Expressive Completeness of Temporal Logic of Action

Alexander Rabinovich

Department of Computer Science Raymond and Beverly Sackler Faculty of Exact Sciences Tel Aviv University*, Tel Aviv 69978, Israel,
e.mail: rabino@math.tau.ac.il

Abstract. The paper compares the expressive power of monadic second order logic of order, a fundamental formalism in mathematical logic and theory of computation, with that of a fragment of Temporal Logic of Actions introduced by Lamport for specifying the behavior of concurrent systems.

1 Introduction

The Temporal Logic of Actions (TLA) was introduced by Lamport [3] as a logic for specifying concurrent systems and reasoning about them. One of the main differences of TLA from other discrete time temporal logics is its inability to specify that one state should immediately be followed by the other state, though it can be specified that one state is followed by the other state at some later time.

Lamport [2] argued in favor of this decision *'The number of steps in a Pascal implementation is not a meaningful concept when one gives an abstract, high level specification'*. For example, programs like $Pr_1 :: x := True; y := False$ and $Pr_2 :: x := True; Skip; y := False$ are not distinguishable by the TLA specifications, however, they are distinguishable in linear time temporal logic, one of the most popular temporal logics [4].

As a consequence of the decision not to distinguish between '*doing nothing and taking a step that produces no changes*' [3], the language of TLA contains the next time operator in a very restricted form. For the same reasons the TLA existential quantifier $\exists^{TLA}$ has a semantics different from the standard existential quantifier.

In this paper we consider the fragment of Lamport's Temporal Logic of Action where variables can only receive boolean values (BTLA). We compare the expressive power of BTLA with that of monadic second order logic of order.

One of the consequences of TLA design decision is that only *stuttering* closed languages are definable in TLA. We will show that

(1) if a stuttering closed ω-language is definable in monadic second order logic of order then it is definable in BTLA.

* Supported by a research grant of Tel Aviv University.

Together with Theorem 6 from [6] this shows that an ω-language is definable in BTLA if and only if it is stuttering closed and definable in monadic second order logic.

In [6] we proved that there is no compositional translation from BTLA into monadic second order logic. The proof of (1) provides a translation from monadic logic into BTLA. However, this translation is also not compositional.

A continuous time interpretation for TLA was suggested in [6] and it was shown there that this interpretation is more appropriate than the standard discrete time interpretation. A compositional translation from BTLA into monadic logic under the continuous time interpretation was given in [6]. Here we will show that

(2) there exists a compositional translation from monadic second order logic into BTLA under the continuous time interpretation.

Hence, under the continuous time interpretation, BTLA and second order monadic logic can be translated one into the other in a compositional way.

The paper is organized as follows. In section 2 we fix terminology and notations. Section 3 recalls the syntax and the semantics of monadic second order logic of order. Section 4 recalls the connection between automata on ω-strings and monadic second order logic (see [9, 8] for a survey). We also provide here an automata theoretical characterization of the languages definable in the logic under a continuous time interpretation. The syntax and the semantics of BTLA is provided in section 5. Section 6 characterizes the expressive power of BTLA.

2 Terminology and Notations

Notations: $\mathbf{N}$ is the set of natural numbers; $\mathbf{R}$ is the set of real numbers, $\mathbf{R}^{\geq 0}$ is the set of non negative reals; **BOOL** is the set of booleans and Σ is a finite non-empty set.

A function from $\mathbf{N}$ to Σ is called an ω-string over Σ. A function h from the non-negative reals into a finite set Σ is called a *finitely variable signal* over Σ if there exists an unbounded increasing sequence $\tau_0 = 0 < \tau_1 < \tau_2 \ldots < \tau_n < \ldots$ such that h is constant on every interval (τ_i, τ_{i+1}). Below we will use 'signal' for 'finitely variable signal'. We say that a signal x is *right continuous* at t iff there is $t_1 > t$ such that $x(t) = x(t')$ for all t' which satisfies $t < t' < t_1$. We say that a signal is *right continuous* if it is right continuous at every t.

A set of ω-strings over Σ is called an ω-language over Σ. Similarly, a set of finitely variable (respectively, right continuous) signals over Σ is called a finitely variable (respectively, right continuous) Σ-signal language.

Let σ be an ω-string. We denote by $\sigma_{[n,\infty)}$ the ω-string $\langle s_n, s_{n+1}, \ldots \rangle$ and by $head(\sigma)$ its first letter s_0. For an ω-string σ and a letter s we denote by $s\sigma$ the ω-string $\langle s,\ s_0,\ s_1, \ldots \rangle$.

The collapse of an ω-string $\sigma = \langle s_0, s_1 \ldots s_n, \ldots \rangle$ is the ω-string $\natural\sigma$ which is defined recursively as follows:

$$\sharp\sigma = \begin{cases} \sigma & \text{if } \forall i.\ s_i = s_0 \\ s_0\sharp\sigma_{[i,\infty)} & \text{if } s_i \neq s_0 \text{ and } s_j = s_0 \text{ for all } j < i \end{cases}$$

Hence, operator $\sharp$ assigns to each ω-string σ the ω-string obtained by replacing every finite maximal subsequence $\langle s_i, s_{i+1} \ldots \rangle$ of identical letters in σ by a letter s_i. The ω-strings $\sigma = \langle s_0, s_1 \ldots s_n, \ldots \rangle$ and $\sigma' = \langle s'_0, s'_1 \ldots s'_n, \ldots \rangle$ are stuttering equivalent (notations $\sigma \simeq \sigma'$) if $\sharp\sigma = \sharp\sigma'$. Let L be an ω-language. We use the notation $Stutt(L)$ for the stuttering closure of L which is defined as $\{\sigma\ :\ \text{there exists } \sigma' \in L \text{ such that } \sigma \simeq \sigma'\}$. We say that an ω-language L is stuttering closed if $L = Stutt(L)$.

3 Monadic Second Order Theory of Order

3.1 Syntax

The language $L_2^<$ of monadic second order theory of order has individual variables, monadic second order variables, a binary predicate $<$, the usual propositional connectives and first and second order quantifiers $\exists^1$ and $\exists^2$. We use $t,\ v$ for individual variables and $x,\ y$ for second order variables.

The atomic formulas of $L_2^<$ are formulas of the form: $t < v$ and $x(t)$. The formulas are constructed from atomic formulas by logical connectives and first and second order quantifiers.

We write $\psi(x, y, t, v)$ to indicate that the free variables of a formula ψ are among $x,\ y,\ t,\ v$.

3.2 Semantics

A structure $K = \langle A,\ B, <_K\ \rangle$ for $L_2^<$ consists of a set A partially ordered by $<_K$ and a set B of monadic functions from A into $BOOL$. The satisfiability relation $K,\ \tau_1, \ldots \tau_m, \mathbf{x_1} \ldots \mathbf{x_n} \models \psi(t_1, \ldots t_m,\ x_1, \ldots x_m)$ is defined in a standard way.

We will be interested mainly in the following structures:

1. Structure $\omega = \langle \mathbf{N}, 2^{\mathbf{N}}, <_N\ \rangle$, where $2^{\mathbf{N}}$ is the set of all monadic functions from $\mathbf{N}$ into $BOOL$.
2. The signal structure Sig is defined as $Sig = \langle \mathbf{R}^{\geq 0},\ SIG, <_R\ \rangle$, where SIG is the set of finitely variable boolean signals.
3. The right continuous signal structure is defined as $\langle \mathbf{R}^{\geq 0},\ RSIG, <_R\ \rangle$, where $RSIG$ is the set of right continuous boolean signals.

3.3 Definability

Let $\phi(x)$ be an $L_2^<$ formula and $K = \langle A,\ B, <_K\ \rangle$ be a structure. We say that a set $C \subseteq B$ is definable by $\phi(x)$ if $\mathbf{x} \in C$ if and only if $K,\ \mathbf{x} \models \phi(x)$.

Example (Interpretations of Formulas).

1. The formula $\forall t_1 \forall t_2.\ t_1 < t_2 \wedge (\neg\exists t_3.\ t_1 < t_3 < t_2) \rightarrow (x(t_1) \leftrightarrow \neg x(t_2))$ defines the ω-language $\{(01)^\omega,\ (10)^\omega\}$ in the structure ω and defines the set of all signals in the signal and right continuous signal structures.
2. The formula $\exists y.\ \exists t'.\ y(t') \wedge \forall t.\ x(t) \rightarrow y(t) \wedge \forall t_1 \forall t_2 . t_1 < t_2 \wedge y(t_1) \wedge y(t_2) \rightarrow \exists t_3 . t_1 < t_3 < t_2 \wedge \neg y(t_3)$ defines in the structure ω the set of strings in which between any two occurrences of 1 there is an occurrence of 0. In the signal structure the above formula defines the set of signals that receive value 1 only at isolated points. The formula defines the empty language under right continuous signal interpretation.

In the above examples, all formulas have one free second order variable and they define languages over alphabet $\{0,\ 1\}$. A formula $\psi(x_1, \ldots x_n)$ with n free second order variables defines a language over alphabet $\{0,\ 1\}^n$.

4 Labeled Transition Systems

4.1 Syntax

A Labeled Transition System T is a triple $\langle Q,\ \Sigma,\ \rightarrow\rangle$ that consists of a set Q of states, a finite alphabet Σ of actions and a transition relation $\rightarrow$ which is a subset of $Q \times \Sigma \times Q$; we write $q \overset{a}{\rightarrow} q'$ if $\langle q,\ a,\ q'\rangle \in\rightarrow$; if Q is finite we say that the LTS is finite.

Sometimes the alphabet Σ of T will be the Cartesian product $\Sigma_1 \times \Sigma_2$ of other alphabets; in such a case we will write $q \overset{a,b}{\rightarrow} q'$ for the transition from q to q' labeled by the pair $(a,\ b)$.

An **automaton** $\mathcal{A}$ over Σ is a triple $\langle T,\ INIT(\mathcal{A}),\ FAIR(\mathcal{A})\rangle$, where $T = \langle Q,\ \Sigma,\ \rightarrow\rangle$ is an LTS over alphabet Σ; $INIT(\mathcal{A}) \subseteq Q$ - the initial states of $\mathcal{A}$. and $FAIR(\mathcal{A})$ - a collection of fairness conditions (subsets of Q).

4.2 Semantics

A run of an automaton $\mathcal{A}$ is an ω-sequence $q_0 a_0 q_1 a_1 \ldots$ such that $q_i \overset{a_i}{\rightarrow} q_{i+1}$ for all i. Such a run meets the initial conditions if $q_0 \in INIT(\mathcal{A})$. A run meets the fairness conditions if the set of states that occur in the run infinitely many times is a member of $FAIR(\mathcal{A})$.

An ω-string $a_0, a_1 \ldots$ over Σ is accepted by $\mathcal{A}$ if there is a run $q_0 a_0 q_1 a_1 \ldots$ that meets the initial and fairness conditions of $\mathcal{A}$. The ω-language *definable* (or accepted) by $\mathcal{A}$ is the set of all ω-strings acceptable by $\mathcal{A}$.

A right continuous signal x over Σ is accepted by $\mathcal{A}$ if there are an ω-string $a_0 a_1 \ldots a_n \ldots$ over alphabet Σ acceptable by $\mathcal{A}$ and an unbounded increasing sequence $0 = \tau_0 < \tau_1 < \ldots < \tau_i < \ldots$ of reals such that $x(\tau) = a_i$ for $\tau \in [\tau_i,\ \tau_{i+1})$.

A signal x over Σ is accepted by $\mathcal{A}$ if $\mathcal{A}$ is an automaton over the alphabet $\Sigma \times \Sigma$ and there are an ω-string $\langle a_0, b_0\rangle\langle a_1, b_1\rangle \ldots \langle a_n, b_n\rangle \ldots$ acceptable by $\mathcal{A}$

and an unbounded increasing sequence $0 = \tau_0 < \tau_1 < \ldots < \tau_i < \ldots$ of reals such that $x(\tau_i) = a_i$ and $x(\tau) = b_i$ for $\tau \in (\tau_i,\ \tau_{i+1})$.

It is clear that if the ω-languages acceptable by $\mathcal{A}$ and $\mathcal{B}$ are stuttering equivalent then $\mathcal{A}$ and $\mathcal{B}$ define the same right continuous signal language.

We say that an ω-language (respectively, finite variability signal language or right continuous language) is definable in monadic logic if the language is definable in monadic logic in the structure ω (respectively, the structure *Sig* or the structure of right continuous signals).

Theorem 1. *(Büchi [1]) An ω-language is definable by a finite state automaton iff it is definable by a monadic formula.*

Theorem 2. *[7] A finitely variable signal language is definable by a finite state automaton if and only if it is definable by a monadic formula.*

Theorem 3. *A right continuous signal language is definable by a finite state automaton if and only if it is definable by a monadic formula.*

Proof. The only if direction is obtained by a direct formalization of the behavior of a finite state automaton.

The if direction is obtained by the method of interpretation [5] as follows. Let $\phi(x_1, \ldots x_n)$ be a monadic formula. First, we construct a monadic formula $\phi^*(x_1, \ldots x_n)$ such that the language definable by ϕ under finitely variable interpretation coincides with the language definable by ϕ under right continuous signal interpretation.

Let $rsignal(x)$ be the formula $\forall t \exists t'.\ t' > t \land \forall t''. t < t'' < t' \rightarrow x(t) = x(t'')$. It is clear that a signal satisfies $rsignal(x)$ iff it is right continuous.

Let ϕ' be obtained from ϕ by relativizing all the second order quantifiers to the right continuous signals, i.e., by replacing "$\forall x.\ \ldots$" (respectively "$\exists x.\ \ldots$") by "$\forall x.\ rsignal(x) \rightarrow \ldots$" (respectively, "$\exists x.\ \land \ldots$"). It is easy to see that a right continuous signal satisfies ϕ under right continuous interpretation iff it satisfies ϕ' under finitely variable interpretation. Hence, the required formula $\phi^*(x_1, \ldots x_n)$ can be defined as $rsignal(x_1) \land rsignal(x_2) \land \ldots \land rsignal(x_n) \land \phi'$.

By Theorem 2, there exists an automaton over $\Sigma \times \Sigma$ such that the finitely variable signal language definable by $\mathcal{A}$ is the same as the language definable by ϕ^*. Let $\mathcal{B}$ be the automaton over Σ defined as follows: (1) remove from $\mathcal{A}$ all transitions of the form $q_1 \xrightarrow{a,b} q_2$ for $a \neq b$. (2) replace transitions of the form $q_1 \xrightarrow{a,a} q_2$ by $q_1 \xrightarrow{a} q_2$. It is not difficult to verify that the right continuous signal language definable by ϕ is the same as the right continuous signal language definable by $\mathcal{B}$. $\square$

5 Temporal Logic of Actions

We consider the fragment of Lamport's [3] Temporal Logic of Action where variables can only receive boolean values (BTLA).

5.1 Syntax

The symbol set of BTLA consists of:

1. A set Var of variables;
2. A set Var' of primed version for variables; $Var' = \{x' \ : \ x \in Var\}$.
3. Logical connectives $\wedge$ and $\neg$.
4. TLA existential quantifier $\exists^{TLA}$.
5. Modal operator $\Box$.
6. The special operator **Enabled**.

The syntax of BTLA formulas is summarized in Fig. 1.

$\langle$formula$\rangle$	$\triangleq$ $\langle$elementary formula$\rangle$ | $\neg$ $\langle$formula$\rangle$ | $\langle$formula$\rangle$ $\wedge$ $\langle$formula$\rangle$ | $\Box\langle$formula$\rangle$ | $\exists^{TLA}x.\langle$formula$\rangle$
$\langle$elementary formula$\rangle$	$\triangleq$ $\langle$simple state formula$\rangle$ | $\langle$enabled formula$\rangle$ | $\langle$action formula$\rangle$
$\langle$enabled formula$\rangle$	$\triangleq$ **Enabled**($\langle$action$\rangle$)
$\langle$action formula$\rangle$	$\triangleq \Box[\langle action\rangle]_{\langle simple\ state\ formula\rangle}$
$\langle$action$\rangle$	$\triangleq$ boolean combination of variables and primed variables.
$\langle$simple state formula$\rangle$	$\triangleq$ boolean combination of variables.

Fig. 1. Syntax of BTLA

Remark: (Primed variables) Priming a variable in TLA 'corresponds' to applying the next operator in temporal logic (see definition 1 (2) below). One can see that this next operator is used in BTLA in very restricted form.
Remark: (Free and bound occurrences of variables) A variable x occurs free in x and in x'. The only binding operator of BTLA is the existential quantifier. $\exists^{TLA}x.\psi$ binds all free occurrences of x in ψ.

5.2 Semantics of BTLA

A state is a function from a set Var of variables into the boolean set **BOOL**. A state sequence σ is an ω-sequence $\langle s_0, s_1, \ldots, s_n \ldots\rangle$ of states. State sequences $\sigma = \langle s_0, s_1 \ldots s_n, \ldots\rangle$ and $\sigma' = \langle s'_0, s'_1 \ldots s'_n, \ldots\rangle$ are equivalent up to a variable x (notation $\sigma =_x \sigma'$) if for every n, the states s_n and s'_n coincide on all variables distinct from x.

There is a natural correspondence between the states for variables $v_1, \ldots, v_k$ and the letters of the alphabet $\{0,\ 1\}^k$. This correspondence is lifted to the correspondence between state sequences (respectively, sets of state sequences) and ω-strings (respectively, ω-languages) over the alphabet $\{0,\ 1\}^k$. We will say that state sequences σ and σ' are stuttering equivalent up to x (notations $\sigma \simeq_x \sigma'$) if there exist σ_1, σ'_1 such that $\sigma =_x \sigma_1$, $\sigma' =_x \sigma'_1$ and σ_1 is stuttering equivalent to σ'_1.

We are going to recall the definition of the satisfaction relation between state sequences and a superset of BTLA formulas, which was called raw TLA by Lamport [3]. In the following definition x denotes a BTLA variables and A denotes an action.

Definition 1. *The satisfaction relation $\models$ is defined as follows:*

1. $\sigma \models x$ *if* $head(\sigma)(x)$ *is equal to* TRUE.
2. $\sigma \models x'$ *if* $\sigma_{[1,\infty)} \models x$.
3. $\sigma \models \psi_1 \wedge \psi_2$ *if* $\sigma \models \psi_1$ *and* $\sigma \models \psi_2$
4. $\sigma \models \neg\psi$ *if not* $\sigma \models \psi$.
5. $\sigma \models \mathbf{Enabled}(A)$ *if there exists* σ' *such that* $head(\sigma)\sigma' \models A$.
6. $\sigma \models \Box\psi$ *if* $\sigma_{[n,\infty)} \models \psi$ *for every* n.
7. $\sigma \models \exists^{TLA} x.\psi$ *if there is* σ' *such that* $\sigma \simeq_x \sigma'$ *and* $\sigma' \models \psi$.

For an action A and a simple state formula p, the BTLA action formula $\Box[A]_p$ is considered as an abbreviation of the raw TLA formula $\Box(A \vee (p \leftrightarrow p'))$, where p' the formula obtained from p by replacing every variable x by its primed version x'. We will also use $\Box[A]_{p,q}$ as a shortand for a BTLA formula which is equivalent to the raw BTLA formula $\Box(A \vee ((p \leftrightarrow p') \wedge (q \leftrightarrow q')))$. As ususal $\Diamond\phi$ is an abbreviation for $\neg\Box\neg\phi$.

Note that the set of sequences which satisfies a BTLA formula is closed under stuttering, i.e., $\sigma \models \psi$ and $\sigma \simeq \sigma'$ imply $\sigma' \models \psi$.

6 Expressive Completeness of BTLA

6.1 Discrete Time

Theorem 4. *An ω-language is definable in BTLA if and only if it is stuttering closed and definable in $L_2^{<}$.*

Proof. The only if direction is Theorem 5 of [6]. In order to show the if direction it is enough to show that for every ω-language accepted by a finite state automaton its stuttering closure is definable in BTLA.

For an automaton $\mathcal{A} = \langle T,\ INIT(\mathcal{A}),\ FAIR(\mathcal{A})\rangle$, where $T = \langle Q,\ \Sigma,\ \rightarrow\rangle$ is an LTS over alphabet Σ we first define the formulas $Init_{\mathcal{A}}, Next_{\mathcal{A}}, Fair_{\mathcal{A}}$ as follows

$$Init_{\mathcal{A}} \triangleq \bigvee_{q \in Init(\mathcal{A})} state = q$$

$$Next_{\mathcal{A}} \triangleq \Box[\bigvee_{q_1 \xrightarrow{a} q_2} (state = q_1 \wedge state' = q_2 \wedge x = a)]_{q,x}$$

$$Fair_{\mathcal{A}} \triangleq \bigvee_{Q \in FAIR(\mathcal{A})} Fair_Q, \text{ where}$$

for a set $Q = \{q_1, \ldots q_n\}$ the formula $Fair_Q$ is

$$Fair_Q \triangleq (\Diamond\Box \bigvee_{q \in Q} state = q) \wedge (\bigwedge_{q \in Q} \Box\Diamond state = q)$$

Let $\phi_{\mathcal{A}}$ be the formula $\exists^{TLA} state.\ (Init_{\mathcal{A}} \wedge Next_{\mathcal{A}} \wedge Fair_{\mathcal{A}})$.

In these definitions *state* and x range over ω-strings over the alphabets Q and Σ respectively. It is well-known that the strings over an alphabet of size $n < 2^k$ can be coded by k-tuples of strings over the binary alphabet. So, the above formulas are the shorthands of the BTLA formulas.

It is easy to check that an ω-string $a_0 a_1 \ldots a_n \ldots$ satisfies $\phi_{\mathcal{A}}$ if either it is stuttering equivalent to an ω-string acceptable by $\mathcal{A}$ or there are ω-strings $b_0 b_1 \ldots b_n \ldots$ and $q_0 q_1 \ldots q_n \ldots$ such that (1) $a_0 a_1 \ldots a_n \ldots$ and $b_0 b_1 \ldots b_n \ldots$ are stuttering equivalent and (2) there is n such that $\{q_{n+1}\} \in FAIR(\mathcal{A})$ and $\forall i > n.\ b_i = b_n \wedge q_i = q_{n+1}$ and $\forall i \leq n.\ q_i \xrightarrow{b_i} q_{i+1}$.

Therefore, if all fairness conditions of $\mathcal{A}$ contain at least two states then the language definable by $\phi_{\mathcal{A}}$ is the stuttering closure of the language accepted by $\mathcal{A}$.

It is not difficult to show that every automaton is equivalent to an automaton with all fairness conditions of cardinality at least two. Hence, every stuttering closed ω-language definable in $L_2^{<}$ is definable in BTLA. □

6.2 Continuous Time

We say that an ω-string $s = a_0 \ldots a_n \ldots$ represents a right continuous signal x if there is an unbounded ω-sequence $0 = \tau_0 < \tau_1 \ldots \tau_n \ldots$ such that $x(\tau) = a_i$ for $\tau \in [\tau_i,\ \tau_{i+1})$. It is clear that the set of ω-strings that represents a right continuous signal is stuttering closed.

A signal (a right continuous signal) language L is *speed-independent* if for every bijective increasing function $\rho:\ \mathbf{R}^{\geq 0} \rightarrow\ \mathbf{R}^{\geq 0}$ the following condition holds: $x \in L$ iff $x \circ \rho \in L$. It is clear that the set of right continuous signals representable by an ω-string is speed independent. Representability induces one-one correspondence between stuttering closed ω-languages and speed-independent right continuous signal languages. Through this correspondence we associate with every BTLA formula ϕ the set of right continuous signals that are representable by the ω-strings which satisfy ϕ. This set of right continuous signals is said to be definable by a BTLA formula ϕ under right continuous interpretation. It was shown in [6] that right continuous signal interpretation for BTLA is more appropriate than the standard discrete time interpretation.

Proposition 5. *For every monadic formula ϕ there exists a BTLA formula ϕ^* such that the language definable by ϕ under right continuous interpretation is the same as the right continuous signal language definable by ϕ^*.*

Proof. Let ϕ be a monadic formula and let L be a language definable by ϕ under right continuous interpretation. By Theorem 3, there exists an automaton $\mathcal{A}$ such that L is the right continuous signal language definable by $\mathcal{A}$. Let L' be the ω-language definable by $\mathcal{A}$. Let L'' be the stuttering closure L'. Note that the ω-languages L' and L'' represent the right continuous signal language L. From the proof of Theorem 4 it follows that L'' is definable by a BTLA formula ϕ^*. Therefore, ϕ and ϕ^* define the same language under right continuous signal interpretation. This completes the proof of the proposition. □

Proposition 5 implies the *if direction* of the following theorem; the *only-if direction* is Corollary 18 of [6].

Theorem 6. *A right continuous signal language is definable in BTLA if and only if it is definable in monadic second order logic of order.*

Observe that the proof of Theorem 6 is constructive. In particular, one can extract from the proof translation algorithms $Alg : L_2^< \to$ BTLA and $Alg' :$ BTLA $\to L_2^<$ such that the formulas ϕ and $Alg(\phi)$ (respectively, ϕ and $Alg'(\phi)$) define the same right continuous signal language.

Let us comment first on the translation algorithm from monadic logic into BTLA. In the proof of Proposition 5 we first translated monadic formulas into automata and then translated automata into a BTLA formulas. The size of the automaton obtained from ϕ might be much larger than the size of ϕ. In fact, for every k and every n there is a formula of the size $m > n$ such that the corresponding automaton has at least $exp_k(m)$ states, where $exp_k(n)$ is the k time iterated exponential function (e.g. $exp_2(n) = 2^{2^n}$). Obviously, the above translation algorithm from monadic logic into BTLA is not compositional.

On the other hand, the translation from BTLA into monadic logic that is extracted from the proof of the *only-if direction* of Theorem 6 is compositional (see Section 8 in [6]). Below an alternative proof of the *if direction* of Theorem 6 is provided in which we define a compositional translation Tr from monadic logic into BTLA. In the full version of the paper we show that the translation has the following property: the right continuous signal languages definable by a formula $\phi(x_1, \ldots x_n)$ without free first order variables and by BTLA formula $Tr(\phi)$ are the same.

Hence, under the right continuous signal interpretation, BTLA and second order monadic logic can be translated one into the other in a compositional way.

Let $Sing(t)$ be a BTLA formula that defines the ω-language $0^*11^*0^\omega$.

Let $ORDER(t_1, t_2)$ be the formula

$$\begin{aligned} &\Box(t_1 \leftrightarrow t_2) \\ &\qquad\qquad \vee\Diamond(t_1 \wedge \neg t_2 \wedge \Diamond(\neg t_1 \wedge \neg t_2 \wedge \Diamond t_2)) \\ &\qquad\qquad \vee\Diamond(t_2 \wedge \neg t_1 \wedge \Diamond(\neg t_1 \wedge \neg t_2 \wedge \Diamond t_1)) \end{aligned}$$

Let $Contains(x, t)$ be the formula

$$(\Diamond(x \wedge t)) \to ((\Box(t \to x)) \wedge \Box[(t \to x') \vee (t \leftrightarrow t' \wedge x \leftrightarrow x')]_x)$$

The translation is defined in Fig. 2. We use $free_1(\phi)$ (respectively, $free_2(\phi)$ for the set of first order (respectively, second order) variables which are free in ϕ.

$$
\begin{aligned}
Tr(t_1 < t_2) &\triangleq \Box(t_1 \rightarrow \Diamond t_2)\\
Tr(x(t)) &\triangleq \Box(t \rightarrow x)\\
Tr(\phi_1 \wedge \phi_2) &\triangleq Tr(\phi_1) \wedge Tr(\phi_2)\\
Tr(\neg\phi) &\triangleq \neg Tr(\phi)\\
Tr(\exists^1 t.\ \phi) &\triangleq \exists^{BTLA} t.\ Sing(t) \wedge\ Tr(\phi) \wedge\\
&\qquad \bigwedge_{t_i \in free_1(\phi)} Ordered(t, t_i) \wedge \bigwedge_{x \in free_2(\phi)} (Contains(x,t) \vee Contains(\neg x, t))\\
Tr(\exists^2 x.\ \phi) &\triangleq \exists^{BTLA} x.\ Tr(\phi) \wedge \bigwedge_{t \in free_1(\phi)} (Contains(x,t) \vee Contains(\neg x, t))
\end{aligned}
$$

Fig. 2. Translation from $L_2^<$ into BTLA

References

1. J. R. Büchi. On a decision method in restricted second order arithmetic In *Proc. International Congress on Logic, Methodology and Philosophy of Science*, E. Nagel at al. eds, Stanford University Press, pp 1-11, 1960.
2. L. Lamport. What good is temporal logic. In R. E. A. Manson editor *Information Processing 83, Proceedings of IFIP 9th World Congress*, Paris pp. 657-668. IFIP, North Holland.
3. L. Lamport. The Temporal Logic of Actions. ACM Transactions on Programming Languages and Systems, 16(3), pp. 872-923, 1994.
4. Z. Manna and A. Pnueli. The Temporal Logic of Reactive and Concurrent Systems, Springer Verlag, 1992.
5. M. O. Rabin. Decidable theories. In J. Barwise editor *Handbook of Mathematical Logic*, North-Holland, 1977.
6. A. Rabinovich. On Translations of Temporal Logic of Actions into Monadic Second Order Logic. Theoretical Computer Science 193 (1998), 197-214.
7. A. Rabinovich and B. A. Trakhtenbrot. From Finite Automata to Hybrid Systems. Fundamentals of Computation Theory 1997, LNCS vol. 1279, pages 411-422, 1997.
8. W. Thomas. Automata on Infinite Objects. In J. van Leeuwen editor *Handbook of Theoretical Computer Science*, The MIT Press, 1990.
9. B. A. Trakhtenbrot and Y. M. Barzdin. Finite Automata, North Holland Amsterdam, 1973.