Deciding Properties of Integral Relational Automata*

(Extended Abstract)

Kārlis Čerāns**
Institute of Mathematics and Computer Science
University of Latvia

Summary. This paper investigates automated model checking possibilities for CTL* formulae over infinite transition systems represented by relational automata (RA). The general model checking problem for CTL* formulae over RA is shown *undecidable*, the undecidability being observed already on the class of Restricted CTL formulae. The decidability result, however, is obtained for another substantial subset of the logic, called A-CTL*+, which includes all "linear time" formulae.

1 Introduction

The area of automated deciding of properties of *infinite* state systems has during the recent years received quite a number of impressive positive developments. For instance, in the theory of Timed Automata (TA) the decidability has been established for the reachability and infinite behaviour possibility problems, as well as other TA properties expressible in temporal logic TCTL [3, 4]. Also bisimulation equivalence for TA has been shown decidable [10]. In [1] it was showed that safety properties (and liveness without fairness constraints) are decidable for systems of automata communicating over unbounded lossy channels. The decidability of state (marking) reachability for Petri Nets is due already back to [19, 20], this result has been extended to decidability of linear time temporal logic formulae over PN's in [17]. The bisimulation equivalence has been proved decidable for various subclasses of Petri Nets in [13] (for the full class of Petri Nets the bisimulation equivalence is undecidable [18]). One should continue this list with decidability results on context-free processes ([8], [14]), and probably many others.

In this paper we consider deciding properties of infinite transition systems which are represented by so-called Relational Automata (RA). Each RA, besides its finite control structure, has a finite number of *data variables*, which can assume values from certain *ordered* domain, in this paper taken to be the *integral numbers* (with the ordering relation "<"). The operation repertoire of RA includes comparison of variable values on their ordering, as well as input/output and dummy operations.

RA were originally introduced in [6, 7] where they naturally emerged as a model for studying possibilities of automated complete test set generation for data processing programs. It seems that RA could be also applicable in modelling and analyzing real time programs for data stream processing (when mutual relations of data values can take part in determining the control of processing). When suitably combined with data abstraction facilities, RA can model phenomena like *fairness*[3]. Still, in this paper we do not focus on the possible applications of RA, but allow them to speak mostly on their own.

* Part of this work was performed while the author was visiting Department of Computing Science, Chalmers University of Technology, Göteborg, Sweden.

** Email: karlis@mii.lu.lv

[3] This modelling makes heavy use of the integrality of the allowed variable value domain.

Our primary concern in this paper is to investigate the *decidability* of various verification problems over RA. In particular, we shall consider the possibilities of finding general procedures, which given an instance of RA, say P, and a temporal logic formula ξ decides whether ξ "holds" on P (i.e., whether P is a model for ξ). This problem in literature is often called "model checking" (due to [15]?). The logic used for describing properties of RA will be CTL* [15, 16], it contains modalities for specifying both the properties of linear behaviours of automata ("in next state", "until", "always"), and their transition system branching structure (quantifiers "for all (for some) computation paths"). The logic CTL* has showed its importance in describing and deciding properties of finite state systems.

The decidability of *vertex reachability* problem for RA has been shown already in [6, 7], see [5] for survey on this and related results. The problem of infinite behaviour set emptiness for RA (also admitting fairness constraints) has been shown decidable in [9]. The main contribution of this paper consists in showing (i) undecidability of the full CTL* model checking problem over RA (obtained already when considering the "purely branching" CTL* sublogic CTL), and (ii) designing a large subclass of CTL* for which a general deciding procedure exists (this subclass includes all A-CTL* formulae, and, so, also all "linear time" formulae)[4]. The general CTL* model checking undecidability result should be contrasted with the decidability of the similar model checking problem over the class of RA modified by allowing their variables to assume arbitrary rational values [11].

Partial preliminary reports on this work have appeared as [11] and [12], what contain some of the proofs only briefly outlined here. A full version of this paper is in preparation.

2 Relational Automata: The Basics

An (integral) relational automaton is a program with a finite number (say, k) of integral valued variables and the following allowed operations over them:

- $?x$ - input of a (new) integral value into the variable x,
- $!x$ ($!c$) - output of the value of the variable x (the constant c),
- $x < y$ ($x < c, c < y$) - a comparison of the values of two variables (the value of a variable with a constant), the operator produces an output flag "+" or "−" used to determine possible further control flow of the automaton,
- $x \leftarrow y$ ($x \leftarrow c$) - assignment of the value of the variable y (the constant c) to the variable x,
- NOP - the dummy operation.

Two simple examples of relational automata can be found in Figure 1. The first automaton in every execution loop after retrieving values into the variables x and y compares them and places the largest one into the variable (memory cell) z. After the assignment, the computation in every branch is leading to a vertex with a NOP operator in it, which can be thought of as being some operator not reflected in the defined language (e.g. it could do some computations with the value of z[5]). The second automaton illustrates the "fairness modelling" idea in RA: every infinite computation of it will output a value infinitely often.

[4] The presented decidability results rely on such mathematical constructions as simulation relation between states of RA, monotonicity of validity sets of temporal formulae and their automata based representation [24], graphose representation and symbolic manipulation of linear inequality systems, finite minorability of considered inequality system set wrt. certain natural ordering, and others. The obtained undecidability result relies on a standard technique of modelling 2 counter machine behaviours.

[5] In general, one can think of NOP as any operator which is not affecting the values of variables used further in computing the branching predicates.

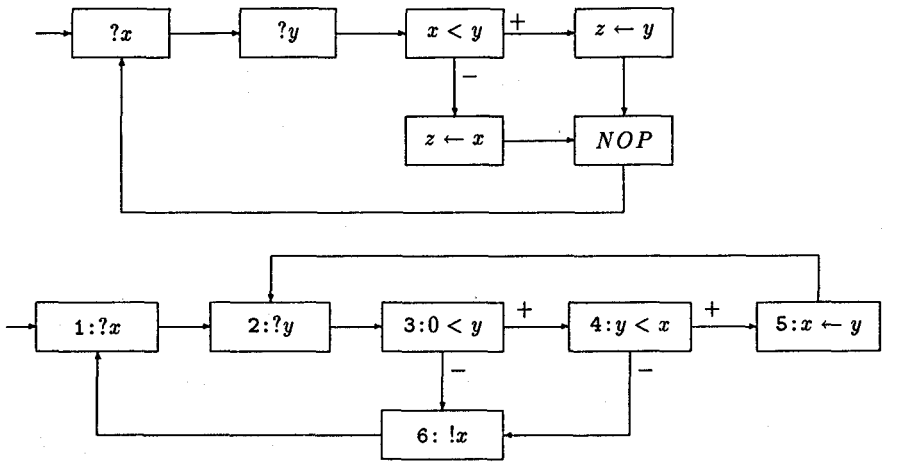

Figure 1 Examples of relational automata

One could easily imagine defining the notions of input and output ports for RA, as well as connecting the input and output operators to specific ports, what would allow to turn RA into certain process algebra with value passing. RA could be adequate also for representing some abstract view on shared variable computational models. However, these additional "decorations" on the RA formalism, though possibly being very important in applications, appear not relevant for our considered verification problems.

Formally a RA is defined to be a finite initial directed graph P with every vertex $n \in V(P)$ labelled by an operator $l(n)$ and every edge $e \in E(P)$ labelled by a flag $l(e)$ being "+" or "−". Relational automata are possibly nondeterministic since the ways of putting the "+" and "−" flags on the edges are, in general, not constrained. We say that a RA is *complete*, if every of its vertexes has at least one exit and every vertex with comparison operator has for every flag ("+" and "−") an exit labelled with it. Unless otherwise emphasized, we shall assume that RA under our consideration are complete[6]. Let us by $Vars(P)$ and $Cons(P)$ denote the sets of variables and constants used in operators of P (take also $0 \in Cons(P)$ and $c \in Cons(P)$ whenever $c_1 \leq c \leq c_2$ and $c_1, c_2 \in Cons(P)$).

The semantics of a RA P is given in a naive way by its state transition system (all program variables are initially set to 0). The set of states of P is defined to be $\mathcal{S}(P) = V(P) \times Z^k$, an arbitrary element of this set is a pair $\langle n, v \rangle$ where v is called automaton's variable value vector. For $x \in Vars(P)$ let $v.x$ denote the value of x in the vector v, moreover let $v.c = c$ for all $c \in Z$.

The transition relation[7] $\rightarrow \subseteq \mathcal{S}(P) \times E(P) \times \mathcal{S}(P)$ contains elements $s \xrightarrow{e} s'$ each denoting a possible (one step) transition from s to s' along the edge e. Given $n, n' \in V(P)$ and an edge $e \in E(P)$ from n to n' we have $\langle n, v \rangle \xrightarrow{e} \langle n', v' \rangle$ iff one of the following holds:

- $l(n)$ is input operator $?x$ and $v'.y = v.y$ for all $y \in Vars(P) \setminus \{x\}$ [8],

[6] There are easy ways of "completing" any RA by introducing to it an extra vertex and leading every "missing" edge there.

[7] Note that we are abstracting from the actual data values transmitted by input and output operators of RA. The change of the transition relation to reflect them is easy, however, that is not necesary for our considered problems.

[8] Note that input operators generate infinite branching.

- $l(n)$ is output or dummy operator, and $v' = v$,
- $l(n)$ is assignment $x \leftarrow a$ for $a \in Vars(P) \cup Cons(P)$ and $v' = v[x \leftarrow v.a]$ (using suggestive notation),
- $l(n)$ is comparison $a < b$, and $v = v'$, $v.a < v.b$, $l(e) =$" +", or
- $l(n)$ is comparison $a < b$, and $v = v'$, $v.a \geq v.b$, $l(e) =$" −".

We write $s \rightarrow s'$ whenever $s \xrightarrow{e} s'$ for some $e \in E(P)$.

The initial state of P is $s_0^P = \langle n_0^P, \mathbf{0} \rangle$, where $n_0^P \in V(P)$ is the initial vertex of P.

A *history* of a RA is a (finite or infinite) sequence of its states $\overline{s} = s_0, s_1, s_2, \ldots$ provided $s_i \rightarrow s_{i+1}$ for all i. An infinite history of a RA is called also a *computation*. Let $Comp(P)$ stand for the set of all P computations. For $\overline{s} \in Comp(P)$ and $m \in N$ we define the mth suffix of $\overline{s}$ to be $\overline{s}^{>>m} = s_m, s_{m+1}, \ldots$.

3 The Logic CTL*

We use the following logic CTL* [16, 15] to express properties of relational automata.

Given a RA P we define first the set $Pr(P)$ of *atomic state predicates* on P, ranged over by p, to be generated by the grammar ($n \in V(P)$ is a P vertex, $x, y \in Vars(P)$ and $c \in Z$):

$$p \quad ::= \quad n \mid x > y \mid x > c.$$

The CTL* formulae over P are defined as *state formulae* (ranged over by ξ) and *path formulae* (ranged over by ψ) mutually recursive way by the following grammar (see e.g. [15]):

$$\begin{array}{lll} \xi & ::= & p \mid true \mid \neg\xi \mid \xi \vee \xi \mid E\psi \\ \psi & ::= & \xi \mid \neg\psi \mid \psi \vee \psi \mid \psi \mathcal{U} \psi \mid \bigcirc\psi. \end{array}$$

The meanings $[\![\xi]\!] \subseteq \mathcal{S}(P)$ and $[\![\psi]\!]' \subseteq Comp(P)$ (path formulae are interpreted over *infinite* histories) are defined in a standard way[9] (see [16], $\bigcirc$ means "next", $\mathcal{U}$ means "until" and E is the existential quantifier over computations). We introduce also the standard abbreviations for formulae $false = \neg true$, $\xi \wedge \xi' = \neg((\neg\xi) \vee (\neg\xi'))$, $\psi \wedge \psi' = \neg((\neg\psi) \vee (\neg\psi'))$, $\Diamond\psi = true\ \mathcal{U}\ \psi$ ("sometimes") and $\Box\psi = \neg\Diamond\neg\psi$ ("always"), as well as $A\psi = \neg E\neg\psi$ ("for every computation" ψ holds).

Since the logic is standard we do not go into detail with any expressibility discussions.

A CTL* formula is called a *linear time formula* iff it is of the form $A\psi_0$ for ψ_0 being a path formula represented in the following grammar:

$$\psi_0 \quad ::= \quad p \mid true \mid \neg\psi_0 \mid \psi_0 \vee \psi_0 \mid \psi_0\, \mathcal{U}\, \psi_0 \mid \bigcirc\psi_0.$$

A CTL* formula is said to be in the "purely branching time" logic CTL iff it is generated by the grammar:

$$\xi_1 \quad ::= \quad p \mid true \mid \neg\xi_1 \mid \xi_1 \vee \xi_1 \mid E \bigcirc \xi_1 \mid A \bigcirc \xi_1 \mid A(\xi_1\, \mathcal{U}\, \xi_1) \mid E(\xi_1\, \mathcal{U}\, \xi_1).$$

[9] The meaning of an atomic state predicate is defined by $s \in [\![n]\!]$ iff $s = \langle n, v \rangle$ for some v, and

$$s = \langle n, v \rangle \in [\![x > y]\!] \text{ iff } v.x > v.y \text{ and } s = \langle n, v \rangle \in [\![x > c]\!] \text{ iff } v.x > v.c = c.$$

For a compound state formula we have $[\![true]\!] = \mathcal{S}(P)$, $[\![\neg\xi]\!] = \mathcal{S}(P) \setminus [\![\xi]\!]$, $[\![\xi_1 \vee \xi_2]\!] = [\![\xi_1]\!] \cup [\![\xi_2]\!]$ and $s \in [\![E\psi]\!]$ iff there exists a computation $\overline{s} = s_0, s_1, \ldots \in [\![\psi]\!]'$ such that $s_0 = s$.

For path formulae, first, $\overline{s} \in [\![\xi]\!]'$ iff $s_0 \in [\![\xi]\!]$, $[\![\neg\psi]\!]' = Comp(P) \setminus [\![\psi]\!]'$ and $[\![\psi_1 \vee \psi_2]\!]' = [\![\psi_1]\!]' \cup [\![\psi_2]\!]'$, as well as

- $\overline{s} \in [\![\bigcirc\psi]\!]'$ iff $\overline{s}^{>>1} \in [\![\psi]\!]'$ and
- $\overline{s} \in [\![\psi_1 \mathcal{U} \psi_2]\!]'$ iff $\overline{s}^{>>m} \in [\![\psi_2]\!]'$ for some $m \in N$ and $\overline{s}^{>>r} \in [\![\psi_1]\!]'$ for all $r < m$.

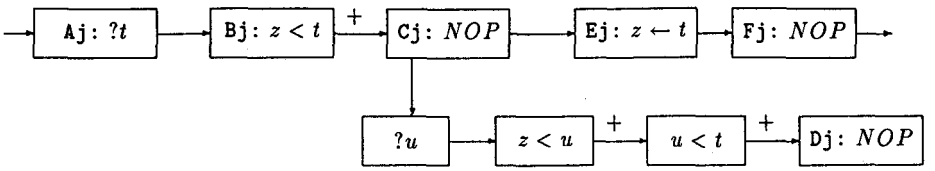

Figure 2 A block for modelling counter increment

A CTL formula is in RCTL, if it does not contain $A\bigcirc$ and $E\bigcirc$ modalities.

For any RA P let $Form(P)$ be some set of CTL* formulae over P. Then the *model checking problem* for the family (class) of formulae $Form$ over RA is, *given* a RA P and a state formula $\xi \in Form(P)$, *determine*, whether ξ is satisfied by the initial state of the automaton (i.e., if $s_0^P \in [\![\xi]\!]$).

4 The Undecidability Result

Theorem 1 *The model checking problem for CTL* formulae over RA is undecidable (the undecidability already takes place for RCTL formulae).*

Proof: By modelling of 2 counter machines [21] with each counter having increment, decrement and zero test possibilities (zero test assumed to be asking, if $x > 0$). For each Minsky machine M_i we shall construct a RA R_i and a RCTL formula ξ_i over R_i such that $s_0^{R_i} \in [\![\xi_i]\!]$ iff M_i can reach a certain designated ("halting") vertex H. In view of the undecidability of the reachability problem for Minsky machines, this will prove the theorem.

Let z be a metavariable to denote any of Minsky machine's M_i counters x and y. To build R_i corresponding to M_i we replace every counter increment operation $z := z + 1$, residing at jth vertex in M_i, by the script of Figure 2 [10]. Every decrement operation is replaced by a similar script with all inequalities turned around. t and u are auxiliary variables.

Let a state $s \in \mathcal{S}(R_i)$ be *j-fair* iff

$$s \in [\![Q_j]\!] \text{ for } Q_j = (\neg \mathtt{Cj}:) \vee \neg E(\neg \mathtt{Aj}: \mathcal{U}\ \mathtt{Dj}:)\ [11].$$

A history $\overline{s}$ of R_i is j-fair iff all its states are.

Fact 4.1 *If $\langle Aj:, v\rangle, \langle Bj:, v'\rangle, \langle Cj:, v'\rangle, \langle Ej:, v'\rangle, \langle Fj:, v''\rangle$ is a fragment of a j-fair history of R_i, then $v''.z = v.z + 1$.*

The vertex H will be reachable in M_i iff it is reachable in R_i by a history which is j-fair for all occurences j of counter increment/decrement operations in M_i. So we reduce the checking of reachability of H in M_i to checking, if $s_0^{R_i} \in [\![\xi_i]\!]$ for $\xi_i = E(\bigwedge_j Q_j\ \mathcal{U}\ H)$. □

Note that the model checking for CTL* formulae over finite state systems is decidable (for CTL formulae this decidability is even easy in the sense of complexity [15]). As follows from [11], the model checking for CTL* formulae is decidable also for a modification of relational automata to allow rational variable values[12] (one cannot implement any "counting" operations using the comparison of rational valued variables).

[10] All non specified edges are leading to a dead vertex.

[11] Reading: $\langle n, v\rangle$ is j-fair iff either $n \neq \mathtt{Cj}:$, or no history starting at $\langle n, v\rangle$ can reach $\mathtt{Dj}:$ without prior passing through $\mathtt{Aj}:$. For $n = \mathtt{Cj}:$ the second condition is equivalent to $v.t - v.z \leq 1$, what in view of the inequality observed at $\mathtt{Bj}:$ implies $v.t - v.z = 1$.

[12] For "rational variable" version of RA one can show that the whole state space of any such automaton can be finitely partitioned with respect to a bisimulation equivalence abstracted from the actual data values, but sensitive to edge names along which the transitions are performed, see [11].

5 Decidable Classes of Formulae

In this section some decidability results for model checking of subclasses of CTL* formulae over RA are formulated, and their proofs outlined.

5.1 Existential and Universal Formulae

For X being a set of CTL* state formulae, ranged over by ξ, let the set $PLF(X)$ of *positive linear path formulae* over X be produced by the following grammar:

$$\psi \quad ::= \quad \xi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \psi \mathcal{U} \psi \mid \bigcirc \psi \mid \Box \psi.$$

A CTL* state formula is inductively defined to be *existential* (resp. *universal*) formula over given automaton P iff it is either

- an atomic state predicate $p \in Pr(P)$, or its negation $\neg p$,
- a conjunction ($\wedge$) or disjunction ($\vee$) of two existential (resp. universal) formulas, or
- a formula $E\psi$ (resp. $A\psi$) for $\psi \in PLF(X)$, where X is some set of existential (resp. universal) formulae.

We denote the class of all existential formulae by E-CTL* and the class of all universal formulae by A-CTL*.

Theorem 2 *The model checking problem for E-CTL* formulae over RA is decidable.*

Since the negation of a universal formula can be rewritten (using standard CTL* identities) to an equivalent existential formula, and any linear time CTL* formula can be transformed into equivalent universal formula, from Theorem 2 one obtains the following corollaries.

Corollary 5.1 *The model checking problem for A-CTL* formulae over RA is decidable.*

Corollary 5.2 *The model checking problem for linear time CTL* formulae over RA is decidable.*

Proof of Theorem 2 (outline): Let for a given RA P and its states $s = \langle n, v \rangle$ and $s' = \langle n', v' \rangle$ the state s' be called *sparser* than s, written $s' \succeq s$, iff

- $n' = n$ (the states are at the same vertex) and
- $v' \succeq v$ (v' is sparser than v) meaning
  - $v'.a \geq v'.b$ iff $v.a \geq v.b$ for all $a, b \in Vars(P) \cup Cons(P)$ (v' and v coincide on their variable and P constant value ordering) and
  - $v.a > v.b$ implies $v'.a - v'.b \geq v.a - v.b$.

For instance, if $Cons(P) = \{0, 1, 2\}$, then (variable value vector) $\langle 2, 10, 12, 1994 \rangle$ is sparser than $\langle 2, 4, 6, 1000 \rangle$, but not sparser $\langle 1, 10, 12, 1994 \rangle$ since the value of the first variable is no longer equal to the constant 2, and not sparser than $\langle 2, 4, 7, 17 \rangle$ since $7 - 4 > 12 - 10$.

Lemma 5.3 *The ordering $\succeq$ is a simulation, i.e. for $s, s', s_1 \in \mathcal{S}(P)$, whenever $s \rightarrow s'$ and $s_1 \succeq s$ then $s_1 \rightarrow s_1'$ for some $s_1' \succeq s'$.*

Let a state set $U \subseteq \mathcal{S}(P)$ be called *upwards closed*, if $s \in U$ and $s' \succeq s$ implies $s' \in U$. Lemma 5.3 allows to prove the following proposition, which exhibits an invariant used in the structural induction proof of Theorem 2.

Proposition 5.4 *Whenever for a given automaton P $[\![\xi]\!]$ is upwards closed for all $\xi \in X$, then also $[\![E\psi]\!]$ is upwards closed for all $\psi \in PLF(X)$.*

What remains to be done in the proof, is (i) defining a finite representation $Repr(U)$ for any upwards closed set $U \subseteq \mathcal{S}(P)$, (ii) showing how to obtain a representation of $[\![E\psi]\!]$ for $\psi \in PLF(X)$ given $Repr([\![\xi]\!])$ for all $\xi \in X$, and (iii) showing, how to perform test of inclusion whether $s \in U$ given $s \in \mathcal{S}(P)$ and $Repr(U)$ for upwards closed U [13].

We proceed to developping our analysis techniques.

5.2 Algebra of Graphose Inequality Systems

A graphose inequality system ("GS", for short) is defined to be a triple $G = \langle D, C, w \rangle$, where D is a finite set of *variables*, $C \subseteq Z$ is a finite set of integral constants and $w : A \times A \to Z^-$ for $A = D \cup C$ and $Z^- = Z \cup \{-\infty\}$ is a *weight function.*

A mapping $\Gamma : D \to Z$ is said to be a solution of G, written $\Gamma \in Sol(G)$, iff

$$\Gamma(a) - \Gamma(b) \geq w(a,b) \text{ for all } a, b \in A,$$

where for $c \in C$ by definition $\Gamma(c) = c \in Z$[14].

A GS is called *consistent*, if it has a solution. Let for a GS $G = \langle D, C, w \rangle$ and $a, b \in D \cup C$ $p(a,b) \in Z^- \cup \{\infty\}$ be the lowest upper bound[15] of the weight sums on all paths in G from a to b.

Lemma 5.5 *A GS $G = \langle D, C, w \rangle$ is consistent iff $p(a,b) < \infty$ for all $a, b \in D \cup C$ and $p(c,d) \leq c - d$ for all $c, d \in C$.*

For a consistent GS $G = \langle D, C, w \rangle$ let a GS $G' = \langle D, C, w' \rangle$ with $w'(a,b) = p(a,b)$ for all $a, b \in D \cup C$ be called the *normal form* of G, written $G' = |G|$. Let also $\top$ be some unique normal form for all inconsistent GSs. Assuming $Sol(\top) = \emptyset$, one can show that $Sol(|G|) = Sol(G)$ for every GS G.

For $G_1 = \langle D_1, C_1, w_1 \rangle$ and $G_2 = \langle D_2, C_2, w_2 \rangle$ let their *intersection* $G = G_1 \otimes G_2 = \langle D, C, w \rangle$ have $D = D_1 \cup D_2$, $C = C_1 \cup C_2$ and

$$w(a,b) = max\{w'_1(a,b), w'_2(a,b)\}, \text{ where}$$

$w'_i(a,b) = w_i(a,b)$, if $a, b \in D_i \cup C_i$, and $w'_i(a,b) = -\infty$ otherwise $(i = 1, 2)$.

We say that a GS $G = \langle D, C, w \rangle$ is a *projection* of a GS $G' = \langle D', C', w' \rangle$ onto D, written $G = Proj(G', D)$, provided $D \subseteq D'$, $C = C'$ and for all $a, b \in D \cup C$ $w(a,b) = w''(a,b)$, where $G'' = \langle D', C', w'' \rangle = |G'|$ is a normal form of G' (a projection of a contradictory GS is $\top$). One can show that normal form, intersection and projection computation over GSs is effective.

A GS $G = \langle D, C, w \rangle$ is called *positive* (referred as "PGS") iff for any $a, b \in D \cup C$ either $w(a,b) \geq 0$, or $w(a,b) = -\infty$ (intuitively, a GS is positive iff all weights on the existing edges are nonnegative).

---

[13] Notably, (i) and (iii) can be solved by a general technique, requiring from the relation $\succeq$ just the property called "finite minorability" (see below Section 5.2.1) - any u.c. set could be represented just by the (finite) set of its minorants. As to (ii), in this representation it would be easy to *enumerate* the minorant set for $[\![E\psi]\!]$. However, as recent results in theory of verification of unbounded lossy channel automata show [2] (see [1] for formulation of the model and some decidability results), the requirements of simulation together with finite minorability alone are not sufficient to decide even "fair liveness" properties, easily expressible in linear time logic as $A(\Box\Diamond p)$.

[14] One can depict the GS G as a graph with the set A of all its variables and constants as the set of *vertexes*, having an edge from $a \in A$ to $b \in A$ with the *weight* $w(a,b)$ whenever $w(a,b) > -\infty$. Finding a solution to such a G means assigning integral values to the variable vertexes in it so that for every edge in G the difference between its source and target vertex values is at least the weight associated with the edge.

[15] The maximum, if it exists.

An *ordering* relation $\leq$ in the set of GSs is introduced in a way that $G = \langle D, C, w\rangle \leq G' = \langle D', C', w'\rangle$, if $D' \subseteq D$, $C' \subseteq C$ and for every $a, b \in D' \cup C'$ we have $w(a,b) \leq w'(a,b)$. We let also $G \leq \top$ for every GS G (the intuition is that a "bigger" GS is "more restrictive"). One shall observe that $\leq$ is well founded in the set of PGSs over a fixed set of variables and constants. Some important properties of the introduced operations over GSs follow.

Lemma 5.6 *For $G = \langle D, C, w\rangle$ and $G' = \langle D', C', w'\rangle$ always*

- *for $\Gamma : D \cup D' \to Z$ $\Gamma \in Sol(G \otimes G')$ if and only if $\Gamma|_D \in Sol(G)$ and $\Gamma|_{D'} \in Sol(G')$ ($|_A$ is function narrowing); so, for $D = D'$ we have $Sol(G \otimes G') = Sol(G) \cap Sol(G')$;*
- *if $G = Proj(G', D)$, then $\Gamma \in Sol(G)$ for $\Gamma : D \to Z$ iff $\Gamma = \Gamma'|_D$ for some $\Gamma' \in Sol(G')$ (the solution set of G is the projection of the solution set of G');*
- *if $G \leq G'$, then $Sol(G') \subseteq Sol(G)$ (observe "reversed order");*
- *if $G \leq G'$, then $|G| \leq |G'|$, $G \otimes G'' \leq G' \otimes G''$ and $Proj(G, D) \leq Proj(G', D)$;*
- *if G and G' are PGSs, then PGSs are also $|G|$, $Proj(G, D)$ and $G \otimes G'$.*

5.2.1 Finite Minorability Property

Let X be a set with a defined well-founded partial order relation $\preceq$ on it. We define for every subset $X_1 \subseteq X$ its *minorant set* $Min(X_1)$ wrt. $\preceq$ to consist of those and only those $x \in X_1$ for which there is no $x' \in X_1$ such that $x' \neq x$ and $x' \preceq x$ (due to the well-foundedness of $\preceq$ (all decreasing chains wrt. it are finite) we have also that $Min(X_1)$ is precisely the smallest subset of X_1 containing a minorant $x' \preceq x$ for every $x \in X_1$).

Definition 5.7 *A set X is called* finitely minorable *wrt. the partial order relation $\preceq$ defined on it, if for every subset $X_1 \subseteq X$ the minorant set $Min(X_1)$ is finite.*

Lemma 5.8 *The set of PGSs over a fixed set of variables and constants is finitely minorable wrt. $\leq$.*

Proof: Follows from the finite minorability of the natural number vector set wrt. the relation $\leq_n$ defined $\langle a_1, \ldots, a_n\rangle \leq_n \langle b_1, \ldots, b_n\rangle$ iff $a_i \leq b_i$ for all $i \leq n$ (the proof of finite minorability of $\leq_n$ can be found in e.g. [22]). □

Lemma 5.9 *The state set of any RA P is finitely minorable wrt. $(\succeq)^{-1}$.*

5.3 Representing State Sets and Relations

In this section we continue the outline of the proof of Theorem 2, and show, at first, how we shall represent RA state sets and relations on these sets in form of (P)GSs.

For $Vars = Vars(P) = \{x_1, \ldots, x_k\}$ being the set of the program P variables let $Vars' = \{x'_1, \ldots, x'_k\}$ and $Vars'' = \{x''_1, \ldots, x''_k\}$. Let also for any constant $c \in Z$ $c' = c'' = c$.

A GS $H = \langle D, C, w\rangle$ is called *local* for the RA P iff $D = Vars$ and $C = Cons(P)$. We say that a P variable value vector $\boldsymbol{v}$ *satisfies* H, and write $\boldsymbol{v}$ **sat** H iff the mapping $\Gamma : D \to Z$ defined $\Gamma(x) = \boldsymbol{v}.x$ is a solution of H.

A GS $G = \langle D, C, w\rangle$ is *transitional* for P iff $D = Vars \cup Vars'$ and $C = Cons$. A pair of P variable value vectors $\langle \boldsymbol{v}, \boldsymbol{v}'\rangle$ is said to satisfy G, written $\langle \boldsymbol{v}, \boldsymbol{v}'\rangle$ **sat** G iff the mapping $\Gamma : D \to Z$, defined by $\Gamma(x) = \boldsymbol{v}.x$ and $\Gamma(x') = \boldsymbol{v}'.x$ for $x \in Vars$, is a solution of G.

A state set $U \subseteq \mathcal{S}(P)$ is said to be *PGS-represented* by a family of finite sets $(\mathcal{H}_n)_{n \in V(P)}$ of local PGSs provided a state $\langle n, \boldsymbol{v}\rangle \in U$ iff $\boldsymbol{v}$ **sat** H for some PGS $H \in \mathcal{H}_n$. The PGS representation $\mathcal{H}$ of U is called *simple* iff for every $n \in V(P)$ the set $\mathcal{H}_n$ consists of single PGS. A state set representation by a family of arbitrary GS sets is defined in a similar way.

Note that every PGS-representable set $U \subseteq \mathcal{S}(P)$ is also upwards closed. By Lemma 5.9 one can show (non-constructively) also that every upwards closed set is PGS-representable.

A finite set $\mathcal{G}$ of transitional PGSs is said to *characterize* a relation $\rightsquigarrow_0 \subseteq \mathcal{S}(P) \times \mathcal{S}(P)$ for a pair of vertexes $\langle n, n' \rangle \in V(P) \times V(P)$ provided $\langle n, v \rangle \rightsquigarrow_0 \langle n', v' \rangle$ iff $\langle v, v' \rangle$ **sat** G for some $G \in \mathcal{G}$.

Since the membership test for a state set in a PGS representation is straightforward, the following result will lead us to the proof of Theorem 2 (and to the deciding algorithm).

Theorem 3 *The validity set of an E-CTL* formula ξ over given RA P is effectively PGS-representable.*

Proof: By induction on the structure of given E-CTL* formula ξ. First, both $[\![p]\!]$ and $[\![\neg p]\!]$ are trivially PGS-representable (a single PGS for every vertex $n \in V(P)$ is sufficient).

For $\mathcal{H}^{[\xi_i]}$ being PGS representations of $[\![\xi_i]\!]$ define the PGS representation $\mathcal{H}^{[\xi]}$ of $[\![\xi]\!]$ in a way:

- for $\xi = \xi_1 \vee \xi_2$ let $\mathcal{H}^{[\xi]}(n) = \mathcal{H}^{[\xi_1]}(n) \cup \mathcal{H}^{[\xi_2]}(n)$ for all $n \in V(P)$, and
- for $\xi = \xi_1 \wedge \xi_2$ let $\mathcal{H}^{[\xi]}(n) = \{H_1 \otimes H_2 \mid H_1 \in \mathcal{H}^{[\xi_1]}(n)\ \&\ H_2 \in \mathcal{H}^{[\xi_2]}(n)\}$ $(n \in V(P))$.

For the case when $\xi = E\psi$ we prove the following "constructive" version of Proposition 5.4.

Lemma 5.10 *For ψ being a positive linear formula over X one can construct the PGS representation of $[\![E\psi]\!]$ from given PGS representations of $[\![\xi]\!]$ for all $\xi \in X$.*

Proof: Let for a state set $U \subseteq \mathcal{S}(P)$ and a vertex set $F \subseteq V(P)$ the formula $E\Box^F U$ be defined true on a state $s \in \mathcal{S}(P)$ (written $s \in [\![E\Box^F U]\!]$) if and only if there exists a computation $\overline{s} = s_0, s_1, \ldots$ starting at s for which both

- $s_i \in U$ for all $i \in N$, and
- $s_i = \langle n_i, v_i \rangle$ with $n_i \in F$ for *infinitely many* i.

Using the techniques of automata-based program verification, as developped in [23], [24], one can reduce the building of a PGS representation for the set $[\![E\psi]\!]$ to the building PGS representation for $[\![E\Box^F U]\!]$ where the PGS representation of U is available[16].

5.4 Characterization of "Fair Liveness" Sets

Lemma 5.11 *There exists an algorithm which given a RA P, a distinguished set of its vertexes $F \subseteq V(P)$ and a state set $U \subseteq \mathcal{S}(P)$ represented by a family of PGSs $\mathcal{H}^U_n$, $n \in V(P)$, constructs a PGS representation of the set $[\![E\Box^F U]\!]$.*

Proof: Without loosing generality one can assume that for every vertex n the PGS set $\mathcal{H}^U_n$ consists of a single local inequality system (one can make copies of all vertexes n for which $\mathcal{H}^U_n$ consists of more than one inequality system - one copy for each contained system).

To proceed further, we need some more notation. For $U \subseteq \mathcal{S}(P)$ and $\overline{n} = n_0, n_1, \ldots, n_p$, $p > 0$ being a path in P we say that a state $s_p = \langle n_p, v_p \rangle$ is U-reachable from $s_0 = \langle n_0, v_0 \rangle$ along $\overline{n}$, written $s_0 \rightsquigarrow^U_{\overline{n}} s_p$, iff there exists a history $s_0, s_1, \ldots, s_p$ such that $\forall i. s_i = \langle n_i, v_i \rangle \in U$. We let $s \rightsquigarrow^U s'$ iff $s \rightsquigarrow^U_{\overline{n}} s'$ for some $\overline{n}$, as well as omit the index U in $\rightsquigarrow^U_{\overline{n}}$ and $\rightsquigarrow^U$ when $U = \mathcal{S}(P)$. A history $\overline{s}$ with all $s_i \in U$ will be called U-history.

What follows is a principal result about PGS representations of relations $\rightsquigarrow^U$ for PGS represented Us. A transitional PGS G over P is called U-inequality system for the path $\overline{n}$ from n to n' in P iff the singleton set $\{G\}$ characterizes $\rightsquigarrow^U_{\overline{n}}$ for the pair of vertexes $\langle n, n' \rangle$.

[16] Usually this leads to considering a larger RA P' instead of P, the vertex set $F \subseteq V(P')$ is also constructed in an appropriate way. The construction of PGS representation of U is obtained at every vertex $n \in V(P')$ by taking unions and intersections of appropriate PGS sets representing ξs at various vertexes of P.

Theorem 4 *There exists an algorithm which given a RA P, a simple PGS-representation $\mathcal{H}$ of a set $U \subseteq \mathcal{S}(P)$, and a pair of vertexes $\langle n, n' \rangle \in V(P) \times V(P)$ constructs a finite set $\mathcal{G}^U(n,n')$ of transitional PGSs characterizing the* reachability *relation $\rightsquigarrow^U$ for $\langle n, n' \rangle$. Moreover, every $G \in \mathcal{G}^U(n,n')$ is the U-inequality system for some path in P from n to n'.*

Proof outline: We start by inductively obtaining U-inequality systems for paths of P. In case of one edge paths such a system can be got directly from the operational semantics definition; a defined composition operation on PGSs (using intersection, projection and variable renaming) works for induction step. The characterization of the whole relation $\rightsquigarrow^U$ in terms of $\mathcal{G}^U(n,n')$ for every $\langle n, n' \rangle$ is obtained by simultaneously building finite approximations characterizing the relations $\bigcup\{\rightsquigarrow_{\overline{n}} | \ length(\overline{n}) < 2^k\}$, $k \in N$, at every step eliminating inequality systems covered (subsumed) by less restrictive ones (in sense of $\leq$), until the stabilization is reached for all n, n'. The termination of the algorithm is due to the finite minorability of the PGS set wrt. $\leq$ (Lemma 5.8). More details are in [11, 12]. □

For $X, U \subseteq \mathcal{S}(P)$ we define $re^U(X) = \{s \in \mathcal{S}(P) \mid \exists s' \in X : s \rightsquigarrow^U s'\}$.

Corollary 5.12 *For $X, U \subseteq \mathcal{S}(P)$, given a simple PGS representation $\mathcal{H}^U$ of U and a GS representation $\mathcal{H}^X$ of X, the construction of a GS representation $\mathcal{H}$ of $re^U(X)$ is effective. Further, if $\mathcal{H}^X$ is a PGS-representation of X, $\mathcal{H}$ will be a PGS-representation of $re^U(X)$.*

Proof: We define for every $n \in V(P)$

$$\mathcal{H}(n) = \{Proj(G \otimes H', Vars) | n' \in V(P), G \in \mathcal{G}^U(n,n'), H \in \mathcal{H}^X(n')\},$$

where H' is obtained from H just by priming all variables. The correctness of construction follows from properties of GSs (Lemma 5.6).□

For $n \in V(P)$ let $Loop(n)$ denote the set of all paths from n to n.

For $\mathcal{G}^U(n,n) = \{G(\overline{n}^1), G(\overline{n}^2), \ldots, G(\overline{n}^m)\}$ being the set of inequality systems characterizing the reachability relation $\rightsquigarrow^U$ for the pair of vertexes $\langle n, n \rangle$ (the existence of such a system is by Theorem 4), we define $Rep(n) = \{\overline{n}^1, \overline{n}^2, \ldots, \overline{n}^m\} \subseteq Loop(n)$. Clearly, $\langle n, v \rangle \rightsquigarrow^U \langle n, v' \rangle$ iff $\langle n, v \rangle \rightsquigarrow^U_h \langle n, v' \rangle$ for some $h \in Rep(n)$.

For $h \in Loop(n)$ we define $Rea^\infty(h)$ to be the set of states $\langle n, v \rangle$ such that there exists an U-computation along the path h^∞ (the infinite path consisting of infinitely many times repeated path h) which starts at $\langle n, v \rangle$. Let $Sp(h)$ be the set of states $s = \langle n, v \rangle$ such that $s \rightsquigarrow^U_h s'$ for some $s' \succeq s$.

By simulation properties of $\succeq$ (Lemma 5.3) $Sp(h) \subseteq Rea^\infty(h)$ for every h. We let

$$L = \{s \in \mathcal{S}(P) \mid s \in Sp(h), \ \ for \ \ h \in Loop(n), n \in F\},$$
$$L_0 = \{s \in \mathcal{S}(P) \mid s \in Sp(h), \ \ for \ \ h \in Rep(n), n \in F\},$$
$$K_0 = \{s \in \mathcal{S}(P) \mid s \in Rea^\infty(h), \ \ for \ \ h \in Rep(n), n \in F\}.$$

Obviously, $L_0 \subseteq L$. Since $SP(h) \subseteq Rea^\infty(h)$ for every h, we have $L_0 \subseteq K_0$.

Lemma 5.13 *$re^U(L_0) = re^U(L) = [\![E \Box^F U]\!] = re^U(K_0)$. Moreover, $[\![E \Box^F U]\!] = re^U(T)$ for any $T \subseteq \mathcal{S}(P)$ satisfying $L_0 \subseteq T \subseteq K_0$.*

Lemma 5.13 together with Corollary 5.12 reduce the proof of Lemma 5.11 and, so, Theorem 3 and Theorem 2, to showing, how to build for fixed $n \in V(P)$ and $h \in Rep(n)$ a local inequality system, denoted H^h, such that

$$Sp(h) \subseteq \{\langle n, v \rangle \in \mathcal{S}(P) \mid v \ \mathbf{sat} \ H^h\} \subseteq Rea^\infty(h).$$

This construction will be explained in the full paper (it goes by taking the U-inequality system G of h, composing it with itself certain amount of times, adding a few new edges (inequalities), and finally taking the projection to the initial variable set). □□□□

5.5 "Advancing" in Logic

Let A-CTL*+ denote the sublogic of CTL*, generated by the grammar

$$\xi ::= \xi_e \mid \xi_a \mid \neg\xi \mid \xi \vee \xi \mid \xi \wedge \xi \mid E\xi_e \mathcal{U} \xi \mid A\Box\xi \mid E \bigcirc \xi \mid A \bigcirc \xi,$$

where ξ_e and ξ_a run over the sets of all existential and universal CTL* formulas, respectively.

Theorem 5 *The validity set of given A-CTL*+ formula ξ over a RA P is effectively GS-representable. So, the model checking problem for A-CTL*+ formulae over RA is decidable.*

Before we prove the theorem, note that a simple property expressible in A-CTL*+, but not in A-CTL*, is (relative[17]) deadlock: $p \wedge A\Box(E \bigcirc p)$ [18].

Lemma 5.14 *Given a GS representation $\mathcal{H}$ of a set $X \subseteq \mathcal{S}(P)$, one can construct a GS representation of $\mathcal{S}(P) \setminus X$.*

Proof idea: For a GS $G = \langle D, C, w\rangle$ and every $a, b \in D \cup C$, if $w(a,b) > -\infty$, produce a GS $G_{\{a,b\}} = \langle D, C, w_{\{a,b\}}\rangle$ with the weight function $w_{\{a,b\}}(b,a) = -(w(a,b)+1)$, and $w_{\{a,b\}}(c,d) = -\infty$ on all other pairs of vertices $c, d \in D \cup C$. For $\Gamma : D \to Z$ clearly $\Gamma \in Sol(G)$ iff $\Gamma \notin Sol(G_{\{a,b\}})$ for all $a, b \in D \cup C$. □

The proof of the theorem now is easily obtained by structural induction, replacing ξ_a by $\neg\xi_e$ for appropriate existential ξ_e, as well as rewriting $A\Box\xi = \neg E(true\ \mathcal{U}(\neg\xi))$ and $A \bigcirc \xi = \neg E \bigcirc (\neg\xi)$, and noticing, as necessary, that

- any PGS representation of a state set is also its GS representation (obtained for $\xi = \xi_e$ due to Theorem 3), and
- $[\![E(\xi_1 \mathcal{U} \xi_2)]\!] = re^{[\xi_1]}([\![\xi_2]\!])$, what due to Corollary 5.12 can be effectively GS-represented[19].

Further details will follow in the full paper. □

6 Concluding Remarks

In the full version of this paper the "fairness" example of Figure 1 shall be by "hand-simulation" of the presented algorithm proved correct wrt. the property $A(\Box\Diamond 6:)$ (running the example still requires some "engineering" enhancements to the presented algorithm).

A possible future work would be to consider the computational costs of the model checking algorithm. One can hope that they could be kept "reasonable" (simply exponential in size of number of variables and logical formula, linear in size of the program) for linear time formulae. However, this requires further investigations since currently the termination proof uses the non-constructive method of finite minorability (Section 5.2.1).

When comparing the model checking results, obtained in case of RA with those in analysis of other discrete infinite models (e.g. Petri Nets, see [17]), they appear to be, in principle, of no surprise, since it seems nowadays to be already a general observation that automated checking linear time formulae are on these models *easier* than checking the branching properties (undecidability usually encountered in branching case). However, this observation should be contrasted with the case of finite state systems, where the branching time logics gained a speedup of verification [15].

Another interesting future question seems to be to investigate the possibilities of transferring the decidability results, obtained in case of RA, to other problems. For instance,

[17] Absolute deadlocks are not possible in complete RA. The property p in the following formula is intended to describe being in a certain incomplete part of automaton.

[18] On the other hand, the formulae ξ_i used in modelling 2 counter machines in Section 4 are still not in A-CTL*+ (why?).

[19] Observe that ξ_1 is required to be existential formula.

one could ask, if there are natural discrete infinite models other than RA, on which model checking for A-CTL* formulae would be decidable (decidability of linear time formulae is at least for Petri Nets [17]). However, it seems doubtful, if a general mathematical theory for deciding properties of (discrete) infinite systems could be developped.

References

[1] P.Abdulla, B.Jonsson, *Verifying Programs with Unreliable Channels*, in Proc. of LICS'93, Montreal, Canada, 1993.

[2] P.A.Abdulla, B.Jonsson, *Undecidable Verification Problems for Programs with Unreliable Channels*, in Proc. of ICALP'94, LNCS, this volume, 1994.

[3] R. Alur and D.Dill, *Automata for Modelling Real-Time Systems*, in Proc. of ICALP'90, LNCS, No.443, 1990.

[4] R. Alur, C.Courcoubetis and D.Dill, *Model-Checking for Real-Time Systems*, in Proc. of LICS'90, pp. 414-425, 1990.

[5] A. Auziņš, J. Bārzdiņš, J. Bičevskis, K. Čerāns and A. Kalniņš, *Automatic Construction of Test Sets: Theoretical Approach*, in Baltic Computer Science, LNCS, No. 502, 1991.

[6] J.M.Barzdin, J.J.Bicevskis and A.A.Kalninsh, *Construction of Complete Sample Systems for Program Testing*, in Latv. Gosudarst. Univ. Uch. Zapiski, Vol.210, 1974 (In Russian).

[7] J.M.Barzdin, J.J.Bicevskis and A.A.Kalninsh, *Automatic Construction of Complete Sample Systems for Program Testing*, in Proc. IFIP Congress, 1977, North-Holland 1977.

[8] O. Burkhart, B. Steffen, *Model Checking for Context-Free Processes*, In Proc. of CONCUR'92, LNCS No. 630, 1992.

[9] K. Čerāns, *Feasibility of Finite and Infinite Paths in Data Dependent Programs*, in Proc. for LFCS'92, Russia, Tver, LNCS No. 620, 1992.

[10] K. Čerāns, *Decidability of Bisimulation Equivalences for Parallel Timer Processes*, in Proc. of CAV'92, LNCS No.663, 1992.

[11] K. Čerāns, *Deciding Properties of Relational Automata: Reachability Relation Characterization*, in Proc. of 4th Nordic Workshop on Program Correctness, Report No.78, Department of Informatics, University of Bergen, Bergen, Norway, 1993.

[12] K. Čerāns, *Deciding Properties of Integral Relational Automata*, in Proc. of *El Wintermöte*, Report No.73, PMG, Dept. of Computer Sciences, Chalmers University of Technology, Göteborg, Sweden, 1993.

[13] S. Christensen, Y. Hirshfeld, F. Moller *Bisimulation equivalence is decidable for Basic Parallel processes*, in Proc. of CONCUR'93, LNCS No. 715, 1993.

[14] S. Christensen, H. Hüttel, C. Stirling, *Bisimulation equivalence is decidable for all context free processes*, in Proc. of CONCUR'92, LNCS No. 630, 1992.

[15] E.M.Clarke, E.A.Emerson and A.P.Sistla, *Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications*, in ACM Transactions on Programming Languages and Systems, Vol.8, No.2, 1986.

[16] E.A.Emerson, J.Y.Hailpern, *"Sometimes" and "not never" revisited: On Branching versus Linear Time*, in Proc. of ACM POPL'83, 1983.

[17] J. Esparza, *On the decidability of model checking for several μ-calculi and Petri nets*, Report ECS-LFCS-93-274, Department of Computer Science, University of Edinburgh, 1993.

[18] P. Jančar, *Decidability Questions for Bisimilarity of Petri Nets and Some Related Problems*, Report ECS-LFCS-93-261, Department of Computer Science, University of Edinburgh, 1993.

[19] S.R. Kosaraju, *Decidability of reachability in vector addition systems*, in Proc. 6th ACM STOC, 1982.

[20] S. Mayr, *An algorithm for the general Petri net reachability problem*, SIAM J. of Computing 13(3), pp.441-460, 1984.

[21] M.Minsky, **Finite and Infinite Machines**, Prentice Hall, Englewood Cliffs, N.Y., 1967.

[22] J.L. Peterson, **Petri Net Theory and the Modelling of Systems**, Prentice Hall, 1981.

[23] A.P.Sistla, M.Y.Vardi, P.Wolper, *The Complementation Problem for Büchi Automata With Applications to Temporal Logic*, Theoretical Computer Science No.49, 1987.

[24] M.Y. Vardi, P. Wolper, *An Automata-Theoretic Approach to Automatic Program Verification*, in Proc. of LICS, 1986.