Bull. Nov. Comp. Center, Comp. Science, 23 (2005), 113–136

Designing tableau-like axiomatization for Propositional Linear Temporal Logic at home of Arthur Prior*

N. V. Shilov

Abstract. Propositional Linear Temporal Logic (PLTL) is a very popular formalism for specification and verification of computer programs and systems. The paper suggests a tableau-like axiomatization for PLTL based on automata-theoretic decision procedure coupled with tableau for local model checking of the propositional μ-Calculus.

1. Introduction

Propositional Linear Temporal Logic (PLTL) is a very popular formalism for specification and verification of computer programs and systems [6, 18]. Fundamental results on decidability, model checking, and axiomatization for PLTL have become a part of the Computer Science classics [11, 20]. Automata-theoretic technique [22] has proved its utility for propositional modal logics of programs. In particular, PSPACE-completeness for PLTL has been proved in this technique [21]. Later the automata-theoretic approach has been extended to model checking [6]. The axiomatization issues for PLTL have been studied first on base of modal logic tradition [14]. But tableau and tableau-base decision procedure for PLTL have been developed with aid of automata-theoretic technique also [23].

After publication of [23], tableau for variants of linear temporal logic have been studied in a number of papers. A comprehensive study of tableaux for first-order temporal logics can be found in [15]. Tableau for combinations of temporal logics with other model logics became a research topic in recent years [10, 17].

At the same time a so called clausal resolution approach for axiomatization of different propositional temporal logic was under development. General outlines of this method has been presented in [12] for propositional linear temporal logic with future and past modalities and operators. The method is based on classical resolution augmented by temporal resolution rule. A special Separated Normal Form (SNF) has been defined for these purposes. For branching time temporal logic CTL a clausal resolution system has been developed in [1]. Algorithmical issues of CTL resolution

*This work is supported by Brain Korea 21 Project, The school of information technology, KAIST in 0000.

theorem prover have been discussed in [2]. The most recent development for branching time propositional logic is sound and complete clausal resolution system for ECTL$^+$ [4]. For PLTL a sound and complete clausal resolution has been developed in [13] and then improved in [9]. Connection between infinite automata and SNF has been examined in [3].

We develope in this paper sound and complete *experimental* tableau-like axiomatization for PLTL. This axiomatization comprises rewriting rules that simplify formulae (Table 1), and a 'tableau' (Table 5). A deduction strategy within the axiomatization consists of a number of stages. These stages are sketched below along with outlines of the rest of the paper. Basically our axiomatization is 'retrieved' from automata-theoretic decision procedure. Implementation of the presented axiomatization and similar tableau-like axiomatization for other propositional program and temporal logics are topics for further research.

Section 2 introduces the rewriting rules that eliminate negations outside literals and emulate subformulae of until constructs $\mathcal{U}$ by new propositional symbols. The rules preserve tautologies and lead to a so-called simple formulae.

Section 3 studies a special class of automata on infinite words. An automaton in this class accepts an infinite word as soon as it enters any accepting control state. A (fairness) constraint is a set of input symbols. An infinite word meets the constraint iff all specified symbols occur finite number of times at most. The halting (termination) problem with the constraint consists in checking whether an automaton accepts all infinite words that meet the constraint (if it is the case than we say that the automaton totally accept the constraint). Lemma 2 proves that the problem is decidable. In principle, the problem can be solved by Büchi automata, but we develop an original methods that is essential for axiomatization.

Section 4 translates simple formulae of PLTL into automata with fairness constraint. Control states of the automata are finite sets of formulae. The main property of this translation is captured in Lemma 3: a formula is a tautology iff the automaton totally accepts the constraint.

In Section 5, the automata are considered as finite labeled transition systems (i.e., Kripke structures) for the propositional μ-Calculus [16], and the halting problem with constrains is encoded by a particular formula of the μ-Calculus. An automaton totally accepts a constraint iff the formula holds in some initial state of the corresponding model (see Lemma 4). In simple words: we interpret halting problem with fairness constraint as the local model checking problem for some fixed formula of the propositional μ-Calculus.

The last Section 6 adopts sound and complete tableau designed for local model checking for the μ-Calculus in finite model [7] and convert it into a tableau-like axiomatization of PLTL.

Acknowledgement. Research was started in 2003, while author was a visiting Erskine Fellow at University of Canterbury, Christchurch, New Zealand. Arthur N. Prior (1914-1967), a founder of tense and temporal logics [8, 19], was Professor of Philosophy at this university in 1952–1956.

2. Propositional Linear Temporal Logic

Let *Prp* be an infinite set of propositional symbols.

Definition 1. The syntax of the Propositional Linear Time Logic (PLTL) consists of formulae that are defined by induction as follows:

- every propositional symbol is a formula;
- negation $(\neg\phi)$ is a formula;
- conjunction $(\phi \wedge \psi)$ and disjunction $(\phi \vee \psi)$ are formulae;
- nextime $(\circ\phi)$, always $(\Box\phi)$, and eventual $(\Diamond\phi)$ are formulae;
- until $(\phi\mathcal{U}\psi)$ and unless $(\phi\mathcal{W}\psi)$ are formulae.

Let $Prp(\xi)$ be a set of all propositional symbols that occur in a formula ξ. Every substring of a formula that is a formula as it is is said to be a subformula; let $Sub(\xi)$ be a set of all subformulae of a formula ξ. We exploit the standard abbreviations $(\phi \to \psi)$ and $(\phi \leftrightarrow \psi)$ for $((\neg\phi) \vee \psi)$ and for $((\phi \to \psi) \wedge (\psi \to \phi))$, respectively. Frequently we omit the most external parenthesis in small formulae and some parenthesis inside formulae in accordance with the standard rules of operation precedence: $\neg$, $\circ$, $\Box$, $\Diamond$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$.

Definition 2. (Bounded) linear structure is pairs of the form (V, B), where

- the bound $B \subseteq Prp$ is a finite set of propositional symbols,
- the valuation sequence V is a countable sequence $V_0 \subseteq B, \dots\ V_i \subseteq B, \dots$.

Definition 3. If (V, B) and (U, C) are two bounded linear structures such that $B \subseteq C$ and $V_i = (U_i \cap B)$ for every $i \geq 0$, then (U, C) is said to be a (semantics) extension of (V, B) on (propositional symbols in) $C \setminus B$.

Definition 4. A point is a triple of the form (V, B, i), where $i \geq 0$ is an integer and (V, B) is a bounded linear structure.

Definition 5. Semantics of PLTL is defined in terms of the satisfyability relation $\models$ between points and formulae by induction on formula structure[1]:

- $(V, B, i) \models p$ iff $p \in V_i$ for $p \in Prp$;
- $(V, B, i) \models (\neg\phi)$ iff $(V, B, i) \not\models \phi$;
- $(V, B, i) \models (\phi \wedge \psi)$ iff $(V, B, i) \models \phi$ and $(V, B, i) \models \psi$,
  $(V, B, i) \models (\phi \vee \psi)$ iff $(V, B, i) \models \phi$ or $(V, B, i) \models \psi$;
- $(V, B, i) \models (\circ\phi)$ iff $(V, B, i + 1) \models \phi$,
  $(V, B, i) \models (\Box\phi)$ iff $(V, B, j) \models \phi$ for every $j \geq i$,
  $(V, B, i) \models (\Diamond\phi)$ iff $(V, B, j) \models \phi$ for some $j \geq i$;
- $(V, B, i) \models (\phi\mathcal{U}\psi)$ iff $(V, B, k) \models \psi$ for some $k \geq i$ and
  $(V, B, j) \models \phi$ for every $j \in [i..k[$,
  $(V, B, i) \models (\phi\mathcal{W}\psi)$ iff $(V, B, i) \models (\Box\phi)$ or $(V, B, i) \models (\phi\mathcal{U}\psi)$.

A formula is said to be a tautology iff it holds in every point. Formulae are said to be equivalent iff they have equal semantics (i.e. for every point they hold or do not hold in the point simultaneously). As usual, formulae ϕ and ψ are equivalent iff the formula $(\phi \leftrightarrow \psi)$ is a tautology. A literal is a propositional symbol or its negation.

Definition 6. Normal formulae are formulae that do not use abbreviations $\rightarrow$ and $\leftrightarrow$, and that can use the negation in literals only. Simple formulae are normal formulae that do not use $\mathcal{W}$ operator and that can use $\mathcal{U}$ operator with propositional symbols only.

Every PLTL formula is equivalent to some normal formula due to standard 'normalizing' De-Morgan-like tautologies

$$\begin{array}{ll}
\neg\neg\phi \leftrightarrow \phi & \neg \circ \phi \leftrightarrow \circ\neg\phi \\
\neg\Box\phi \leftrightarrow \Diamond\neg\phi & \neg\Diamond\phi \leftrightarrow \Box\neg\phi \\
\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi & \neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi \\
\neg(\phi\mathcal{U}\psi) \leftrightarrow \neg\psi\mathcal{W}(\neg\phi \wedge \neg\psi) & \neg(\phi\mathcal{W}\psi) \leftrightarrow \neg\psi\mathcal{U}(\neg\phi \wedge \neg\psi)
\end{array}$$

The major disadvantage of this normalization is exponential space complexity (since it multiplies copies of some subformulae). We overcome this problem by shifting equivalence to metaequivalence as follows.

Definition 7. Formulae ϕ and ψ are metaequivalent ($\phi \overset{<}{\leftrightarrow} \psi$) iff for every point (V, B, i) the following holds:

$$\begin{array}{c}
(V, B, i) \models \phi \\
\Updownarrow \\
(U, C, i) \models \psi \text{ for every extension } (U, C) \text{ of } (V, B) \text{ on } C \setminus B.
\end{array}$$

[1]We assume $Prp(\phi) \cup Prp(\phi) \subseteq B$.

The equivalence implies the metaequivalence but not vice versa. The metaequivalence is not a symmetric, but metaequivalent formulae are tautologies or are not tautologies simultaneously.

A new (or fresh) propositional symbol is a symbol from infinite alphabet Prp that has not been used yet. Syntax substitution of some *object* instead of (any/all) instance(s) of some *target* in a *subject* is denoted by $subject^{object}_{target}$.

Lemma 1.
For every formula ξ, every its subformula θ that is out of scope of any negation in ξ, and every new propositional symbol p the formulae ξ and $\Box(\theta \to p) \to \xi^p_\theta$ are metaequivalent: $\xi \overset{<}{\leftrightarrow} \Box(\theta \to p) \to \xi^p_\theta$.

(Let us remark for justification that $\Box(\theta \to p)$ implies that p is interpreted as θ at least; at the same time ξ^p_θ is *monotonous* on p since p replaces *positive* instances of θ.)

Combining Lemma 1 with another tautology $\phi \mathcal{W} \psi \leftrightarrow \Box\phi \vee (\phi \mathcal{U} \psi)$, we can formulate the following lemma.

Lemma 2.
For every formula ξ, all its subformulae ϕ and ψ, and new propositional symbols p and q the following metaequivalences hold as soon as all substitutions are out of the range of any negation:

- $\xi \overset{<}{\leftrightarrow} \Box((\phi \to p) \wedge (\psi \to q)) \to \xi^{p\mathcal{U}q}_{\phi\mathcal{U}\psi}$,
- $\xi \overset{<}{\leftrightarrow} \Box((\phi \to p) \wedge (\psi \to q)) \to \xi^{\Box p \vee (p\mathcal{U}q)}_{\phi\mathcal{W}\psi}$.

Combining the normalizing equivalences with Lemma 2 we obtain the following proposition.

Proposition 1.
Every formula is metaequivalent to some simple formula that can be constructed in polynomial time by the rewriting system presented in Table 1.

3. Stuttering Automata with Fairness Constraint

Speaking informally, a stuttering automaton is just a nondeterministic finite automaton with the input alphabet consisting of subsets of some finite set, that can stutter (stay for awhile) on cells of the input tape.

$$\begin{array}{ll} \zeta \Rightarrow \zeta^{\phi}_{\neg\neg\phi} & \zeta \rightarrow \zeta^{\circ\neg\phi}_{\neg\circ\phi} \\ \zeta \Rightarrow \zeta^{\leftrightarrow\Diamond\neg\phi}_{\neg\Box\phi} & \zeta \Rightarrow \zeta^{\Box\neg\phi}_{\neg\Diamond\phi} \\ \zeta \Rightarrow \zeta^{\neg\phi\vee\neg\psi}_{\neg(\phi\wedge\psi)} & \zeta \Rightarrow \zeta^{\neg\phi\wedge\neg\psi}_{\neg(\phi\vee\psi)} \end{array}$$

$$\zeta \Rightarrow \Box((\neg\phi \rightarrow p) \wedge (\neg\psi \rightarrow q)) \rightarrow \zeta^{\Box p\vee(p\mathcal{U}(p\wedge q))}_{\neg(\phi\mathcal{U}\psi)}$$
$$\zeta \Rightarrow \Box((\neg\phi \rightarrow p) \wedge (\neg\psi \rightarrow q)) \rightarrow \zeta^{p\mathcal{U}(p\wedge q)}_{\neg(\phi\mathcal{W}\psi)}$$
$$\zeta \Rightarrow \Box((\phi \rightarrow p) \wedge (\psi \rightarrow q)) \rightarrow \zeta^{p\mathcal{U}q}_{\phi\mathcal{U}\psi}$$
$$\zeta \Rightarrow \Box((\phi \rightarrow p) \wedge (\psi \rightarrow q)) \rightarrow \zeta^{\Box p\vee(p\mathcal{U}q)}_{\phi\mathcal{W}\psi}$$

(where p and q are new propositional symbols)

Table 1. Rewriting rules for simplification

Definition 8. A stuttering automaton is a tuple $(Sts, Bnd, Ini, Fin, Prg)$, where

- Sts is a finite set of control states, Ini and Fin are subsets of Sts that comprise initial and final states;
- Bnd is a finite set, called the bound, while the powerset 2^{Bnd} is called the input alphabet;
- Prg is the program that consists of transitions of 2 types:
  - reading type $(q', S) \rightarrow q''$,
  - moving type $(q', S, \mathbf{next}) \rightarrow q''$,

  where $q', q'' \in Sts$, $S \subseteq Bnd$, and **next** is a special reserved symbol.

The stuttering automata work on countable sequences of input symbols. Every sequence of this kind can be thought as input tape consisting of cells (or positions) that are enumerated and that contain subsets of the bound set per cell. Every stuttering automaton has a pointer that every time points to a cell of the input tape and can remains in this position or move to the next cell to the right.

Definition 9. Let $(Sts, Bnd, Ini, Fin, Prg)$ be a stuttering automaton. A configuration of the automaton is a triple (W, i, q) that comprises an infinite word $W \in (2^{Bnd})^{\omega}$ (input tape), an integer $i \geq 0$ (pointer position), and a control state $q \in Sts$. We say that a configuration (W, i, q) contains the word W. A configuration is said to be initial iff $i = 0$ and $q \in Ini$ (i.e., the pointer is on the leftmost cell and the automaton is in an initial state). A configuration is said to be final iff $q \in Fin$ (the automaton is in a final state).

Definition 10. Let $(Sts, Bnd, Ini, Fin, Prg)$ be a stuttering automaton. The automaton shifts from one configuration to another in accordance with the program: it can shift from (W, i, q') to

- (W, i, q'') iff $(q', W_i) \to q'' \in Prg$;
- $(W, (i+1), q'')$ iff $(q', W_i, \mathbf{next}) \to q'' \in Prg$.

Definition 11. For a given stuttering automaton $\mathcal{A}$ shifting is a binary relation $\to_{\mathcal{A}}$ on its configurations: for every pair of configurations c' abd c'', we write $c' \to_{\mathcal{A}} c''$ iff $\mathcal{A}$ can shift from c' to c''. We denote by $\to^*_{\mathcal{A}}$ the reflexive and transitive closure of the binary relation $\to_{\mathcal{A}}$.

Definition 12. A stuttering automaton $\mathcal{A}$ accepts an infinite word W (written in its input alphabet) iff there are an initial configuration c' and a final configuration c'' such that c' and c'' both contain the word W and $c' \to^*_{\mathcal{A}} c''$. The language of the automaton $\mathcal{L}(\mathcal{A})$ comprises all infinite words that the automaton accepts. The automaton $\mathcal{A}$ is said to be total iff $\mathcal{L}(\mathcal{A})$ comprises all infinite words $(2^B)^\omega$.

Now we are going to define a very important notion of fairness constraint.

Definition 13. Let D be some set, $seq \in D^\omega$ be an infinite sequence of elements in D, and $(\mathcal{P} : D \to Bool)$ be some property of elements in D. The sequence seq is said to be fair with respect to the property $\mathcal{P}$ iff infinitely many elements of the sequence seq enjoy this property. In contrast, we say that the sequence seq meets a fairness constrain $\mathcal{P}$ iff the property $\mathcal{P}$ holds finitely often at most in seq (i.e., $\mathcal{P}$ does not hold after some point in seq).

We are most interested in some fairness constraint for stuttering automata.

Definition 14. Let Bnd be a finite set and $S \subseteq Bnd$. A competence property $\mathcal{P}_S$ is the following property $\lambda T \subseteq Bnd.\ (S \cap T \neq \emptyset)$.

The following lemma is straightforward[2].

Lemma 3. *Let Bnd be a finite set, $S \subseteq Bnd$, and $W \in (2^{Bnd})^\omega$ be an infinite word. Then W meets fairness constraint $\mathcal{P}_S$ iff all propositional symbols in S disappear after some position in W.*

[2]It explains the title 'competence property' for $\mathcal{P}_S$: the fairness constraint $\mathcal{P}_S$ grants full rights to elements in $Bnd \setminus S$ while imposes right limitations for elements in S. The former can occur infinitely often, the later - finitely often at most.

Definition 15. Let $(Sts, Bnd, Ini, Fin, Prg)$ be a stuttering automaton, $S \subseteq Bnd$ and $\mathcal{P}_S$ be corresponding competence property. We say that the automaton $\mathcal{A}$ totally meets the fairness constraint $\mathcal{P}_S$ iff language $\mathcal{L}(\mathcal{A})$ includes all infinite words $W \in (2^B)^\omega$ that meet the constraint $\mathcal{P}_S$.

In conjunction with Lemma 3 it immediately implies the following lemma.

Lemma 4. *Let $(Sts, Bnd, Ini, Fin, Prg)$ be a stuttering automaton, $S \subseteq Bnd$, and $\mathcal{P}_S$ be corresponding competence property. The automaton totally meets the fairness constraint $\mathcal{P}_S$ iff it accepts every infinite word in $(2^{Bnd})^\omega$ that contains finite (at most) number of instances of symbols in S.*

Definition 16. The total (halting, termination) problem for stuttering automata with competence fairness constraint is to check for input stuttering automaton $\mathcal{A}$ and input competence property $\mathcal{P}_S$ whether the automaton $\mathcal{A}$ totally meets the fairness constraint $\mathcal{P}_S$.

There are several opportunities to decide the total problem for stuttering automata with competence fairness constraint. The most high level but indirect approach can exploit the formalism of Büchi automata and the decidability of the emptiness problem for these automata. In contrast, let us present below a variant of a low level but direct decision procedure.

Proposition 2. *The total problem for stuttering automata with competence fairness constraint is decidable in time exponential in the size of the automaton.*

Proof. (Sketch.) Let $\mathcal{A} = (Sts, Bnd, Ini, Fin, Prg)$ and $\mathcal{P}_S$ be particular stuttering automaton and competence property. Let M be a set of control states that occur in the right-hand side of moving transitions.

First, for every $T \subseteq B$ let us define a binary relation $\overset{T}{\rightsquigarrow}$ on $Sts \times (M \cup Fin)$:

$$q' \overset{T}{\rightsquigarrow} q''$$
iff
$$q' \equiv q'' \in Fin \text{ or}$$
there is some sequence of control states $q_0, \dots\ q_n$ $(n \geq 0)$ such that
$$q' = q_0,\ (q_0, T) \to q_1 \in Prg,\ \dots\ (q_{n-1}, T) \to q_n \in Prg, \text{ and}$$
$$(q_n, T, \mathbf{next}) \to q'' \in Prg.$$

(I.e., on final states the relation $\overset{T}{\rightsquigarrow}$ does not evolve further, and on non-final states $\overset{T}{\rightsquigarrow}$ it is the reflexive and transitive closure of T-readings up to the first T-move.)

Then let us extend $\overset{T}{\rightsquigarrow}$ on $2^{Sts} \times 2^{M \cup Fin}$ as follows:

$$Q' \overset{T}{\leadsto} Q''$$
$$\text{iff}$$
$$\text{for every } q'' \in Q'' \text{ there is some } q' \in Q' \text{ such that } q' \overset{T}{\leadsto} q''.$$

The decision procedure is given in Table 3 (third column) in terms of $\overset{T}{\leadsto}$ on $2^{Sts} \times 2^{M \cup Fin}$. Please refer to Section 6.1 in the Appendix for correctness of this procedure. Exponential complexity bound follows immediately from monotonicity arguments (since $H_0 \subseteq H_1 \subseteq ...H_i \subseteq H_{i+1} \subseteq ... \subseteq Sts$ and $Sts \supseteq G_0 \supseteq G_1 \supseteq ...G_i \supseteq G_{i+1} \supseteq ... \supseteq$).

4. PLTL and stuttering automata

Below we present the algorithm of translation of simple formulae of PLTL to stuttering automata with fairness constraint. The algorithm inputs a simple formula ξ and a set of propositional symbols B that includes all symbols that occur in ξ, and outputs a stuttering automaton $\mathcal{A}(\xi, B)$ and two sets of new propositional symbols $F(\xi, B)$ and $G(\xi, B)$. The description of the algorithm follows.

Let ξ be a simple formula and $B \supseteq Prp(\xi)$ be a set of propositional symbols. Let us enumerate all instances of conjunctions $\wedge$ and modalities $\Box$ within ξ; it gives us an opportunity to index conjunctions and always modalities and refer every particular instance by the assigned index. Let C and A be set of indexes assigned to conjunctions and to $\Box$ modalities by this enumeration. Let us assign a new propositional symbol f_c for every conjunction $\wedge_c$ ($c \in C$) and a new propositional symbol g_a for every modality $\Box_a$ ($a \in A$) within ξ. Let $F(\xi, B)$ be $\{f_c : c \in C\}$ and $G(\xi, B)$ be $\{g_a : a \in A\}$.

The control states Sts of resulting automaton are subformulae of the formula ξ extended by a special state **accept**. The initial state is the formula ξ, the final state is **accept**: $Ini = \{\xi\}$ and $Fin = \{\textbf{accept}\}$. The input alphabet bound Bnd is $B \cup F(\xi, B) \cup G(\xi, B)$, i.e., the input alphabet comprises all sets of propositional symbols in B and new propositional symbols associated with conjunctions and $\Box$ modalities in the formula ξ. The program Prg of the automaton $\mathcal{A}(\xi, B)$ is presented in Table 2 in structured and unstructured forms. In structured formalism, we exploit **;** for sequencing, **U** for non-deterministic choice, ***** for non-deterministic iteration, **if** $-$**then** $-$**else** for deterministic choice, **while** $-$**do** for deterministic iteration. We also use tests of two kinds: if p is a propositional symbol then the test $p?$ means the $p \in W_{now}$ and the test $\neg p?$ means the $p \notin W_{now}$, where $W_{now} \subseteq B \cup F(\xi, B) \cup G(\xi, B)$ is the currently reading position of the input word W.

$(p, S) \to$ **accept** iff $p \in S$ $(\neg p, S) \to$ **accept** iff $p \notin S$	$P(p) = p?$ **; accept** $P(\neg p) = \neg p?$ **; accept**
$(\circ\phi, S, \mathbf{next}) \to \phi$	$P(\circ\phi) =$ **next ;** $P(\phi)$
$(\Box_a\phi, S, \mathbf{next}) \to \Box_a\phi$ iff $g_a \in S$ $(\Box_a\phi, S) \to \phi$ iff $g_a \notin S$	$P(\Box_a\phi) =$ **while** $g_a?$ **do next ;** $P(\phi)$
$(\Diamond\phi, S) \to \phi$ $(\Diamond\phi, S, \mathbf{next}) \to \Diamond\phi$	$P(\Diamond\phi) =$ **next* ;** $P(\phi)$
$(\phi \wedge_c \psi, S) \to \phi$ iff $f_c \in S$ $(\phi \wedge_c \psi, S) \to \psi$ iff $f_c \notin S$	$P(\phi \wedge_c \psi) =$ **if** f_c **then** $P(\phi)$ **else** $P(\psi)$
$(\phi \vee \psi, S) \to \phi$ $(\phi \wedge \psi, S) \to \psi$	$P(\phi \vee \psi) = P(\phi)$ **U** $P(\psi)$
$(p\mathcal{U}q, S) \to q$ $(p\mathcal{U}q, S, \mathbf{next}) \to p\mathcal{U}q$ iff $p \in S$	$P(p\mathcal{U}q) = (p?$ **; next)* ;** $q?$ **; accept**

Table 2. Automaton program and its structured flowchart

Lemma 5. *Let ξ be a simple formula and $B \supseteq Prp(\xi)$ be a set of propositional symbols. Let $F = F(\xi, B)$ and $G = G(\xi, B)$ be two sets of new propositional symbols constructed in accordance with the translation algorithm. Then for every subformula θ of the formula ξ, for every point (V, B, i) the following holds:*

$$(V, B, i) \models \theta$$

iff

the stuttering automaton

$$(\overbrace{(\{accept\} \cup Sub(\theta))}^{control\ states}, \overbrace{(B \cup F \cup G)}^{bound}, \overbrace{\{\theta\}}^{Ini\ states}, \overbrace{\{\mathbf{accept}\}}^{Fin\ states}, \overbrace{P(\theta)}^{program})$$

starting in position i accepts every extension $(V,\ B \cup F \cup G)$ of (V, B) on propositional symbols in $F \cup G$ that meets the fairness constraint $\mathcal{P}_G$.

(In simple words: θ holds in the point (V, B, i) iff the corresponding automaton accepts every word $(V_i \cup F_i \cup G_i)$, $(V_{i+1} \cup F_{i+1} \cup G_{i+1})$, $\ldots$, where F_i, $F_{i+1}, \cdots \subseteq F$ and G_i, $G_{i+1}, \cdots \subseteq G$ but $\emptyset = G_j = G_{j+1} = \ldots$ for some $j \geq i$.)

Please refer to Section 6.2 in the Appendix for the proof. In turn, the above Lemma 5 immediately implies the following proposition.

Proposition 3. *Let ξ be a simple formula and $B \supseteq Prp(\xi)$ be a set of propositional symbols. Let $F = F(\xi, B)$ and $G = G(\xi, B)$ be two sets of new propositional symbols constructed in accordance with the translation algorithm. Then the following holds:*

ξ *is a tautology*
iff
the stuttering automaton
$((\{accept\} \cup Sub(\theta))\ ,\ (B \cup F \cup G)\ ,\ \{\theta\}\ ,\ \{\mathbf{accept}\}\ ,\ P(\theta))$
totally accepts the fairness constraint $\mathcal{P}_G$.

By the way, Propositions 1, 2, and 3 imply a well-known upper bound for decidability of PLTL.

Corollary 1. *PLTL is decidable in exponential time.*

5. Model checking interpretation

Let $\{true, false\}$ be Boolean constants, Var and Act be disjoint finite alphabets of propositional and action variables respectively. For avoiding ambiguities, we assume that these new alphabets Var and Act are disjoint with the alphabet of propositional symbols Prp that are in use in PLTL.

Definition 17. The syntax of the propositional μ-Calculus (μC) consists of formulae that are defined by induction as follows:

- every propositional variable is a formula;
- negation $(\neg\phi)$ is a formula;
- conjunction $(\phi \wedge \psi)$ and disjunction $(\phi \vee \psi)$ are formulae;
- $(\langle a\rangle\phi)$ and $([a]\phi)$ are formulae for every $a \in Act$;
- the least $(\mu x\ .\ \phi)$ and the greatest $(\nu x\ .\ \phi)$ fixpoints are formulae for every $x \in Var$ without instances in the range of odd amount of negations in ϕ.

As in the PLTL framework, sometimes we omit the most external parenthesis in small formulae and some parenthesis inside formulae in accordance with the standard rules of boolean operation precedence. We also use syntax substitution $subject^{object}_{target}$ and another related metanotation: let $subject^{0}_{target}(object)$ stay for the $object$, and for let $subject^{n+1}_{target}(object)$ stay for $subject^{subject^{n}_{target}(object)}_{target}$ every $n \geq 0$.

Semantics of μC is defined in models that are called labeled transition systems in Computer Science or Kripke structures in modal logic tradition. For avoiding ambiguities with PLTL, we use terminology and notation of modal logic for reasoning about μC.

Definition 18. Every model M is a triple (D, R, E), where the universe $D \neq \emptyset$ consists of worlds, the interpretation $R : Act \to 2^{D\times D}$ assigns a binary relation $R(a) \subseteq D \times D$ to every action variable a, the evaluation $E : Var \to 2^D$ assigns a monadic predicate $E(x) \subseteq D$ to every propositional variable x.

Models can be infinite, but we are most interested in finite models only.

Definition 19. Semantics of μC is a ternary validity relation $|\models$ between worlds, models, and formulae. This relation is defined by induction on formulae structure. In finite models, it can be defined as follows:

- $w|\models_M true$ and $w|\not\models_M false$,
  $w|\models_M x$ iff $w \in E(x)$ for $x \in Var$;
- $w|\models_M (\neg\phi)$ iff $w|\not\models_M \phi$;
- $w|\models_M (\phi \wedge \psi)$ iff $w|\models_M \phi$ and $w|\models_M \psi$,
  $w|\models_M (\phi \vee \psi)$ iff $w|\models_M \phi$ or $w|\models_M \psi$;
- $w|\models_M ([a]\phi)$ iff $u|\models_M \phi$ for every u such that $(w, u) \in R(a)$,
  $w|\models_M (\langle a\rangle\phi)$ iff $u|\models_M \phi$ for some u such that $(w, u) \in R(a)$;
- $w|\models_M \nu x.\phi$ iff $w|\models_M \phi^n_x(true)$ for all $n \geq 0$,
  $w|\models_M \mu x.\phi$ iff $w|\models_M \phi^n_x(false)$ for some $n \geq 0$.

(Informally speaking, $\nu x.\phi$ and $\mu x.\phi$ are abbreviations for infinite conjunction $\bigwedge_{n\geq 0} \phi^n_x(true)$ and infinite disjunction $\bigvee_{n\geq 0} \phi^n_x(false)$, respectively.)

Model checking is testing a model against a formula. The global checking problem consists in calculation of the set $M(\xi)$ of all worlds of the input model M, where the input formula ξ is valid. The local checking problem consists in testing the validity $w|\models_M \xi$ of the input formula ξ in the input world w in the input model M. We are most interested model checking of finite models.

Definition 20. Let x, y, $Nice$, and Fin be fixed propositional variables, and $A, B \subseteq Act$ be sets of action variables. Then let

- $GOOD$ be formula $(Nice \wedge \bigwedge_{b\in B}\langle b\rangle y)$,
- $HALT$ be formula $(Fin \vee x \vee \bigwedge_{a\in A}\langle a\rangle x)$,
- $PROVER$ be formula $(\nu\ y.GOOD)^{(\mu\ x.HALT)}_{Nice}$.

Since Definition 20 is obscure, let us explane the formulae were provided. Formula $\nu y.GOOD$ states that for every infinite sequence w of actions from B, there is a path labeled by w on which $Nice$ holds at each state. In $PROVER$, $Nice$ is $\mu x.HALT$, which states that every infinite path labeled

Stages	Model checking	Decision procedure
Stage 1	$H_0 := E(Fin)$; $i := 0$; **do** $H_{i+1} := H_i \cup$ $\cup\Big(\bigcap_{a\in A}\big(R(a)\big)^{-}(H_i)\Big)$ **until** $H_i \neq H_{i+1}$; $H := H_i$;	$H_0 = \{Q' : Q' \subseteq Fin\}$, $H_{i+1} = H_i \cup \{Q' :$ for every $T \subseteq (B \setminus S)$ there is $Q'' \in H_i$ that $Q' \overset{T}{\rightsquigarrow} Q''\}$ for every $i \geq 0$, $H = \bigcup_{i\geq 0} H_i$;
Stage 2	$G_0 := H$; $i := 0$; **do** $G_{i+1} := G_i \cap$ $\cap\Big(\bigcap_{b\in B}\big(R(b)\big)^{-}(G_i)\Big)$ **until** $G_i \neq G_{i+1}$; $G := G_i$;	$G_0 = H$, $G_{i+1} = \{Q' \in G_i :$ for every $T \subseteq B$ there is $Q'' \in G_i$ that $Q' \overset{T}{\rightsquigarrow} Q''\}$ for every $n \geq 0$, $G = \bigcap_{i\geq 0} G_i$;
Stage 3	$M(PROVER) := G$	$\mathcal{A}$ totally meets the constraint $\mathcal{P}_S$ iff there exists $Good \in G$ that $Good \subseteq Ini$ and $Good \neq \emptyset$.

Table 3. Model checking and decision procedure

with actions from A only eventually reaches Fin. So $PROVER$ states that any infinite sequence of actions from B labels some path in such a way that if at any point the remaining actions all fall into A, then the path eventually reaches Fin.

Lemma 6. *Let $M = (D, R, E)$ be a finite model. Then global checking of formula $PROVER$ in M can carried out by the algorithm*[3] *in Table 3 (second column).*

(For justification let observe that the algorithm is just a straightforward specialization of semantics of μC for formula $PROVER$ in finite models.)

We are especially interested in the model checking $PROVER$ in models that are generated by stuttering automata with constraint.

[3]We exploit inverse for binary relations and inverse images: if r is a binary relation on a set D with a subset $S \subseteq D$, then $r^{-} = \{(d', d'') : (d'', d') \in r\}$ is the inverse of r and $r^{-}(S) = \{d' \in D : (d', d'') \in r$ for some $d'' \in S\}$ is the inverse image of S.

Definition 21. Let $\mathcal{A} = (Sts, Bnd, Ini, Fin, Prg)$ and $\mathcal{P}_S$ be a stuttering automaton and a competence property. Let $\mathcal{M}(\mathcal{A}, \mathcal{P}_S)$ be the following finite model $(\mathcal{D}, \mathcal{R}, \mathcal{E})$. The universe $\mathcal{D}$ is 2^{Sts}, i.e., comprises all sets of control states. The alphabet *Act* of action variables consists of subsets of the bound *Bnd*, i.e., coincides with the input alphabet of the automaton: $Act = 2^{Bnd}$. For every $T \in Act$ let $\mathcal{R}(T)$ be $\overset{T}{\leadsto}$. The alphabet of propositional variables Var includes one 'special' variable Fin and two auxiliary variables x, y. The most important is evaluation of the special variable: $\mathcal{E}(Fin) = 2^{Fin}$, i.e., it comprises all subsets of the final sets.

Definition 22. Let $\mathcal{A} = (Sts, Bnd, Ini, Fin, Prg)$ and $\mathcal{P}_S$ be a stuttering automaton and a competence property. The model checking interpretation for the total problem for the automaton $\mathcal{A}$ with the fairness constraint $\mathcal{P}_S$ consists in the finite model $\mathcal{M}(\mathcal{A}, \mathcal{P}_S)$ and two sets 2^{Bnd} and $2^{Bnd \setminus S}$ adopted as $A \subseteq Act$ and $B \subseteq Act$ in the formula $PROVER$.

The importance of the model checking interpretation follows from the following Proposition 4. The correctness of this proposition immediately follows from comparison of the second and the third columns of Table 3.

Proposition 4.

A stuttering automaton $\mathcal{A}$ totally accepts a fairness constraint $\mathcal{P}_S$
iff
$Q \models_{\mathcal{M}(\mathcal{A},\mathcal{P}_S)} PROVER$ for some set Q of initial control states of $\mathcal{A}$.

6. Axiomatization via local model checking

There are two papers [7, 5] that have suggested tableau for local model checking of formulae of the propositional μ-Calculus. The first cited paper addresses finite state systems, while the second one deals with infinite systems as well as finite. We prefer system from [7] since we are interested in model checking interpretation of the halting problem with fairness constraint, i.e., we are bound by finite models $\mathcal{M}$ single formula $PROVER$. In the following paragraphs, we sketch the approach, the tableau, and soundness and completeness results from [7].

There are only 2 syntax differences between variants of the propositional μ-Calculus, that was discussed in Section 5 and that is in use in [7]. First, the set of propositional variable in [7] is divided on two disjoint sets: the variables that cannot be bound by fixpoints and the variables that have to be bound by fixpoints; the former we refer as model constants, the later – as model variables. Next, [7] exploits syntax without $\wedge$, $[\ ]$, and μ. Both variants enjoy equal expressive power due to standard De-Morgan-like tautologies: $\phi \wedge \psi \leftrightarrow \neg(\neg\phi \vee \neg\psi)$, $[a]\phi \leftrightarrow \neg\langle a\rangle\neg\psi$, $\mu x.\psi \leftrightarrow \neg(\nu x.(\psi_x^{\neg x}))$. For

$$\begin{tikzcd}
 & & & \nu y.\neg\psi(y) \arrow[d] & & & \\
 & & & \neg\psi(y) \arrow[d] & & & \\
 & & & \psi(y) \arrow[dl] \arrow[dr] & & & \\
 & & \nu x.\neg\phi(x) \arrow[d] & & \neg(\langle T\rangle y) \ \substack{\text{for every}\\ T\subseteq Bnd} \arrow[dr] & & \\
 & & \neg\phi(x) \arrow[dll] \arrow[d] \arrow[dr] & & & \langle T\rangle y \arrow[dr] & \\
\neg(\bigvee_{T\subseteq Bnd\setminus S}(\neg\langle T\rangle\neg x)) \arrow[d] & & Fin & \neg x \arrow[dr] & & & y \\
\bigvee_{T\subseteq Bnd\setminus S}(\neg\langle T\rangle\neg x) \arrow[d] & & & & x & & \\
\neg\langle T\rangle\neg x \ \substack{\text{for every}\\ T\subseteq Bnd\setminus S} \arrow[rr] & & \langle T\rangle\neg x \arrow[uur] & & & &
\end{tikzcd}$$

Figure 1. Subformulae of $\nu PROVER$

avoiding ambiguities, we distinguish these two variants and refer variant from [7] by acronym νC.

In particular, in the formula $PROVER$ propositional variable Fin is a modal constant and propositional auxiliary variables x and y are modal variables. νC-representation of this formula is the following formula $\nu PROVER$:

$$\nu y.\neg\Bigg(\underbrace{\nu x.\neg\Bigg(\overbrace{Fin\vee\neg x\vee\neg\Big(\bigvee_{T\subseteq Bnd\setminus S}\neg(\langle T\rangle\neg x)\Big)}^{\phi(x)}\Bigg)\vee\Big(\bigvee_{T\subseteq Bnd}\neg(\langle T\rangle y)\Big)}_{\psi(y)}\Bigg).$$

The subformula relation ('strict subterm' in [7]) $\prec$ is treated in pure syntax manner like in PLTL: a subformula is a substring of the formula that is a formula itself. The complete graph of the immediate subformula relation for subformulae of $\nu PROVER$ is depicted in Figure 1.

The tableau from [7] is represented in Table 4. We only turned upside-down all rules so that the goals are under subgoals (since we prefer proof-search tree to grow upward). The proof rules operate on sequents of the form $(H \vdash_M w \in \xi)$, where ξ is a formula of νC, M is a finite model, w is a world and H is a set of hypothesis (or assumptions) of the form $(u : \theta)$, where u is a world and θ is a closed fixpoint formula. [7] has proved the general soundness and completeness results (Theorems 4.18 and 4.19). We would like to summarize them both in the following corollary.

Axiom Schemata	
$H \vdash_M w \in p$ iff p is a modal constant and $w \in E(p)$	
$H \vdash_M w \in \neg p$ iff p is a modal constant and $w \notin E(p)$	
$H \vdash_M w \in \neg(\langle a \rangle \xi)$ iff $\{v : (w, v) \in R(a)\}$ is empty set $\emptyset$	
$H \vdash_M w \in \nu x.\xi$ iff $(w : \nu x.\xi) \in H$	
Inference Rules	
$\frac{H \vdash_M w \in \xi}{H \vdash_M w \in \neg\neg\xi}$	$\frac{H \vdash_M w \in \neg\xi \ , \ H \vdash_M w \in \neg\theta}{H \vdash_M w \in \neg(\xi \vee \theta)}$
$\frac{H \vdash_M w \in \xi}{H \vdash_M w \in \xi \vee \theta}$	$\frac{H \vdash_M w \in \theta}{H \vdash_M w \in \xi \vee \theta}$
$\frac{H \vdash_M u \in \xi}{H \vdash_M w \in \langle a \rangle \xi}$ for $u \in \{v : (w, v) \in R(a)\}$	
$\frac{H \vdash_M u_1 \in \neg\xi \ ,\dots \ H \vdash_M u_n \in \neg\xi}{H \vdash_M w \in \neg(\langle a \rangle \xi)}$ where $\{u_1, \dots u_n\} = \{v : (w, v) \in R(a)\}$	
$\frac{H' \cup \{(w : \nu x.\xi)\} \vdash_M w \in \xi_x^{\nu x.\xi}}{H \vdash_M w \in \nu x.\xi}$ where $(w : \nu x.\xi) \notin H$ and $H' = H \setminus \{(u : \theta) : \nu x.\xi \prec \theta\}$	
$\frac{H' \cup \{(w : \nu x.\xi)\} \vdash_M w \in \neg\xi_x^{\nu x.\xi}}{H \vdash_M w \in \neg(\nu x.\xi)}$ where $(w : \nu x.\xi) \notin H$ and $H' = H \setminus \{(u : \theta) : \nu x.\xi \prec \theta\}$	

Table 4. Sound and complete system for local model checking from [7]

Corollary 2. *For every formula ξ of νC, for every finite model M and every world w within this model the following holds: $\emptyset \vdash_M w \in \xi$ iff $w \models_M \xi$.*

Our tableau-like deduction system is presented in Table 5. This system is bound for a target simple PLTL formulae to be proved η. In this system:

- W and U range over sets of subformulae of η extended by label **accept**;
- H and H' range over collections of assumptions in the form $(W : \nu y.\neg\psi(y))$ or $(U : \nu x.\neg\phi(x))$, where $\nu y.\neg\psi(y)$ and $\nu x.\neg\phi(y)$ are the only two closed formulae in Figure 1;
- T ranges over sets of $Prp(\eta) \cup F(\eta, Prp(\eta)) \cup G(\eta, Prp(\eta))$ (i.e., are sets of propositional symbols, conjunctions and always-modalities in η);
- $\overset{T}{\leadsto}$ is the binary relation on sets of subformulae of η extended by label **accept**, derived from Table 2;
- ξ and θ range over νC-formulae in Figure 1 with $Bnd = (Prp(\eta) \cup F(\eta, Prp(\eta)) \cup G(\eta, Prp(\eta))$ and $S = G(\eta, Prp(\eta))$;
- $\prec$ is the subormula relation in Figure 1.

At last we are ready to combine Propositions 1, 2, 3, and 4 with Corollary 2 and prove our soundness and completeness theorem.

Axiom Schemata
$H \vdash W \in Fin$ iff $W = \{\textbf{accept}\}$
$H \vdash W \in \neg(\langle T\rangle\xi)$ iff $\{U : W \overset{T}{\rightsquigarrow} U\}$ is the emptyset
$H \vdash W \in \nu z.\xi$ iff $(W : \nu z.\xi) \in H$
Inference Rules
$\frac{H\vdash W\in\xi}{H\vdash W\in\neg\neg\xi}$ $\frac{H\vdash W\in\neg\xi\ ,\ H\vdash W\in\neg\theta}{H\vdash W\in\neg(\xi\vee\theta)}$ $\frac{H\vdash W\in\xi}{H\vdash W\in\xi\vee\theta}$ $\frac{H\vdash W\in\theta}{H\vdash W\in\xi\vee\theta}$
$\frac{H\vdash U\in\xi}{H\vdash W\in\langle T\rangle\xi}$ for some U that $W \overset{T}{\rightsquigarrow} U$
$\frac{H\vdash U_1\in\neg\xi\ ,\ldots\ H\vdash U_n\in\neg\xi}{H\vdash W\in\neg(\langle T\rangle\xi)}$ where $\{U_1, \ldots U_n\} = \{U : W \overset{T}{\rightsquigarrow} U\}$
$\frac{H'\cup\{(W:\nu z.\xi)\}\vdash W\in\xi_z^{\nu z.\xi}}{H\vdash W\in\nu z.\xi}$ $\frac{H'\cup\{(W:\nu z.\xi)\}\vdash W\in\neg\xi_z^{\nu z.\xi}}{H\vdash W\in\neg(\nu z.\xi)}$ where $(W : \nu z.\xi) \notin H$ and $H' = H \setminus \{(U : \theta) : \nu z.\xi \prec \theta\}$

Table 5. Sound and complete system for simple formulae of PLTL

Theorem 1.

- *For all formulae ζ and η of PLTL, if ζ can be transformed to η by rewriting rules in Table 1, then ζ is a tautology iff η is a tautology.*
- *Every formula ζ of PLTL, can be transformed by rewriting rules in Table 1 to some simple formula η of PLTL.*
- *For every simple formula η of PLTL the following holds: η is a tautology iff the sequent $(\emptyset \vdash \{\eta\} \in \nu PROVER)$ is provable in the system presented in Table 5.*

References

[1] Bolotov A., Fisher M. A Resolution method for CTL branching-time temporal logic // Proc. of the Fourth Intern. Workshop on Temporal Representation and Reasoning (TIME). — IEEE Press, 1997. — P. 20–27.

[2] Bolotov A., Dixon C. Resolution for branching-time temporal logics: Applying the temporal resolution rule // Proc. of the Seventh Intern. Workshop on Temporal Representation and Reasoning (TIME'00). — IEEE Computer Press, 2000. — P. 163–172.

[3] Bolotov A., Fisher M., Dixon C. On the relationship between omega-automata and temporal logic normal forms // J. of Logic and Computation. — 2002. — Vol. 12, N 4. — P. 561–581.

[4] Bolotov A., Basukoski A. Clausal resolution for extended computation tree logic ECTL$^+$ // Proc. of the Time 2004 Intern. Conf. on Temporal Representation and Reasoning. — IEEE Computer Press, 2004.

[5] Bradfield J., Stirling C. Local model checking for infinite state spaces // Theor. Comput. Sci. — 1992. — Vol. 96. — P. 157–174.

[6] Clarke E.M., Grumberg O., Peled D. Model Checking. — MIT Press, 1999.

[7] Cleaveland R. Tableau-based model checking in the propositional mu-calculus // Acta Informatica. — 1990. — Vol. 27. — P. 725–747.

[8] Copeland J. Logic and Reality: Essays on the Legacy of Arthur Prior. — Clarendon Press and Oxford University Press, 1996.

[9] Degtyarev A., Fisher M., Konev B. A simplified clausal resolution procedure for propositional linear-time temporal logic // Proc. TABLEAUX-02. — Lect. Notes in Comput. Sci. — 2002. — Vol. 2381. — P. 85–99.

[10] Dixon C., Nalon C., Fisher M. Tableaux for temporal logics of knowledge: synchronous systems of perfect recall or no learning // Proc. of TIME-ICTL 2003. — IEEE CS Press, 2003.

[11] Emerson E.A. Temporal and Modal Logic // Handbook of Theor. Comput. Sci. Vol. B. — Elsilver and The MIT Press, 1990. — P. 995–1072.

[12] Fisher M. A resolution method for temporal logic // Proc. of Twelfth Intern. Joint Conf. on Artificial Intelligence (IJCAI), Sydney, Australia, August 1991. — Morgan Kaufmann, 1991.

[13] Fisher M., Dixon C., Peim M. Clausal temporal resolution // ACM Trans. on Computational Logic. — 2001. — Vol. 2, N 1. — P. 12–56.

[14] Gabbay D.M., Pnueli A., Shelah S., Stavi J. On the temporal analysis of fairness // 7^{th} ACM Sympos. on Principles of Programming Languages, Las Vegas, 1980. — P. 163-173.

[15] Kontchakov R., Lutz C., Wolter F., Zakharyaschev M. Temporal tableaux // Studia Logica. — 2004. — Vol. 76, N 1. — P. 91-134.

[16] Kozen D. Results on the propositional mu-calculus // Theor. Comput. Sci. — 1983. — Vol. 27. — P. 333-354.

[17] Lutz C., Sturm H., Wolter F., Zakharyaschev M. Tableaux for temporal description logic with constant domain // Proc. of the Internat. Joint Conf. on Automated Reasoning IJCAR-01. — Lect. Notes in Artifical Intelligence. — 2001. — Vol. 2083. — P. 121-136.

[18] Manna Z., Pnueli A. The Temporal Logic of Reactive and Concurrent Systems. — Springer-Verlag, 1991.

[19] Prior A.N. Time and Modality. — Oxford: Clarendon Press, 1957.

[20] Stirling C. Modal and Temporal Logics // Handbook of Logic in Comput. Sci. — Claredon Press, 1992. — Vol. 2. — P. 477–563.

[21] Vardi M.Y. An automata-theoretic approach to linear temporal logic // Logics for Concurrency: Structure versus Automata. — Lect. Notes in Comput. Sci. — 1996. — Vol. 1043. — P. 238-266.

[22] Vardi M.Y., Wolper P. Automata-theoretic techniques for modal logic of programs // J. Comput. and System Sci. — 1986. — Vol. 32. — P. 183-221.

[23] Wolper P. The tableau method for temporal logic: An overview // Logique et Analyse. — 1985. — Vol. 28. — P. 119–136.

Appendix

6.1. Proof details for Proposition 2

First let us construct the set H of all sets of control states $Q' \subseteq Sts$ such that for every infinite word $W \in (2^{Bnd})^\omega$, for every $i \geq 0$, if $W_j \cap S = \emptyset$ for every $j \geq i$ then $(W, i, q') \to^*_{\mathcal{A}} (W, j, q'')$ for some $q' \in Q'$, $q'' \in Fin$ and $j \geq i$ (i.e., starting with some state q' in Q' the automaton eventually accepts the suffix $W_i W_{i+1} \dots$ of the word W as soon as no symbol from S occurs in the suffix). This set H can be constructed as $\cup_{n \geq 0} H_n$, where $H_0 \subseteq H_1 \subseteq \dots H_n \subseteq H_{n+1} \subseteq \dots$ is non-decreasing sequence of sets H_n ($n \geq 0$) of all sets of control states $Q' \subseteq Sts$ such that for every infinite word $W \in (2^{Bnd})^\omega$, for every $i \geq 0$, if $W_j \cap S = \emptyset$ for every $j \geq i$ then $(W, i, q') \to^*_{\mathcal{A}} (W, j, q'')$ for some $q' \in Q'$, $q'' \in Fin$ and $j \in [i..(i + n)]$ (i.e., starting with some state q' in Q' the automaton accepts the suffix $W_i W_{i+1} \dots$ of the word W after n moves at most as soon as no symbol from S occurs in the suffix). It is easy to see that $H_0 = \{Q' : Q' \subseteq Fin\}$ and $H_{n+1} = H_n \cup \{Q' :$ for every $T \subseteq (B \setminus S)$ there exists some $Q'' \in H_n$ such that $Q' \overset{T}{\leadsto} Q''\}$ for every $n \geq 0$.

Next let us construct the set G of all sets of control states $Q' \subseteq Sts$ such that for every infinite word $W \in (2^{Bnd})^\omega$, for every $i \geq 0$, if W meets the fairness constraint $\mathcal{P}_S$ then $(W, i, q') \to^*_{\mathcal{A}} (W, j, q'')$ for some $q' \in Q'$, $q'' \in Fin$ and $j \geq i$ (i.e. starting with some state q' in Q' the automaton eventually accepts the suffix $W_i W_{i+1} \dots$ of the word W as soon as the suffix meets the fairness constraint). This set G can be constructed as $\cap_{n \geq 0} G_n$ where $G_0 \supseteq G_1 \supseteq \dots G_n \supseteq G_{n+1} \supseteq \dots$ is non-increasing sequence of sets G_n $(n \geq 0)$ of all sets of control states $Q' \subseteq Sts$ such that for every infinite word $W \in (2^B)^\omega$, for every $i \geq 0$, if $W_j \cap S = \emptyset$ for every $j \geq (i+n)$ then $(W, i, q') \to^*_{\mathcal{A}} (W, k, q'')$ for some $q' \in Q'$, $q'' \in Fin$ and $k \geq i$ (i.e., starting with some state q' in Q' the automaton accepts the suffix $W_i W_{i+1} \dots$ of the word W as soon as no symbol from S occurs in the suffix after position $(i+n)$). It is easy to see that $G_0 = H$ and $G_{n+1} = \{Q' \in G_n :$ for every $T \subseteq B$ there exists some $Q'' \in G_n$ such that $Q' \overset{T}{\rightsquigarrow} Q''\}$ for every $n \geq 0$.

Finally we have: the automaton $\mathcal{A}$ totally meets the fairness constraint $\mathcal{P}_S$ iff there exists a non-empty subset of the initial states $Good \subseteq Ini$ such that $Good \in G$. It finishes a sketch of the direct decision procedure.

6.2. Proof of Lemma 5

Proof by induction of subformulae structure. Here we extensively exploit structured notation.

If θ is a propositional symbol or negation of a propositional symbol then proof is straightforward. For example, let θ be a propositional symbol p and let $(V^B, i$ be a point. Then we have:

$$\begin{gathered}
(V^B, i) \models \theta \Leftrightarrow p \in V^B_i \Leftrightarrow \\
\Leftrightarrow p \in (V^B_i \cup F_i \cup G_i) \text{ for every } F_i \subseteq F \text{ and } G_i \subseteq G \Leftrightarrow \\
\Leftrightarrow \text{the automaton} \\
(\{p,\ \mathbf{accept}\}, (B \cup F \cup G), \{p\}, \{\mathbf{accept}\}, \{(p, S) \to \mathbf{accept} : p \in S\}) \\
\text{accepts every word } (V^B_i \cup F_i \cup G_i),\ (V^B_{i+1} \cup F_{i+1} \cup G_{i+1}),\ \dots, \\
\text{where } F_i,\ F_{i+1},\ \dots \subseteq F \text{ and } G_i,\ G_{i+1},\ \dots \subseteq G \\
\text{but } \emptyset = G_j = G_{j+1} = \dots \text{ for some } j \geq i.
\end{gathered}$$

If θ is $p\mathcal{U}q$ then we have:

$$\begin{gathered}
(V^B, i) \models \theta \Leftrightarrow (V^B, k) \models q \text{ for some } k \geq i \text{ and } (V^B, j) \models p \text{ for all } j \in [i..k[\\
\Leftrightarrow \\
\Leftrightarrow q \in V^B_k \text{ for some } k \geq i \text{ and } p \in V^B_j \text{ for all } j \in [i..k[\Leftrightarrow \\
\Leftrightarrow \text{the automaton} \\
(\{q, p\mathcal{U}q,\ \mathbf{accept}\}, (B \cup F \cup G), \{\ p\mathcal{U}q\}, \{\mathbf{accept}\}, \\
\{(p\mathcal{U}q, S) \to q : S \subset B \cup F \cup G\} \cup \{(p\mathcal{U}q, S, \mathbf{next}) \to p\mathcal{U}q : p \in S\} \cup \\
\cup \{(q, S) \to \mathbf{accept} : q \in S\}) \\
\text{accepts every word}
\end{gathered}$$

$$(V_i^B \cup F_i \cup G_i),\ (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\ \dots\ (V_k^B \cup F_k \cup G_k),$$
$$(V_{k+1}^B \cup F_{k+1} \cup G_{k+1}),\ \dots,$$
$$\text{where } F_i,\ F_{i+1},\ \dots,\ F_k,\ F_{k+1},\ \dots \subseteq F \text{ and } G_i,\ G_{i+1},\ \dots,\ G_k,\ G_{k+1},$$
$$\dots \subseteq G,$$
$$\text{but } \emptyset = G_l = G_{l+1} = \dots \text{ for some } l \geq k \Leftrightarrow$$
$$\Leftrightarrow \text{ the automaton}$$
$$(\{q, p\mathcal{U}q,\ \mathbf{accept}\}, (B \cup F \cup G), \{\ p\mathcal{U}q\}, \{\mathbf{accept}\},$$
$$\{(p\mathcal{U}q, S) \to q : S \subset B \cup F \cup G\} \cup \{(p\mathcal{U}q, S, \mathbf{next}) \to p\mathcal{U}q : p \in S\} \cup$$
$$\cup \{(q, S) \to \mathbf{accept} : q \in S\})$$
$$\text{accepts every word}$$
$$(V_i^B \cup F_i \cup G_i),\ (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\ \dots$$
$$\text{where } F_i,\ F_{i+1},\ \dots \subseteq F \text{ and } G_i,\ G_{i+1},\ \dots \subseteq G,$$
$$\text{but } \emptyset = G_j = G_{j+1} = \dots \text{ for some } j \geq i.$$

If θ is $\circ\phi$ then we have:

$$(V^B, i) \models \theta \Leftrightarrow (V^B, i+1) \models \phi \Leftrightarrow$$
$$\Leftrightarrow \text{ the automaton}$$
$$(\{\mathbf{accept}\} \cup Sub(\phi), (B \cup F \cup G), \{\phi\}, \{\mathbf{accept}\}, P(\phi))$$
$$\text{accepts every word } (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\ (V_{i+2}^B \cup F_{i+2} \cup G_{i+2}),\ \dots,$$
$$\text{where } F_{i+1},\ F_{i+2},\ \dots \subseteq F \text{ and } G_{i+1},\ G_{i+2},\ \dots \subseteq G$$
$$\text{but } \emptyset = G_j = G_{j+1} = \dots \text{ for some } j \geq i+1 \Leftrightarrow$$
$$\Leftrightarrow \text{ the automaton}$$
$$(\{\mathbf{accept}\} \cup Sub(\theta), (B \cup F \cup G), \{\theta\}, \{\mathbf{accept}\}, (\mathbf{next}\ ;\ P(\phi)))$$
$$\text{accepts every word}$$
$$(V_i^B \cup F_i \cup G_i),\ (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\ (V_{i+2}^B \cup F_{i+2} \cup G_{i+2}),\ \dots,$$
$$\text{where } F_i,\ F_{i+1},\ F_{i+2},\ \dots \subseteq F \text{ and } G_i,\ G_{i+1},\ G_{i+2},\ \dots \subseteq G$$
$$\text{but } \emptyset = G_j = G_{j+1} = \dots \text{ for some } j \geq i+1 \Leftrightarrow$$
$$\Leftrightarrow \text{ the automaton}$$
$$(\{\mathbf{accept}\} \cup Sub(\theta), (B \cup F \cup G), \{\theta\}, \{\mathbf{accept}\}, P(\theta))$$
$$\text{accepts every word } (V_i^B \cup F_i \cup G_i),\ (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\ \dots,$$
$$\text{where } F_i,\ F_{i+1},\ \dots \subseteq F \text{ and } G_i,\ G_{i+1},\ \dots \subseteq G$$
$$\text{but } \emptyset = G_j = G_{j+1} = \dots \text{ for some } j \geq i.$$

If θ is $\Box_a\phi$ then we have:

$$(V^B, i) \models \theta \Leftrightarrow (V^B, j) \models \phi \text{ for every } j \geq i \Leftrightarrow$$
$$\Leftrightarrow \text{ for every } j \geq i \text{ the automaton}$$
$$(\{\mathbf{accept}\} \cup Sub(\phi), (B \cup F \cup G), \{\phi\}, \{\mathbf{accept}\}, P(\phi))$$
$$\text{accepts every word } (V_j^B \cup F_j \cup G_j),\ (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\ \dots,$$
$$\text{where } F_j,\ F_{i+1},\ \dots \subseteq F \text{ and } G_i,\ G_{i+1},\ \dots \subseteq G$$
$$\text{but } \emptyset = G_k = G_{k+1} = \dots \text{ for some } k \geq j \Leftrightarrow$$
$$\Leftrightarrow \text{ for every } j \geq i \text{ the automaton}$$
$$(\{\mathbf{accept}\} \cup Sub(\theta), (B \cup F \cup G), \{\theta\}, \{\mathbf{accept}\}, (\mathbf{while}\ g_a?\mathbf{do\ next}\ ;\ P(\phi)))$$
$$\text{accepts every word}$$

$(V_i^B \cup F_i \cup G_i)$, $(V_{i+1}^B \cup F_{i+1} \cup G_{i+1})$, $\dots$ $(V_j^B \cup F_j \cup G_j)$,
$(V_{j+1}^B \cup F_{j+1} \cup G_{j+1})$, $\dots$,
where F_i, F_{i+1}, $\dots$, F_j, F_{j+1}, $\dots \subseteq F$ and G_i, G_{i+1}, $\dots$, G_j, G_{j+1},
$\dots \subseteq G$, $g_a \in G_i$, $g_a \in G_{i+1}$, $\dots$, $g_a \in G_{j-1}$,
but $g_a \notin G_j$ and $\emptyset = G_k = G_{k+1} = \dots$ for some $k \geq j \Leftrightarrow$
$\Leftrightarrow$ the automaton
$(\{\textbf{accept}\} \cup \mathit{Sub}(\theta), (B \cup F \cup G), \{\theta\}, \{\textbf{accept}\}, P(\theta))$
accepts every word $(V_i^B \cup F_i \cup G_i)$, $(V_{i+1}^B \cup F_{i+1} \cup G_{i+1})$, $\dots$,
where F_i, F_{i+1}, $\dots \subseteq F$ and G_i, G_{i+1}, $\dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$.

If θ is $\Diamond\phi$ then we have:

$(V^B, i) \models \theta \Leftrightarrow (V^B, j) \models \phi$ for some $j \geq i \Leftrightarrow$
$\Leftrightarrow$ for some $j \geq i$ the automaton
$(\{\textbf{accept}\} \cup \mathit{Sub}(\phi), (B \cup F \cup G), \{\phi\}, \{\textbf{accept}\}, P(\phi))$
accepts every word $(V_j^B \cup F_j \cup G_j)$, $(V_{i+1}^B \cup F_{i+1} \cup G_{i+1})$, $\dots$,
where F_j, F_{i+1}, $\dots \subseteq F$ and G_i, G_{i+1}, $\dots \subseteq G$
but $\emptyset = G_k = G_{k+1} = \dots$ for some $k \geq j \Leftrightarrow$
$\Leftrightarrow$ for some $j \geq i$ the automaton
$(\{\textbf{accept}\} \cup \mathit{Sub}(\theta), (B \cup F \cup G), \{\theta\}, \{\textbf{accept}\}, (\textbf{next*}\ ;\ P(\phi)))$
accepts every word
$(V_i^B \cup F_i \cup G_i)$, $(V_{i+1}^B \cup F_{i+1} \cup G_{i+1})$, $\dots$ $(V_j^B \cup F_j \cup G_j)$,
$(V_{j+1}^B \cup F_{j+1} \cup G_{j+1})$, $\dots$,
where F_i, F_{i+1}, $\dots$, F_j, F_{j+1}, $\dots \subseteq F$ and G_i, G_{i+1}, $\dots$, G_j, G_{j+1},
$\dots \subseteq G$,
but $\emptyset = G_k = G_{k+1} = \dots$ for some $k \geq j \Leftrightarrow$
$\Leftrightarrow$ the automaton
$(\{\textbf{accept}\} \cup \mathit{Sub}(\theta), (B \cup F \cup G), \{\theta\}, \{\textbf{accept}\}, P(\theta))$
accepts every word $(V_i^B \cup F_i \cup G_i)$, $(V_{i+1}^B \cup F_{i+1} \cup G_{i+1})$, $\dots$,
where F_i, F_{i+1}, $\dots \subseteq F$ and G_i, G_{i+1}, $\dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$.

If θ is $\phi \wedge_c \psi$ then we have:

$(V^B, i) \models \theta \Leftrightarrow (V^B, i) \models \phi$ and $(V^B, i) \models \psi \Leftrightarrow$
$\Leftrightarrow$ the automaton
$(\{\textbf{accept}\} \cup \mathit{Sub}(\phi), (B \cup F \cup G), \{\phi\}, \{\textbf{accept}\}, P(\phi))$
accepts every word $(V_i^B \cup F_i \cup G_i)$, $(V_{i+1}^B \cup F_{i+1} \cup G_{i+1})$, $\dots$,
where F_i, F_{i+1}, $\dots \subseteq F$ and G_i, G_{i+1}, $\dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
and the automaton
$(\{\textbf{accept}\} \cup \mathit{Sub}(\psi), (B \cup F \cup G), \{\psi\}, \{\textbf{accept}\}, P(\psi))$
accepts every word $(V_i^B \cup F_i \cup G_i)$, $(V_{i+1}^B \cup F_{i+1} \cup G_{i+1})$, $\dots$,

where $F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
$\Leftrightarrow$ the automaton
$(\{\mathbf{accept}\} \cup Sub(\phi), (B \cup F \cup G), \{\phi\}, \{\mathbf{accept}\}, P(\phi))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$
where $f_c \in F_i,\, F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
and the automaton
$(\{\mathbf{accept}\} \cup Sub(\psi), (B \cup F \cup G), \{\psi\}, \{\mathbf{accept}\}, P(\psi))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$
where $f_c \notin F_i,\, F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
$\Leftrightarrow$ the automaton
$(\{\mathbf{accept}\} \cup Sub(\theta), (B \cup F \cup G), \{\theta\}, \{\mathbf{accept}\}, (\mathbf{if}\ f_c \mathbf{then}\ P(\phi) \mathbf{else}\ P(\psi)))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$
where $F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
$\Leftrightarrow$ the automaton
$(\{\mathbf{accept}\} \cup Sub(\theta), (B \cup F \cup G), \{\theta\}, \{\mathbf{accept}\}, P(\theta))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$
where $F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$.

If θ is $\phi \vee \psi$ then we have:

$(V^B, i) \models \theta \Leftrightarrow (V^B, i) \models \phi$ or $(V^B, i) \models \psi \Leftrightarrow$
$\Leftrightarrow$ the automaton
$(\{\mathbf{accept}\} \cup Sub(\phi), (B \cup F \cup G), \{\phi\}, \{\mathbf{accept}\}, P(\phi))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$
where $F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
or the automaton
$(\{\mathbf{accept}\} \cup Sub(\psi), (B \cup F \cup G), \{\psi\}, \{\mathbf{accept}\}, P(\psi))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$
where $F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
$\Leftrightarrow$ the automaton
$(\{\mathbf{accept}\} \cup Sub(\theta), (B \cup F \cup G), \{\theta\}, \{\mathbf{accept}\}, (P(\phi)\ \mathbf{U}\ P(psi)))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$
where $F_i,\, F_{i+1},\, \dots \subseteq F$ and $G_i,\, G_{i+1},\, \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$
$\Leftrightarrow$ the automaton
$(\{\mathbf{accept}\} \cup Sub(\theta), (B \cup F \cup G), \{\theta\}, \{\mathbf{accept}\}, P(\theta))$
accepts every word $(V_i^B \cup F_i \cup G_i),\, (V_{i+1}^B \cup F_{i+1} \cup G_{i+1}),\, \dots,$

where $F_i,\ F_{i+1},\ \dots \subseteq F$ and $G_i,\ G_{i+1},\ \dots \subseteq G$
but $\emptyset = G_j = G_{j+1} = \dots$ for some $j \geq i$.

Proof end.