all pairs of states $\langle s', s''\rangle$ in $\mathcal{F}$ and $\langle r', r''\rangle$ in $\mathcal{E}$. This encoding is intended to estimate the extent to which the intermediate data states of program runs "diverge" to that moment. Then, using this encoding, we construct a graph structure Γ_{π^1,π^2} to represent all pairs of runs $\rho(\pi^1, M)$, $\rho(\pi^2, M)$ of programs π^1, π^2 on the models based on the frames $\mathcal{F}$ and $\mathcal{E}$. We show that to check the equivalence of π^1 and π^2 we only need to analyze a fragment of Γ_{π^1,π^2} whose size is polynomial in $|\pi^1|$ and $|\pi^2|$. The construction of Γ_{π^1,π^2} involves solutions to the reachability problem "$s' \sqsubseteq s''$?" for the frame $\mathcal{F}$ and the identity problem "$w' = w''$?" for the semigroups U and V. If these problems are decidable in polynomial time, the equivalence-checking problem for GPSPs w.r.t. $\mathcal{F}$, $\mathcal{E}$ is decidable in polynomial time as well. Using this technique, we demonstrate that the equivalence-checking problem for LRPs w.r.t. the frames associated with free commutative monoids is decidable in polynomial time.

Suppose that U is a finitely generated monoid, and u^+, u^* are the distinguished elements in U. Denote by $\circ$ and e the binary operation on U and the unit of U, respectively.

Definition 8. *The triple $K = \langle U, u^+, u^*\rangle$ is said to be a* criteria system *for a semigroup frame $\mathcal{F} = \langle S, s_0, Q\rangle$ if K and $\mathcal{F}$ meet the following requirements:*

(R1) there exists a homomorphism φ of $S \times S$ into U such that

$$[t_1] = [t_2] \iff u^+ \circ \varphi(\langle [t_1], [t_2]\rangle) \circ u^* = e$$

holds for every pair $t_1,\ t_2$ in $\mathcal{A}^*$,

(R2) for every element u in $U \circ u^*$ the equation $X \circ u = e$ has at most one solution X in the coset $u^+ \circ U$.

Let $\mathcal{F} = \langle S, s_0, Q\rangle$ and $\mathcal{E} = \langle R, r_0, P\rangle$ be semigroup frames, and $\mathcal{F}$ an ordered frame. Suppose that $K_{\mathcal{F}} = \langle U, u^+, u^*\rangle$ and $K_{\mathcal{E}} = \langle V, v^+, v^*\rangle$ are criteria systems for these frames such that $\varphi\colon S \times S \to U$ and $\psi\colon R \times R \to V$ are the required homomorphisms. We assume that the coset $u^+ \circ U$ is divided into four disjoint sets $U_= = \{u^+ \circ \varphi(\langle s, s\rangle)\}$, $U_< = \{u^+ \circ \varphi(\langle s', s''\rangle) : s' \sqsubset s''\}$, $U_> = \{u^+ \circ \varphi(\langle s', s''\rangle) : s'' \sqsubset s'\}$ and $U_\emptyset = (u^+ \circ U) - (U_= \cup U_< \cup U_>)$. Since $\mathcal{F}$ is an ordered semigroup frame, checking reachabilities $s' \sqsubseteq s''$ and $s'' \sqsubseteq s'$ would suffice to decide which of these classes contains $u^+ \circ \varphi(\langle s', s''\rangle)$.

Given a pair of GPSPs π^1, π^2 such that $\mathcal{P}_{\pi^1} \cap \mathcal{P}_{\pi^2} = \emptyset$, we define a rooted labeled directed graph Γ_{π^1,π^2} as follows.

The nodes of Γ_{π^1,π^2} are quadruples (F_1, F_2, u, v) such that F_1 and F_2 are procedures from π^1 and π^2, respectively, and u, v are elements from the cosets $u^+ \circ U$ and $v^+ \circ V$, respectively.

The root of Γ_{π^1,π^2} is the node $w_0 = (\mathbf{entry}, \mathbf{entry}, u^+, v^+)$.

The arcs of Γ_{π^1,π^2} are marked with pairs (c_1, c_2) in $(\mathcal{C} \cup \{\varepsilon\}) \times (\mathcal{C} \cup \{\varepsilon\})$. The arcs connect nodes in Γ_{π^1,π^2} according to the following rules. Consider an arbitrary node $w = (F_1, F_2, u, v)$ in Γ_{π^1,π^2}.

1. If $u \in U_\emptyset$ and $F_1, F_2 \neq \mathbf{exit}$ then for every pair $(c_1, c_2) \in \mathcal{C} \times \mathcal{C}$ the arc marked with (c_1, c_2) leads from w to $w' = (F_1', F_2', u', v')$ such that $F_1' = T_{\pi^1}(F_1, c_1)$, $F_2' = T_{\pi^2}(F_2, c_2)$, $u' = u \circ \varphi(\langle [t_1]_\mathcal{F}, [t_2]_\mathcal{F}\rangle)$, $v' = v \circ \psi(\langle [t_1]_\mathcal{E}, [t_2]_\mathcal{E}\rangle)$, where $t_1 = B_{\pi^1}(F_1, c_1)$ and $t_2 = B_{\pi^2}(F_2, c_2)$].
2. If $u \in U_>$ or $F_1 = \mathbf{exit}$ then for every $c \in \mathcal{C}$ the arc marked with (ε, c) leads from w to the node $w' = (F, F_2', u', v')$ such that $F_2' = T_{\pi^2}(F_2, c)$, $u' = u \circ \varphi(\langle [\lambda]_\mathcal{F}, [t_2]_\mathcal{F}\rangle)$, $v' = v \circ \psi(\langle [\lambda]_\mathcal{E}, [t_2]_\mathcal{E}\rangle)$, where $t_2 = B_{\pi^2}(F_2, c)$].
3. If $u \in U_<$ or $F_2 = \mathbf{exit}$ then for every $c \in \mathcal{C}$ the arc marked with (c, ε) leads from w to the node $w' = (F_1', F_2, u', v')$ such that $F_1' = T_{\pi^1}(F_1, c)$, $u' = u \circ \varphi(\langle [t_1]_\mathcal{F}, [\lambda]_\mathcal{F}\rangle)$, $v' = v \circ \psi(\langle [t_1]_\mathcal{E}, [\lambda]_\mathcal{E}\rangle)$, where $t_1 = B_{\pi^1}(F_1, c)$].
4. If $u \in U_=$ and $F_1, F_2 \neq \mathbf{exit}$ then for every $c \in \mathcal{C}$ the arc marked with (c, c) leads from w to the node $w' = (F_1', F_2', u', v')$ such $F_1' = T_{\pi^1}(F_1, c)$, $F_2' = T_{\pi^2}(F_2, c)$, $u' = u \circ \varphi(\langle [t_1]_\mathcal{F}, [t_2]_\mathcal{F}\rangle)$, $v' = v \circ \psi(\langle [t_1]_\mathcal{E}, [t_2]_\mathcal{E}\rangle)$, where $t_1 = B_{\pi^1}(F_1, c)$ and $t_2 = B_{\pi^2}(F_2, c)$].

The directed paths in Γ_{π^1,π^2} encode all possible pairs of runs $r(\pi^1, M)$, $r(\pi^2, M)$ of the GPSPs π^1 and π^2 on the models M based on the frames $\mathcal{F}$ and $\mathcal{E}$. The main characteristic feature of the graph Γ_{π^1,π^2} is presented in the following lemma:

Lemma 1. *Suppose $w_0, w_1, \ldots, w_m$, $m \geq 1$, is a finite sequence of nodes in Γ_{π^1,π^2} such that w_0 is the root of Γ and $w_i = (F_1^i, F_2^i, u^i, v^i)$, $1 \leq i \leq m$. Then this sequence of nodes forms a path*

$$w_0 \xrightarrow{(c_1^1, c_2^1)} w_1 \xrightarrow{(c_1^2, c_2^2)} \ldots \xrightarrow{(c_1^m, c_2^m)} w_m$$

in Γ_{π^1,π^2} iff there exists a GPSP-model M based on the frames $\mathcal{F}$ and $\mathcal{E}$ such that each of the runs $r(\pi^i, M)$, $i = 1, 2$, of the programs π^1 and π^2 has a prefix of the form

$$(\boldsymbol{entry}, c_i^{j_1^i}, s_i^{j_1^i}, r_i^{j_1^i}), (F_i^1, c_i^{j_2^i}, s_i^{j_2^i}, r_i^{j_2^i}), \ldots, (F_i^k, c_i^{j_k^i}, s_i^{j_k^i}, r_i^{j_k^i}),$$

where $c_i^{j_1^i}, c_i^{j_2^i}, \ldots, c_i^{j_k^i}$ is the subsequence of all those elements in $c_i^1, c_i^2, \ldots, c_i^m$ that are different from ε. Moreover, for every j, $1 \leq j \leq m$, these prefixes satisfy the following requirements

$$\begin{array}{rcl} u^j & = & u^+ \circ \varphi(\langle s_1^{l_j^1}, s_2^{l_j^2}\rangle), \\ v^j & = & v^+ \circ \psi(\langle r_1^{l_j^1}, r_2^{l_j^2}\rangle), \end{array}$$

where $l_j^i = \max\{l : l \leq j,\ l \in \{j_1^i, j_2^i, \ldots, j_k^i\}\}$, $i = 1, 2$.

Proof. By induction on m, using the definition of Γ_{π^1,π^2}. $\square$

A node w in Γ_{π^1,π^2} is said to be a *0-rejecting* node if it satisfies one of the following conditions:

1. $w = (\mathbf{exit}, \mathbf{exit}, u, v)$ and $v \circ v^* \neq e$;
2. $w = (F^1, F^2, u, v)$ is such that one of the procedures F^1, F^2 is marginal, whereas the other one is a non-marginal.

Clearly, given a decision procedure for the identity problem "$v' = v''$?" on V it is easy to check whether w is a 0-rejecting node.

A node w_0 in Γ_{π^1,π^2} is said to be a *1-rejecting* node if there exists an infinite path $w_0, w_1, \ldots, w_n, \ldots$ in Γ_{π^1,π^2} which starts at w_0 and satisfies one of the following conditions:

1. almost all nodes $w_n = (F^1, F^2, u, v)$ in this path are such that $u \in U_<$ and F^2 is terminated procedure;
2. almost all nodes $w_n = (F^1, F^2, u, v)$ in this path are such that $u \in U_>$ and F^1 is terminated procedure.

Lemma 2. $\pi^1 \sim_{\mathcal{F},\mathcal{E}} \pi^2$ *iff no rejecting nodes are accessible from the root* Γ_{π^1,π^2}.

Proof. Follows from Lemma 1 and requirement $(R1)$. If a 1-rejecting node is accessible from the root of Γ_{π^1,π^2} then there is a GPSP-model M such that one of the runs $\rho(\pi^1, M)$, $\rho(\pi^2, M)$ terminates, whereas the other loops. If a node $w = (F^1, F^2, u, v)$ is accessible from the root of Γ_{π^1,π^2} and one of the procedures, say F^1, is marginal, whereas the other (F^2) is a non-marginal then there is a GPSP-model M such that the run $\rho(\pi^1, M)$ terminates and the run $\rho(\pi^2, M)$ loops. If a node $w = (\mathbf{exit}, \mathbf{exit}, u, v)$ is accessible from the root of Γ_{π^1,π^2} and $v \circ v^* \neq e$ then there is a GPSP-model M such that both runs $\rho(\pi^1, M)$, $\rho(\pi^2, M)$ terminate but $[\rho(\pi^1, M)] \neq [\rho(\pi^2, M)]$. □

Lemma 3. *Suppose that both procedures F_1, F_2 are terminated and two different nodes $w' = (F_1, F_2, u, v')$, $w'' = (F_1, F_2, u, v'')$ are accessible from the root of Γ_{π^1,π^2}. Suppose also that neither w', nor w'' is a 1-rejecting node. Then some 0-rejecting node is accessible from the root of Γ_{π^1,π^2}.*

Proof. By Lemma 1, each path in Γ_{π^1,π^2} is associated with the pair of (prefixes of) runs $\rho(\pi^1, M)$, $\rho(\pi^2, M)$. Since F_1 is a terminated procedure and $\mathcal{F}$ is an ordered frame, we may assume that $\rho(\pi^1, M)$ terminates. Since w' and w'' are not 1-rejecting nodes, this means that two different nodes of the form $w'_1 = (\mathbf{exit}, G_2, u_1, v'_1)$ and $w''_1 = (\mathbf{exit}, G_2, u_1, v''_1)$ are accessible from w' and w'' respectively. The requirement (R2) of the criteria system $K_{\mathcal{E}}$ guarantees that $v'_1 \neq v''_1$. If G_2 is non-marginal then each of the nodes w'_1 and w''_1 is 0-rejecting. Otherwise, by applying Lemma 1, we may assume that $\rho(\pi^2, M)$ also terminates. Then two different nodes of the form $w'_2 = (\mathbf{exit}, \mathbf{exit}, u_2, v'_2)$ and $w''_2 = (\mathbf{exit}, \mathbf{exit}, u_2, v''_2)$ are reachable from w'_1 and w''_1. But, by the requirement $(R2)$ of the criteria system $K_{\mathcal{E}}$, at most one of the elements w'_2, w''_2 may be equal to e. Hence, at least one of the nodes w'_2, w''_2 is 0-rejecting. □

Lemma 4. *Suppose both procedures F_1, F_2 are pre-marginal and the node $w = (F_1, F_2, u, v)$ is accessible from the root of Γ_{π^1,π^2}. Suppose also that $u \notin U_=$ and w is not a 1-rejecting node. Then some 0-rejecting node is accessible from the root of Γ_{π^1,π^2}.*

Proof. If F_1, F_2 are pre-marginal nodes and $u \notin U_=$ then we may find a pair (c_1, c_2) of conditions such that the arc marked with (c_1, c_2) leads from w to a node $w' = (F'_1, F'_2, u', v')$, where one of the procedures F'_1, F'_2 is marginal, whereas the other is non-marginal. □

Lemma 5. *Let $N = (\max(|\pi_1|, |\pi_2|))^2 + 1$, and F_1, F_2 be a pair of procedures such that one of them is non-marginal, whereas the other is terminated. Suppose that at least N pairwise different nodes $w_1 = (F_1, F_2, u^1, v^1), \ldots, w_N = (F_1, F_2, u^N, v^N)$ are accessible from the root of Γ_{π^1,π^2} and all these nodes are not 1-rejecting. Then some 0-rejecting node is accessible from the root of Γ_{π^1,π^2}.*

Proof. If exactly one of the procedures F_1, F_2 is non-terminated or marginal then, by Lemma 1, a 0-rejecting node is accessible from any w_i. If $u^i = u^j$ holds for some pair i, j, then $v^i \neq v^j$ and, hence, by Lemma 3, some 0-rejecting node

is also accessible from the root. Thus, it suffices to consider the case when (1) all elements $u_1, \ldots, u_N$ are pairwise different and (2) both procedures F_1, F_2 are non-marginal and terminated. It follows from (2) that from any node w_i it is possible to reach a node $w'_i = (F'_1, F'_2, u'_i, v'_i)$ such that $u'_i \in U_\emptyset \cup U_=$ and one of the procedures, say F'_1, is pre-marginal. If F'_2 is not pre-marginal then at least one of the successors of w'_i in Γ_{π^1,π^2} is a 0-rejecting node. Suppose that both F'_1 and F'_2 are pre-marginal. Then a consequence of (1) and the requirement $(R2)$ for criteria system $K_{\mathcal{F}_1}$ is the fact that a node of the form $w'_j = (F'_1, F'_2, u'_j, v'_j)$ is also reachable from another node w_j (where $i \neq j$), and, moreover, $u'_i \neq u'_j$. By the requirement $(R2)$ of the criteria system $K_{\mathcal{F}}$, at most one of the element u'_i, u'_j is in $U_=$. Hence, by Lemma 4, a 0-rejecting node is accessible from the root of Γ_{π^1,π^2}. □

Lemma 6. *Let $N = (\max(|\pi_1|, |\pi_2|))^2 + 1$, and F_1, F_2 be a pair of marginal procedures. Suppose that $N+1$ pairwise different nodes $w_0 = (F_1, F_2, u_0, v_0), w_1 = (F_1, F_2, u_1, v_1), \ldots, w_N = (F_1, F_2, u_N, v_N)$ are accessible from the root of Γ_{π^1,π^2} and $v_0 \neq v_i$ for all i, $1 \leq i \leq N$. Then some 0-rejecting node is accessible from the root of Γ_{π^1,π^2}.*

Proof. By combining the arguments used in the proofs of Lemmas 4,5. □

Lemma 7. *Let $N = (\max(|\pi_1|, |\pi_2|))^2 + 1$, and F_1, F_2 be a pair of marginal procedures. Suppose that $N+1$ pairwise different nodes $w_0 = (F_1, F_2, u_1, v), w_1 = (F_1, F_2, u_1, v), \ldots, w_N = (F_1, F_2, u_N, v)$ are accessible from the root of Γ_{π^1,π^2}. Then a 0-rejecting node is accessible from w_0 only if a 0-rejecting node is accessible from some w_i, $1 \leq i \leq N$.*

Proof. By combining the arguments used in the proofs of Lemmas 4,5. □

Theorem 3. *Suppose that $\mathcal{F} = \langle S, s_0, Q\rangle$ and $\mathcal{E} = \langle R, r_0, P\rangle$ are semigroup frames, and $K_{\mathcal{F}} = \langle U, u^+, u^*\rangle$ and $K_{\mathcal{E}} = \langle V, v^+, v^*\rangle$ are criteria systems for these frames such that the identity problem "$x = y$?" is decidable in both semigroups U and V in time $\tau_1(n)$. Suppose also that $\mathcal{F}$ is in addition an ordered frame such that the reachability problem "$[t'] \sqsubseteq [t'']$?" is decidable in time $\tau_2(n)$. Then the equivalence-checking problem "$\pi^1 \sim_{\mathcal{F},\mathcal{E}} \pi^2$?" is decidable in time $O(n^6(\tau_1(O(n^4)) + \tau_2(O(n^4))))$, where $n = \max(|\pi_1|, |\pi_2|)$.*

Proof. Let π^1 and π^2 be GPSPs, and $n = \max(|\pi_1|, |\pi_2|)$. By Lemma 2, the equivalence-checking problem for π^1 and π^2 is reduced to the accessibility-checking of rejecting nodes in Γ_{π^1,π^2}. Consider an arbitrary pair of procedures $F_1 \in \mathcal{P}_{\pi^1}$ and $F_2 \in \mathcal{P}_{\pi^2}$. We will show that to check the accessibility of a rejecting node from the root of Γ_{π^1,π^2} it suffices to analyze only a bounded number of the nodes (F_1, F_2, u, v) for every pair of procedures $F_1,\ F_2$.

If both procedures $F_1,\ F_2$ are non-terminated then it is clear that no rejecting nodes are accessible from any node of the form (F_1, F_2, u, v).

If one of the procedures $F_1,\ F_2$ is terminated, whereas the other is non-terminated, then Lemma 1 guarantees that some rejecting node is accessible from any node of the form (F_1, F_2, u, v).

Now consider the case when one of the procedures $F_1,\ F_2$ is non-marginal and the other is terminated. As evidenced by Lemmas 3–5, if n^2+1 nodes of the form $w = (F_1, F_2, u, v)$ are accessible from the root of Γ_{π^1,π^2} then either one of these nodes is 1-rejecting, or some 0-rejecting node is accessible from the root of Γ_{π^1,π^2}.

Finally, suppose that both procedures $F_1,\ F_2$ are marginal. Then, by Lemmas 6 and 7, it suffices to consider only $2n^2+1$ nodes of the form (F_1, F_2, u, v) to check the accessibility of any rejecting node via some node of the form (F_1, F_2, u, v).

Thus, to check the equivalence $\pi^1 \sim_{\mathcal{F},\mathcal{E}} \pi^2$ one need only to check the rooted fragment of Γ_{π^1,π^2} which includes at most $2n^4+n^2$ nodes. When constructing such a rooted fragment of size m we are forced to check inequalities $[t'] \sqsubseteq [t'']$ and identities $u^+ \circ \varphi(\langle [t'_1], [t'_2] \rangle) = u^+ \circ \varphi(\langle [t''_1], [t''_2] \rangle)$, $v^+ \circ \psi(\langle [t'_1], [t'_2] \rangle) = v^+ \circ \psi(\langle [t''_1], [t''_2] \rangle)$, where the size of terms $t', t'', t'_1, t'_2, t''_1, t''_2$ is $O(m)$. $\square$

To demonstrate the use of Theorem 3, we consider the equivalence-checking problem for linear recursive programs w.r.t. commutative frames. Let $\mathcal{F}_{fc}$ be a frame associated with a free commutative monoid. Suppose $\mathcal{A} = \{a^1, \dots, a^N\}$ and denote by Z a free Abelian group of range N generated by some elements $q_1, \dots, q_N$. Then $K = \langle Z, Z, e, e \rangle$ is a criteria system for $\mathcal{F}_{fc}$, assuming $\varphi(\langle [a_i], [\lambda] \rangle) = q_i$ and $\varphi(\langle [\lambda], [a_j] \rangle) = q_j^{-1}$ for every pair of actions a_i, a_j. It should be noted that the reachability problem in $\mathcal{F}_{fc}$ and the identity problem in Z are decidable in linear time. Hence, by Theorem 3 the equivalence-checking problem for GPSPs w.r.t. $\mathcal{F}_{fc}$ is decidable in polynomial time.

As in Section 3, given the set $\mathcal{A}$ of basic actions for LRPs, we introduce the set $\overline{\mathcal{A}}$ of basic actions for GPSPs: $\overline{\mathcal{A}} = \{\langle \lambda, a \rangle : a \in \mathcal{A}\} \cup \{\langle a, \lambda \rangle : a \in \mathcal{A}\}$ and translate every LRP π into GPSP $\overline{\pi}$. Given a free commutative frame $\mathcal{F} = \langle S, s_0, Q \rangle$, we introduce a pair of frames $\mathcal{F} = \langle S, s_0, Q_1 \rangle$,$\mathcal{E} = \langle S, s_0, Q_2 \rangle$ such that

$$\begin{array}{rcl} Q_1(s, \langle \lambda, a \rangle) & = & Q(s, a); \\ Q_1(s, \langle a, \lambda, \rangle) & = & s; \\ Q_2(s, \langle \lambda, a \rangle) & = & Q(s, a); \\ Q_1(s, \langle a, \lambda, \rangle) & = & Q(s, a). \end{array}$$

Theorem 4. *Let π_1 and π_2 be a pair of LRPs, and a frame $\mathcal{F}_{fc}$ is associated with a free commutative frame. Then the frames $\mathcal{F}, \mathcal{E}$ defined above are associated also with free commutative monoids and*

$$\pi_1 \sim_{\mathcal{F}} \pi_2 \iff \overline{\pi}_1 \sim_{\mathcal{F},\mathcal{E}} \overline{\pi}_2.$$

By combining Theorems 3 and 4 we arrive at

Corollary 1. *The equivalence-checking problem for linear recursive programs w.r.t. frames associated with free commutative monoids is decidable in polynomial time.*

5. Conclusions

We introduce a new model of computation—a polysemantic model of propositional sequential programs (GPSPs)—into which both sequential and recursive models of programs can be embedded. This gives a uniform framework for studying the equivalence-checking problem for various classes of programs. This framework substantially extends the algebraic formalism of propositional models of computer programs developed in [27, 10, 18, 19]. An attempt to introduce program semantics where the intermediate and final results of computations are separated was initiated in [16]. In that paper the first-order model of sequential programs is considered and final results of computations are defined as a projection of intermediate results on some subset of program variables. But unlike our approach, this type of semantics for final results does not maintain the composition of program statements.

Theorems 3 and 4 demonstrate that some equivalence-checking techniques initially developed for propositional models of sequential programs [28] can be readily adopted to a more general model of computation—generalized propositional sequential programs. This gives us a hope that some new decidable cases of equivalence-checking problem can still be found.

References

[1] E. Ashcroft, Z. Manna, A. Pnueli, A decidable properties of monadic functional schemes, *J. ACM*, vol 20 (1973), N 3, p.489-499.

[2] R. Bird, P. Walter, *Introduction to Functional Programming*, 1988, Prentice-Hall, Englewood Cliffs, NJ.

[3] J.W. De Bakker, D.A. Scott, A theory of programs. Unpublished notes, Vienna:IBM Seminar, 1969.

[4] A.P. Ershov, Theory of program schemata. In *Proc. of IFIP Congress 71*, Ljubljana, 1971, p.93-124.

[5] S.J. Garland, D.C. Luckham, Program schemes, recursion schemes and formal languages, *J. Comput. and Syst. Sci.*, **7**, 1973, p.119-160.

[6] D. Harel, Dynamic logics. In *Handbook of Philosophical Logics*, D. Gabbay and F. Guenthner (eds.), 1984, p.497-604.

[7] Y. Hirshfeld, F. Moller, Decidable results in automata and process theory. *LNCS*, **1043**, 1996, p.102-148.

[8] V.E. Kotov, V.K. Sabelfeld, *Theory of program schemes*, 1991.

[9] A.A. Lapunov, Yu.I. Yanov, On logical program schemata, In *Proc. Conf. Perspectives of the Soviet Mathematical Machinery*, Moscow, March 12-17, 1956, Part III.

[10] A.A. Letichevsky, On the equivalence of automata over semigroup, *Theoretic Cybernetics*, **6**, 1970, p.3-71 (in Russian).

[11] A.A. Letichevsky, Equivalence and optimization of programs. In *Programming theory*, Part 1, Novosibirsk, 1973, p. 166-180 (in Russian).

[12] A.A. Letichevsky, L.B. Smikun, On a class of groups with solvable problem of automata equivalence, *Sov. Math. Dokl.*, **17**, 1976, N 2, p.341-344.

[13] D.C. Luckham, D.M. Park, M.S. Paterson, On formalized computer programs, *J. Comput. and Syst. Sci.*, **4**, 1970, N 3, p.220-249.

[14] M.S. Paterson, Program schemata, *Machine Intelligence*, Edinburgh: Univ. Press, **3**, 1968, p.19-31.

[15] M.S. Paterson, Decision problems in computational models, *SIGPLAN Notices*, **7**, 1972, p.74-82.

[16] G.N. Petrosyan. On the decidable cases of the inclusion problem for sequential program schemes, in *System Informatics and Theory of Programming*. Novosibirsk, 1974, p.130-151 (in Russian).

[17] S. Peyton-Johns, *The implementation of Functional Programming*, 1987, Prentice-Hall, Englewood Cliffs, NJ.

[18] R.I. Podlovchenko, Hierarchy of program models, *Programming and Software Engineering*, 1981, N 2, p.3-14 (in Russian).

[19] R.I. Podlovchenko, Semigroup program models, *Programming and Software Engineering*, 1981, N 4, p.3-13 (in Russian).

[20] R.I. Podlovchenko, V.A. Zakharov, On the polynomial-time algorithm deciding the commutative equivalence of program schemata, *Reports of the Soviet Academy of Science*, **362**, 1998, N 6 (in Russian).

[21] H.G. Rice. Classes of recursively enumerable sets and their decision problems. *Trans. Amer. Math. Soc.*, bf 89, 1953, p. 25-59.

[22] V.K. Sabelfeld, Logic-term equivalence is checkable in polynomial time. *Reports of the Soviet Academy of Science*, **249**, 1979, N 4, p.793-796 (in Russian).

[23] V.K. Sabelfeld Tree equivalence of linear recursive schemata is polynomial-time decidable, *Information Processing Letters*, 1981, **13**, N 4, p.147-153.

[24] V.K. Sabelfeld, An algorithm deciding functional equivalence in a new class of program schemata, *Theoret. Comput. Sci.*, **71**, 1990, p.265-279.

[25] M.A. Taiclin,The equivalence of automata w.r.t. commutative semigroups, *Algebra and Logic*, **8**, 1969, p.553-600 (in Russian).

[26] V.A. Uspensky, A.L. Semenov, What are the gains of the theory of algorithms: basic developments connected with the concept of algorithm and with its application in mathematics. *LNCS*, bf 122, 1981, p.100-234.

[27] J.I. Yanov, To the equivalence and transformations of program schemata, *Reports of the Soviet Academy of Science*, **113**, 1957, N 1, p.39-42 (in Russian).

[28] V.A. Zakharov, The efficient and unified approach to the decidability of the equivalence of propositional programs. In *LNCS*, **1443**, 1998, p. 246-258.

[29] V.A. Zakharov, On the decidability of the equivalence problem for orthogonal sequential programs, *Grammars*, **2**, 1999, p.271-281.

[30] V.A. Zakharov, On the decidability of the equivalence problem for monadic recursive programs, *Theoretical Informatics and Applications*, **34**, 2000, p.157-171.