On Translations of Temporal Logic of Actions into Monadic Second Order Logic

A. 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

1 Introduction

The Temporal Logic of Actions (TLA) was introduced by Lamport [4] 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 unability 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 [3] 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.

As a consequence of the decision not to distinguish between '*doing nothing and taking a step that produces no changes*' [4], the language of TLA contains the next time operator in very restricted from. For the same reasons the TLA existential quantifier $\exists^{TLA}$ has a semantics different from the standard existential quantifier.

One of our objectives is to investigate the expressive power of this quantifier. We will show that $\exists^{TLA}$ is not definable in monadic second order logic over a discrete time structure.

On the other hand, we will show that $\exists^{TLA}$ 'corresponds' to the standard second order existential quantifier of monadic second order logic in a continuous time structure.

Few comments on the status of monadic second order logic as a specification formalism are in order now.

Many formalisms for specifying discrete time temporal behavior were considered in the literature, e.g., ω-regular expressions, finite state automata, variety of linear time logics. One of the most expressive formalisms for which the equivalence of specifications is still decidable is the monadic second order theory of order (we denote its language by $L_2^<$) over the structure ω of natural numbers.

The special status of $L_2^<$ among all these discrete time formalisms rests on the fact that the specifications in the formalisms mentioned above admit a clear ('compositional') reformulation in $L_2^<$ [12]. Despite the fact that ω-regular expressions have the same expressive power as $L_2^<$, there exists no compositional translation from $L_2^<$ into an equivalent ω-regular expression.

The language $L_2^<$ of monadic second order logic of order contains individual variables, second order variables and the binary predicate $<$. In the structure ω (this structure will be defined precisely in section 5), the individual variables are interpreted as natural numbers, the second order variables as monadic functions from the natural numbers into the booleans and $<$ is the standard order on the set of natural numbers. We will also consider continuous time structures for $L_2^<$. In these structures the individual variables range over real numbers, the second order variables range over monadic functions from the reals into the booleans, and $<$ is the standard order relation on the set of real numbers.

In this paper we consider the fragment of Lamport's Temporal Logic of Action where variables can only receive boolean values (BTLA).

First we investigate the question of existence of a meaning preserving translation from BTLA into $L_2^<$ over the structure ω. Our results are:

(1) There exists no compositional translation from BTLA into $L_2^<$ over ω.

(2) There exists a translation from BTLA into $L_2^<$ over ω.

(1) will follow from

(3) The TLA existential quantifier is not definable[1] in $L_2^<$ over structure ω,

As a by-product of (2) we obtain

(4) The theory of BTLA in its standard discrete time model is decidable.

It is well known [9] that the monadic second order logic of order ($L_2^<$) is undecidable over the structure of real numbers.

We consider a substructure of real numbers which we call the signal structure. In the signal structure individual variables range over non-negative real numbers and the second order variables range over special boolean valued functions which we call signals.

We show that

(5) The $L_2^<$ theory of the signal structure is decidable.

As opposed to (1), we show that

(6) There exists a compositional translation of BTLA into $L_2^<$ over the signal structure.

Moreover, in contrast to (3), in our translation the TLA existential quantifier is translated into the standard second order existential quantifier over signals.

Recall that the monadic second order logic is accepted as a kind of an universal decidable logic for specifying discrete time temporal behavior. (1)-(3) above show that TLA is not compatible with the monadic logic over discrete time. Together with some unexpected laws of BTLA (see section 3.3) and a non-logical nature of $\exists^{TLA}$ (see section 5.1), these indicate that Lamport's decision to provide the discrete time interpretation of temporal logic of action is not the most appropriate. On the other hand, (6) above shows that the signal (a continuous time) interpretation of TLA is compatible with the signal interpretation of the monadic logic. Moreover, (5) demonstrates that the signal interpretation of BTLA is decidable.

The rest of the paper is organized as follows: Section 2 presents some notations and terminology. Syntax and semantics of BTLA are provided in section 3. In subsection

[1]the notion of definability will be discussed in section 5.1 and section 9.2.

3.3 we also point to some unexpected inference rules that are sound in BTLA. Syntax and semantics of monadic second order logic of order $L_2^<$ are given in section 4. In section 5 we investigate the interaction of $\exists^{TLA}$ with $L_2^<$ in the structure ω. We also provide a non-compositional meaning preserving translation from BTLA into $L_2^<$ over structure ω. Section 6 recalls results about two important continuous time structures for $L_2^<$ and introduces the signal structure. In section 7 we introduce the notion of speed independence which is useful for explaining a relationship between signal and discrete time specifications. Section 8 presents a compositional translation of BTLA into $L_2^<$ over the signal structure. Section 9 states some further results.

2 Terminology and Notations

A state is a function from a set *Var* of variables into the boolean set $BOOL=\{FALSE, TRUE\}$. We use symbol *St* for the set of all states and s for a state.

A state sequence σ is an ω-sequence $\langle s_0, s_1, \ldots \rangle$ of states. Let $\sigma = \langle s_0, s_1 \ldots s_i, s_{i+1}, \ldots \rangle$ be a state sequence. We denote by σ^n the state sequence $\langle s_n, s_{n+1}, \ldots \rangle$ and by $head(\sigma)$ the state s_0. For a state sequence σ and a state s we denote by $s\sigma$ the state sequence $\langle s,\ s_0,\ s_1, \ldots \rangle$.

The collapse of a state sequence σ is the state sequence $\natural\sigma$ which is defined recursively as follows:

$$\natural\sigma = \begin{cases} \sigma & \text{if } \forall i.\ s_i = s_0 \\ s_0 \natural \sigma^i & \text{if } s_i \neq s_0 \text{ and } s_j = s_0 \text{ for all } j < i \end{cases}$$

Hence, operator $\natural$ assigns to each state sequence σ the sequence obtained by replacing every finite maximal subsequence $\langle s_i, s_{i+1} \ldots \rangle$ of identical states in σ by a state s_i.

The 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; the state sequences σ and σ' are stuttering equivalent (notations $\sigma \simeq \sigma'$) if $\natural\sigma = \natural\sigma'$; they are stuttering equivalent up to x (notations $\sigma \simeq_x \sigma'$) if there exists σ'' such that $\sigma'' =_x \sigma$ and $\sigma'' \simeq \sigma'$.

It is easy to check that $=_x$, $\simeq$ and $\simeq_x$ are equivalence relations on state sequences.

Let L be a set of state sequence, 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 a set L of state sequences is stuttering closed if $L = Stutt(L)$.

Remark: The formal counterpart of 'a specification abstracts from a number of steps in an implementation' is 'the specification defines a stuttering closed set of state sequences'. For example, it is easy to check that the following transformations on state sequences preserve stuttering equivalence:

- Contraction: replace a subsequence $\langle s_i, s_i \rangle$ of identical states by one state s_i.
- Duplication: replace a state s_i by the subsequence $\langle s_i, s_i \rangle$.

We will use the usual logical abbreviations, e.g., $\psi \vee \psi' \triangleq \neg(\psi \wedge \psi')$ and $\psi \leftrightarrow \psi' \triangleq (\psi \wedge \psi') \vee (\neg\psi \wedge \neg\psi')$.

Usually the second order variables over a set X are interpreted as subsets of X. There is a one-one correspondence between the subsets of X and their characteristic functions. It will be a little bit more convenient for us to deal with characteristic functions instead of subsets. Hence we define a second order environment η over a set X as a function from *Var* into monadic functions from X into the booleans.

Let $\eta \in \{x_1, \dots, x_n\} \to (Nat \to BOOL)$ be a second order environment for variables $\{x_1, \dots x_n\}$ over the set Nat of natural numbers. With η associate the state sequence $\sigma' = \langle s_0, s_1 \dots s_k, \dots \rangle$ defined as $s_k(x_i) \triangleq \eta(x_i)(k)$. The above mapping sets up one-one correspondence between the set $\{x_1, \dots, x_n\} \to (Nat \to BOOL)$ of environment over natural numbers and the set $Nat \to (\{x_1, \dots, x_n\} \to BOOL)$ of state sequences.

Also there exists a one-one correspondence between the set of ω-strings over alphabet $\{0,1\}^n$ and the set $\{x_1, \dots, x_n\} \to (Nat \to BOOL)$ of second order environments for variables $\{x_1, \dots, x_n\}$ over the set of natural numbers. With an environment η for variables $x_1, \dots x_n$ we associate the ω-string $a_0 a_1 \dots a_k \dots$ over alphabet $\{0,1\}^n$ defined by $a_k \triangleq \langle b_1^k, \dots b_n^k \rangle$ where b_i^k is 1 if $\eta(x_i)(k)$ holds and b_i^k is 0 otherwise.

To summarize we will use natural one-one correspondences between following three sets:

1. The set $Nat \to \{x_1, \dots, x_n\} \to BOOL$ of state sequences.
2. The set $\{x_1, \dots, x_n\} \to Nat \to BOOL$ of second order environments for the variables $\{x_1, \dots, x_n\}$ over the set of natural numbers.

3. The set $Nat \to \{0,1\}^n$ of ω-strings over alphabet $\{0,1\}^n$

An ω-language is a set of ω-strings. We will say that an ω-language L over alphabet $\{0,1\}^n$ is definable by a formula $\psi(x_1,\ldots,x_n)$ if L consists of all ω-strings which 'satisfy' ψ in the sense defined below.

3 Temporal Logic of Actions

We consider the fragment of Lamport's [4] Temporal Logic of Action where variables can only receive boolean values (BTLA).

3.1 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.

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 occurences 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.

$$
\begin{array}{lcl}
\langle\text{formula}\rangle & \triangleq & \langle\text{elementary formula}\rangle \mid \neg\, \langle\text{formula}\rangle \mid \langle\text{formula}\rangle \wedge \langle\text{formula}\rangle \\
 & & \mid \Box\langle\text{formula}\rangle \mid \exists^{TLA} x.\langle\text{formula}\rangle \\
\langle\text{elementary formula}\rangle & \triangleq & \langle\text{simple state formula}\rangle \mid \langle\text{enabled formula}\rangle \mid \langle\text{action formula}\rangle \\
\langle\text{enabled formula}\rangle & \triangleq & \mathbf{Enabled}(\langle\text{action}\rangle) \\
\langle\text{action formula}\rangle & \triangleq & \Box[\langle action\rangle]_{\langle simple\ state\ formula\rangle} \\
\langle\text{action}\rangle & \triangleq & \text{boolean combination of variables and primed variables.} \\
\langle\text{simple state formula}\rangle & \triangleq & \text{boolean combination of variables.}
\end{array}
$$

Figure 1: Syntax of BTLA

3.2 Semantics of BTLA

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 [4]. 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* $\sigma^1 \models x$.
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* $\sigma^n \models \psi$ *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'.

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 $\sigma' \models \psi$.

Remarks: (1) It is clear that the union and the intersection of stuttering closed sets of state sequence is a stuttering closed set of state sequence. Also the complementation of a stuttering closed set of state sequences is stuttering closed. (2) Note that the standard existential quantifier does not preserve stuttering closedness. This is the reason that the definition of $\exists^{TLA}$ is different from the definition of the standard existential quantifier. In definition 1.7 of $\exists^{TLA}x\cdot$ stuttering equivalence up to x ($\simeq_x$) is used whereas in the definition of the standard existential quantifier $\exists x\cdot$ equivalence up to x ($=_x$) is used.

3.3 About TLA Existential Quantifier and Stuttering

Lamport argued (see [4]) that TLA existential quantifier "really is existential quantification because it obeys the ordinary laws of existential quantification. In particular, the usual rules ... are sound. From these rules, one can deduce the expected properties of existential quantification, such as $(\exists^{TLA}x.F \vee G) \leftrightarrow (\exists^{TLA}x.F) \vee (\exists^{TLA}x.G)$".

However, the design decision to combine in $\exists^{TLA}$ both the logical existential quantification and the non logical closure under stuttering has some unexpected consequences.

For example, let $\psi(x)$ be a BTLA formula with only one free variable x and assume that $\exists^{TLA}x.\ \psi(x)$ holds. Since every formula of BTLA defines a stuttering closed set, it is clear that at least one of the following cases holds:

1. $\sigma_1 \models \psi(x)$, where all the states of a suffix σ_1^n of σ_1 assign to x the value 0.

2. $\sigma_2 \models \psi(x)$, where all the states of a suffix σ_2^n of σ_2 assign to x the value 1.

3. $\sigma_3 \models \psi(x)$, where all even states of a suffix $\sigma_3^{2\times n}$ of σ_3 assign to x the value 0 and all odd states of $\sigma_3^{2\times n}$ assign to x the value 1.

One of the consequences of the above observation is that the following inference rule is also sound in BTLA:

A BTLA sound inference rule: From the conjunction of $\exists^{TLA}x.\ \psi_i(x)$, where $i = 1,\ldots,4$ deduce the disjunction of $(\exists^{TLA}x_ix_j.\ \psi_i(x_i) \wedge \psi_j(x_j) \wedge \Diamond\Box(x_i = x_j))$, where $1 \leq i < j \leq 4$ and $\Diamond\psi$ is an abreviation for $\neg\Box\neg\psi$.

Lamport extensively comments that in TLA variables have no types and can assume any value[2]. He writes "This approach may seem strange to computer scientists used to types

[2]He considers the full Temopral logic of Actions, in this paper we consider its boolean fragment.

in programming languages, but it captures the way mathematicians have reasoned for thousands years" [4].

However, it seems to us that the decision to consider typeless logic is implicitly forced by soundness of such unexpected inference rules in typed versions of TLA.

4 Monadic Second Order Theory of Order

In this section we recall the definitions of the syntax and the semantics of monadic second order theory of order.

4.1 Syntax

The language $L_2^<$ of monadic second order theory of order has a set Var_1 of individual variables, a set Var_2 of second order variables, a binary predicate $<$, the usual propositional connectives and first and second order quantifiers.

We will use t, u, v for individual variables and x, y for second order variables.

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

We will write $F(x, y, t, u)$ to indicate that the free variables of a formula F are among x, y, t, u.

4.2 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 $BOOL$.

An environment α for individual variables is a function from the set of individual variables into A and an environment η for the second order variables is a function from the set of second order variables into B. Below the satisfaction relation $\alpha,\ \eta \models \psi$ is defined by induction on the structure of $L_2^<$ formulas.

Definition 2 *(Semantics of $L_2^<$ formulas)*

1. $\alpha,\ \eta \models t < u$ *if* $\alpha(t) <_K \alpha(u)$.

2. $\alpha,\ \eta \models x(t)$ *if* $\eta(x)$ *maps* $\alpha(t)$ *to* TRUE.

3. $\alpha,\ \eta \models \psi_1 \wedge \psi_2$ *if* $\alpha,\ \eta \models \psi_1$ *and* $\alpha,\ \eta \models \psi_2$

4. $\alpha,\ \eta \models \neg\psi$ *if not* $\alpha,\ \eta \models \psi$.

5. $\alpha,\ \eta \models \exists^1 t.\psi$ *if there exists* α' *such that* $\alpha(u) = \alpha'(u)$ *for all* $u \neq t$ *and* $\alpha',\ \eta \models \psi$

6. $\alpha,\ \eta \models \exists^2 x.\psi$ *if there exists* η' *such that* $\eta(y) = \eta'(y)$ *for all* $y \neq x$ *and* $\alpha,\ \eta' \models \psi$

Notations: (A) In (5) the first order existential quantifier $\exists^1$ was defined and in (6) the second order existential quantifier $\exists^2$ was defined. Symbol $\exists$ will be used for both these quantifiers in the sequel; the ambiguity will be always resolved by context. If $\exists$ precedes an individual (second order) variable it will refer to the first (second) order existential quantifier. (B) Actually we had to use $\models_K$ for the satisfaction relation in a structure K and $\models_{BTLA}$ for the satisfaction relation of BTLA, however, in the sequel the ambiguity always will be resolved by a context.

5 Monadic Second Order Theory of ω

The structure ω consists of the set of all natural numbers, the standard order relation on the natural numbers and the set of all monadic functions from the naturals into the booleans.

In this section letters $k,\ m$ will denote natural numbers.

Let $\psi(x_1, \dots x_n)$ be a formula which does not contain free occurrences of individual variables. ψ specifies the set of all second order environments which satisfy it. Similarly, we associate a set of second order environments with $\psi(x_1, \dots x_n, k)$, where $\psi(x_1, \dots x_n, t)$ is a formula and k is a natural number.

Recall that the set $\{x_1, \dots x_n\} \to Nat \to BOOL$ of second order environments over ω is in one-one correspondence with the set of all ω-strings over $\{0, 1\}^n$. With a formula ψ as above we will associate the set of ω-strings which satisfy it.

The language $L_2^<$ is a very expressive formalism for specifying ω-languages. In the literature many other formalisms for specifying ω-languages were considered, e.g., ω-regular expressions, linear time temporal logic, etc.. The ω-languages which can be defined in the above mentioned formalisms are also definable in $L_2^<$. Moreover, there exists a compositional translation from the above mentioned formalisms into $L_2^<$. However, theorem 3 stated below will imply that there exists no compositional translation from BTLA into $L_2^<$.

5.1 The Extension of $L_2^<$ by the TLA Quntifier

Definition 3 *(The extension of $L_2^<$ by $\exists^{TLA}$) The extension of $L_2^<$ by TLA existential quantifier is defined by adding the following rules ito the syntax and the semantics of $L_2^<$.*

Syntax: *If ψ is a formula then $\exists^{TLA}x.\psi$ is a formula.*

Semantics: *$\alpha, \eta \models \exists^{TLA}x.\psi$ if there exists η' such that η and η' are stuttering equivalent up to x and $\alpha, \eta' \models \psi$.*

We use the notation $L_2^<[\exists^{TLA}]$ for this extension.

Remarks: (1) In section 9.2 we will comment on resticted versions of $L_2^<[\exists^{TLA}]$ in which $\exists^{TLA}$ is allowed to be applied only to the formulas without free individual variables. (2) Assume that ψ does not have free individual variables. Then (a) $\alpha, \eta \models \exists^{TLA}x.\psi$ if and only if $\alpha', \eta \models \exists^{TLA}x.\psi$ for every α' which is stuttering equivalent to α. (b) if x does not occur in ψ, then the set of ω-strings definable by $\exists^{TLA}x.\psi$ is the stuttering closure of the set of ω-strings definable by ψ.

Remark: (non-logical nature of $\exists^{TLA}$) The following examples demonstrate a non-logical nature of stuttering and of TLA existential quantifier. There exists a formula ϕ of $L_2^<$ which does not contain a second order variable y, however for some α and η the following holds: $\alpha,\ \eta \models \phi$ is not true, yet $\alpha,\ \eta \models \exists^{TLA}y.\ \phi$. Take for example, $x(t)$ for ϕ, $\alpha(t) = 2$ and $\eta(x) = \{0, 1\}$.

Let $TWICE$ be a binary predicate on the natural numbers such that $TWICE(k, m)$ holds iff $k = 2 \times m$.

Lemma 1 *$TWICE$ is definable by an $L_2^<[\exists^{TLA}]$ formula.*

Proof: Let $SUCC$ be a binary relation over natural numbers which is interpreted in the structure ω as $SUCC(k, m)$ iff $m = k + 1$.

It is well known [12] that the successor relation and the unary relation $ZERO$ which holds only on 0 are definable in $L_2^<$. Their defining formulas are

$Succ(t, t') \triangleq t < t' \wedge \neg\exists u.(t < u \wedge u < t')$.

$Zero(t) \triangleq \neg\exists u.u < t$

Let us define three auxiliary predicates by formulas $Alt(x, t)$, $B(x, y, t)$ and $Almost - Twice(t_1, t_2)$.

$$\begin{aligned} Alt(x,t) \triangleq\ & (\exists u_0.Zero(u_0) \wedge x(u_0)) \wedge (\forall u_1.\forall u_2.(u_1 < t \wedge (Succ(u_1, u_2)) \rightarrow (x(u_1) \leftrightarrow \neg x(u_2))) \\ & \wedge \forall u.u \geq t \rightarrow x(u) \end{aligned}$$

For even k, the language defined by $Alt(x, k)$ consisting of a single ω-string $(10)^{k/2}(1)^\omega$ over alphabet $\{0, 1\}$.

For odd k, the empty language is defined by $Alt(x, k)$.

$$\begin{aligned} B(x,y,t) \triangleq\ & Alt(x,t) \\ & \wedge \exists u_0.\exists u_1.(Zero(u_0) \wedge Succ(u_0, u_1) \wedge y(u_0) \wedge y(u_1)) \\ & \wedge \forall u_2.\forall u_3.\forall u_4.(Succ(u_2, u_3) \wedge Succ(u_3, u_4) \wedge u_4 \leq t \rightarrow (y(u_2) \leftrightarrow \neg y(u_4))) \\ & \wedge \forall u.u \geq t \rightarrow y(u) \end{aligned}$$

If k is a multiple of 4, then $B(x, y, k)$ the language defined by $B(x, y, k)$ consists of one ω-string

$$\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)^{k/4} \begin{bmatrix} 1 \\ 1 \end{bmatrix}^\infty \text{ over alphabet } \Sigma \triangleq \left\{\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right\}.$$

If k is not multiple of 4, then the empty language is defined by $B(x, y, k)$.

$Almost - Twice(t_1, t_2) \triangleq \exists y.Alt(y, t_1) \wedge \exists^{TLA} x.B(x, y, t_2)$

$Almost - Twice(k, m)$ holds if k is even and $m = 2 \times k$.

Now the predicate $TWICE$ can be defined by the following formula $Twice$:

$$\begin{aligned} Twice(t_1, t_2) \triangleq\ & Almost - Twice(t_1, t_2) \quad \vee \exists u_1.u_2.u_3.\ (Succ(t_1, u_1) \wedge Succ(t_2, u_2) \wedge Succ(u_2, u_3)\ . \\ & \wedge Almost - Twice(u_1, u_3)) \end{aligned}$$

$\square$

Let $L_2^<[TWICE]$ be the extension of $L_2^<$ by the predicate $TWICE$

Theorem 2 *(Trakhtenbrot [11])*

1. *$TWICE$ is not definable in $L_2^<$, i.e., there is no $L_2^<$ formula $\psi(t, t')$ which is equivalent to $TWICE(t, t')$ in the structure ω.*

2. *There exists an ω-language definable by an $L_2^<(TWICE)$ formula but not definable by any $L_2^<$ formula.*

3. *The set of $L_2^<[TWICE]$ sentences true in ω is undecidable*[3].

Lemma 1 and theorem 2 imply two following theorems

Theorem 3 *There exists an ω-language definable by an $L_2^<[\exists^{TLA}]$ formula but not definable by any $L_2^<$ formula.*

Remark: Note that $\exists^{TLA}$ is a third order operator. There is no standard notion of definability for third order operators. However, any reasonable notion would imply that by adding a definable operator the expressive power of a logical language will not increase. In this sense the above theorem can be interpreted as: $\exists^{TLA}$ is not definable in $L_2^<$ over ω.

Theorem 4 *The set of $L_2^<[\exists^{TLA}]$ sentences true in ω is undecidable.*

It is instructive to compare theorem 4 with

Theorem 5 *(Büchi [1]) The set of $L_2^<$ sentences true in ω is decidable.*

5.2 A Non-Compositional Translation of BTLA into $L_2^<$

In contrast to theorem 3, we will show

Theorem 6 *If a set of state sequences is definable in $BTLA$ then this set is $L_2^<$ definable in structure ω. Moreover, there exists an algorithm which translates every $BTLA$ formula into $L_2^<$ formula which defines the same set of state sequences.*

Remark: In view of the remark following theorem 3 the above translation from BTLA into $L_2^<$ cannot be compositional.

Note also that theorem 5 and theorem 6 imply

[3] In [8] and [12], it is attributed to Tarski that the monadic second-order theory of the structure $\langle \omega, <, + \rangle$ is undecidable (i.e., in the language $L_2^<$ extended by the addition predicate). In [12], the above theorem is stated for the addition predicate. Robinson [8] has shown that the addition predicate is $L_2^<$-definable from TWICE in structure ω.

Theorem 7 *BTLA is decidable.*

In the rest of this section the sketch for the proof of theorem 6 is given. All its arguments are valid for raw BTLA.

Recall [1], that that a set L of ω-strings is $L_2^{<}$ definable iff L is a regular ω-language (see [10] for a survey of automata on infinite objects). Moreover, there exist algorithms for translations between ω-regular expressions and $L_2^{<}$ formulas (see [12]).

We will prove that only regular ω-languages can be defined in BTLA and that there exists an algorithm for translating BTLA formulas into equivalent ω-regular expressions.

The proof is by induction on the structure of BTLA formulas.

It is easy to see that every elementary BTLA formula defines a regular ω-language and it is easy to construct for every elementary formula an equivalent ω-regular expression.

It is also well known that regular ω-languages are closed under complementation, conjunction, projection and $\Box$ operations. Moreover, there exists an algorithm for these operations on ω-regular expressions.

Hence, in order to complete the proof we have to show: for a ω-regular expressions r which defines the same language as a BTLA formula $\psi(x_1, \dots x_n)$, one can construct an ω-regular expression for the ω-languages defined by the formulas $\exists^{TLA} x_1.\psi$.

The proof of this fact follows from lemmas 8, 9 given below.

Recall that we use the notation $Stutt(L)$ for the stuttering closure of a language L.

One can easily show the following

Lemma 8 *If L is definable by a formula $\exists y.\psi(y, x_1 \dots x_n)$ then $\exists^{TLA} y.\psi(y, x_1 \dots x_n)$ defines language $Stutt(L)$.*

Lemma 9 *For a regular ω-language L the ω-language $Stutt(L)$ is regular. Moreover, there exists an algorithm which constructs an ω-regular expression for $Stutt(L)$ from an ω-regular expression for L.*

Proof: Let h be a language morphism defined as $h(a) \triangleq \{a^n : n > 0\}$. It is easy to show that $Stutt(L) = h(h^{-1}(L))$. It is easy to show that $Stutt(L) = h(h^{-1}(L))$. Hence, the first part of the lemma follows from the following easy generalization of a well known fact about regular languages over finite strings (see e.g. [2])

Fact: Regular ω-languages are closed under regular morphisms.

Actually, the proof of this fact gives an algorithm for constructing an ω-regular expression for the image (pre-image) of an ω-language L from an ω-regular expression that defines L and regular expressions that define a morphism. From this the second part of lemma 9 follows.

$\square$

6 Three Continuous Time Structures for $L_2^<$

Let $\mathbf{R}$ be the set of real numbers and let $<_R$ be the standard order on $\mathbf{R}$.

We use the letters τ, τ' to denote real numbers.

Rabin considered the structure $F = \langle \mathbf{R}, F_\sigma, <_R \rangle$, where F_σ is the set of monadic functions from $\mathbf{R}$ into $BOOL$ such that $x \in F_\sigma$ iff x is the characteristic function of a countable union of closed sets.

Rabin has shown

Theorem 10 *(Rabin [5]) The set of $L_2^<$ sentences true in F is decidable.*

Shelah considered the structure $M = \langle \mathbf{R}, 2^{\mathbf{R}}, <_R \rangle$, where $2^{\mathbf{R}}$ is the set of all monadic functions from $\mathbf{R}$ into $BOOL$. He has shown

Theorem 11 *(Shelah [9]) The set of $L_2^<$ sentences true in M is undecidable.*

Now we define signals and a signal structure on the reals. In section 8 a compositional translation of BTLA into $L_2^<$ over signals is provided.

Definition 4 *A function h from the non-negative reals into the set $BOOL$ (a finite set Σ) is called a boolean signal (respectively, Σ-signal) 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 $[\tau_i, \tau_{i+1})$.*

Let $SIGNAL$ be the set of all boolean signals. The signal structure Sig is defined as $Sig = \langle \mathbf{R}^+, SIGNAL, <_R \rangle$, where $\mathbf{R}^+$ is the set of non negative reals. A signal language is a set of signals.

Theorem 12 *The set of $L_2^<$ sentences true in the signal structure Sig is decidable.*

Proof: First let us note that for the restriction of the structure F to non-negative reals, theorem 10 still holds. (Below we will overload notations and notions from the structure F to its restriction on non-negative reals.)

It is clear that if x is a signal then it is the characteristic function of a countable union of closed sets. Hence every signal belongs to F_σ. It is also clear that $x \in F_\sigma$ is a signal if and only if it satisfies the formula $signal(x)$ defined as:

$$\begin{aligned} signal(x) \triangleq\ & \forall t.\exists t_1 > t.\forall t_2.t \leq t_2 \leq t_1 \rightarrow (x(t) \leftrightarrow x(t_2)) \wedge \\ & \forall t > 0.\exists t_1 < t.\forall t_2.t_1 \leq t_2 < t \rightarrow (x(t_1) \leftrightarrow x(t_2)). \end{aligned}$$

Below we provide an interpretation of the signal structure Sig inside structure F. (See [6] for the detailed description of the methods of interpretation.)

If A is a monadic second order formula then the formula A^{Sig} obtained from A by relativizing all second order quantifiers of A to signals is defined inductively on the structure of A by the following rules: (1) If A is without second order quantifiers then $A^{Sig} = A$. (2) If $A = B \wedge C$ or $A = \neg B$ or $A = \exists^1 t.B$ then $A^{Sig} = B^{Sig} \wedge C^{Sig}$ or $A^{Sig} = \neg B^{Sig}$ or $A^{Sig} = \exists^1 t.B^{Sig}$, respectively. (3) If $A = \exists^2 x.B$ or $A = \forall^2 x.B$ then $A^{Sig} = (\exists^2 x.signal(x) \wedge B^{Sig})$ or $A^{Sig} = (\exists^2 x.signal(x) \rightarrow B^{Sig})$, respectively.

This relativization allows to reduce the satisfiability of the formula A in structure Sig to the satisfiability of the formula A^{Sig} in the structure F. In particular, if A is a closed formula, then $\models_{Sig} \ A$ if and only if $\models_F \ A^{Sig}$. Therefore, theorem 12 follows from Theorem 10.

$\square$

7 Speed Independence

Lemma 13 *Let f be an increasing bijective function between non negative reals. Then h is a signal iff $h \circ f \triangleq \lambda v.h(f(v))$ is a signal.*

Definition 5 *Let L be a signal language. We say that L is speed independent if for every bijective increasing function f the following condition holds: $h \in L$ iff $h \circ f \in L$.*

Recall that in section 2 we agree to use a natural one-one correspondence between the following three sets:

1. The set $Nat \to \{x_1, \dots, x_n\} \to BOOL$ of state sequences.
2. The set $\{x_1, \dots, x_n\} \to Nat \to BOOL$ of second order environments for the variables $\{x_1, \dots, x_n\}$ in the structure ω.
3. The set $Nat \to \{0,1\}^n$ of ω-strings over alphabet $\{0,1\}^n$

In a similar way there exists a natural one-one correspondence between

4. The set $\{x_1, \dots, x_n\} \to SIGNAL$ of second order environments for the variables $\{x_1, \dots, x_n\}$ in the structure Sig.

   and

5. The set of signals over alphabet $\{0,1\}^n$

Below we first define a function $Disc$ (discretization) which assigns to every signal a set of ω-strings (ω-language). Then we define a function $Cont$ which assigns to every ω-string a set of signals. These functions are lifted to the function between the set of ω-languages and the set of signal languages. It turns out that, under this correspondence, the image of any signal language is a stuttering closed ω-language. We show (lemma 15) that (1) function $Cont$ is a bijection between the set of stuttering closed ω-languages and speed independent signal languages and (2) $Disc$ is its inverse.

We also show that any $L_2^<$ formulas without free individual variables defines a speed independent signal language.

Definition 6 *An unbounded increasing sequence $\tau_0 < \tau_1 < \dots < \tau_i < \dots$ is a τ-sample sequence for a signal h if*

1. $\tau_0 = \tau$
2. *If h is not continuous at τ' and $\tau < \tau'$ then there exists i such that $\tau_i = \tau'$.*

Definition 7 *An ω-string $a_0, a_1 \dots$ is a discretization of a signal h beginning from τ if there exists a τ-sample sequence $\tau_0, \tau_1, \dots \tau_n \dots$ for h such that $a_i = h(\tau_i)$.*

Definition 8 *A signal h is a continuezation of an ω-string w beginning at τ if w is a discretization of h beginning at τ.*

Notations: We use notation $Disc(h, \tau)$ for the set of all discretizations of signal h beginning at τ; we use $Cont(w, \tau)$ for the set of all continuezations of ω-string w, beginning at τ. We extend $Disc$ to a function from the signal languages in a natural way: $Disc(H, \tau) \triangleq \cup\{Disc(h, \tau) \ : \ h \in H\}$; $Cont$ is extended to a function from ω-languages in a similar way.

The following lemmas are straightforward

Lemma 14 *(Stuttering)*

1. *If $w, \ w' \in Disc(h, \tau)$ then w and w' are stuttering equivalent.*
2. *If $w \in Disc(h, \tau)$ and w' is stuttering equivalent to w then $w' \in Disc(h, \tau)$.*

Lemma 15 *(Speed independence vs stuttering closedness) If L is an ω-language then $Cont(L, 0)$ is a speed independent signal language. If L is a signal language then $Disc(L, 0)$ is a stuttering closed ω-language. Moreover, the mapping $\lambda L.Disc(L, 0)$ is a bijection between the set of speed independent signal languages and the set of stuttering closed ω-languages. $\lambda L.Cont(L, 0)$ is the inverse of $\lambda L.Disc(L, 0)$.*

Lemma 16 *(Speed independence of any $L_2^<$ specification ψ) Let f be an increasing injective function between non negative reals, α be an individual environment and η be a second order environment.*
If

1. *$\alpha'(t) = f(\alpha(t))$ for every individual variable t free in ψ, and*
2. *$\eta'(x) = \eta(x) \circ f$ for every second order variable x free in ψ,*

then $\alpha, \eta \models \psi$ iff $\alpha', \eta' \models \psi$. In particular any $L_2^<$ formula without free individual variables defines a speed independent set of signals.

8 A Compositional Translation of BTLA into $L_2^<$

Let $\psi(x_1, \ldots x_n)$ be a BTLA formula with free (propositional) variables in the set $\{x_1 \ldots x_n\}$. Our translation which is provided below, will map ψ to an $L_2^<$ formula $\psi'(x_1, \ldots x_n, t)$ with free second order variables $x_1, \ldots x_n$ and free individual variable t. Theorem 17 stated below, justifies this translation.

The notations $\psi\{u'/u\}$ will be used for the substitution of u' for all free occurrences of u in ψ.

We will also use the following abbreviations:

$NEXT(t_1, t_2, y) \triangleq t_1 < t_2 \wedge y(t_1) \neq y(t_2) \wedge \forall t'.(t_1 \leq t' < t_2 \rightarrow (y(t') = y(t_1))$

$$\begin{aligned} NEXT(t_1, t_2, x_1, \ldots x_n) \triangleq\ & t_1 < t_2 \wedge (x_1(t_1) \neq x_1(t_2) \vee \ldots \vee x_n(t_1) \neq x_n(t_2)) \wedge \\ & \forall t'.(t_1 \leq t' < t_2 \rightarrow (x_1(t') = x_1(t-1) \wedge \ldots \wedge x_n(t') = x_n(t_1))) \end{aligned} .$$

Compositional Translation:

Let us fix two individual variables t and t'. Our translation Tr is parameterized by these two variables.

Actions: The translation of a boolean combination A of variables $x_1 \ldots x_n$ and their primed versions $x'_1 \ldots x'_n$ is the formula $Tr(A)(x_1, \ldots, x_n, t, t') \triangleq A' \wedge NEXT(t, t', x_1, \ldots x_n)$, where A' is obtained from A by simultaneous substitution of $x_i(t)$ for x_i and $x_i(t')$ for x'_i.

Simple State Formulas: The translation of a boolean combination p of variables $x_1 \ldots x_n$ is the formula obtained from p by simultaneous substitution of $x_i(t)$ for x_i.

Enabled Formulas: $Tr(\mathbf{Enabled}(A)) \triangleq \exists t'.Tr(A)$

Action formulas: Let $x_1, \ldots x_n$ be the free variables of an action A and of a simple state formula p. Then $\Box[A]_p$ is translated as

$$\begin{aligned} & \exists y.(\forall u.(y(u) \leftrightarrow Tr(p)\{u/t\}) \wedge \\ & \wedge \forall t_1 t_2.(t_1 \geq t \wedge NEXT(t_1, t_2, x_1, \ldots x_n, , y)) \rightarrow (Tr(A)\{t_1/t, t_2/t'\} \vee y(t_1) = y(t_2))) \end{aligned}$$

Propositional connectives:

$Tr(\neg\psi) \triangleq \neg Tr(\psi)$

$Tr(\psi_1 \wedge \psi_2) \triangleq Tr(\psi_1) \wedge Tr(\psi_2)$

Quantifier: $Tr(\exists^{TLA} x.\psi) \triangleq \exists x.Tr(\psi)$.

Modality: $Tr(\Box\psi) \triangleq \forall u.u > t \Rightarrow Tr(\psi)\{u/t\}$.

By induction on the structure of BTLA formulas it is easy to show the following theorem which explains the relationship between BTLA formulas and their translations.

Theorem 17 *Let $\psi(x_1, \ldots, x_n)$ be a BTLA formula and let $\psi'(x_1, \ldots, x_n, t)$ be its translation (i.e., $\psi' = Tr(\psi)$). Let α be an individual environment which maps t to τ. Let S be $\{\sigma \ : \ \sigma \models_{BTLA} \psi\}$ and let $H(\tau)$ be $\{\eta \ : \ \alpha, \eta \models_{Sig} \psi'\}$. Then for every τ*

1. $S = Disc(H(\tau), \tau)$.
2. $H(\tau) = Cont(S, \tau)$.

Recall that lemma 15 sets up the correspondence between stuttering closed set of state sequences and speed independent set of signals. By this correspondence one can associate with every BTLA formula a speed independent set of signals. The above theorem implies

Corollary 18 *Let ψ and ψ' be as in theorem 17. Then the set of signals defined by a BTLA formula $\psi(x_1 \ldots x_n)$ is the same as the set of signals defined by the $L_2^<$ formula $\psi'(x_1, \ldots x_n, 0)$.*

Corollary 19 *(Preservation of equivalence and refinement relations by the translation)*

1. $\psi_1 \leftrightarrow \psi_2$ *iff* $Tr(\psi_1) \leftrightarrow Tr(\psi_2)$, *(i.e., ψ_1 and ψ_2 are equivalent BTLA formulas iff their translations are equivalent monadic formulas in the signal structure)*
2. $\psi_1 \rightarrow \psi_2$ *iff* $Tr(\psi_1) \rightarrow Tr(\psi_2)$

9 Further Results

9.1 Expressive Completeness of BTLA

Recall that every BTLA formula defines a stuttering closed ω-language and every $L_2^<$ formula defines a speed independent signal language in the signal structure. Through the bijection (see section 7 lemma 15) between stuttering closed ω-languages and speed independent signal languages, one can associate with every BTLA formula a speed independent signal language. It was shown in [7] that BTLA is complete in the following sense.

Expressive Completeness of BTLA for signal structure: For every $L_2^<$ formula $\psi(x_1, \ldots, x_n)$ there exists a BTLA formula $\psi'(x_1, \ldots, x_n)$ which defines the same signal language.

9.2 Un-definability of $\exists^{TLA}$ by an $L_2^<$ context

The set of $L_2^<$ contexts is defined in a standard way by adding the hole [ ] to the atomic formulas of $L_2^<$. For a context $C[\ ]$ and a formula ψ, the formula $C[\psi]$ is defined by replacing all occurrences of the hole by a formula ψ.

It is instructive to compare theorem 3 with the following result from [7]

Theorem 20 *There exists no $L_2^<$ context $C[\ ]$ such that for any formula $\psi(x_1, \dots x_n)$ which defines a stuttering closed language, the formulas $C[\psi]$ and $\exists^{TLA} x_1.\psi$ define the same language over the structure ω.*

Remarks: Both theorems say that $\exists^{TLA}$ is undefinable in $L_2^<$. Actually, the proof of theorem 3 gives an $L_2^<[\exists^{TLA}]$ formula which is not equivalent to any $L_2^<$ formula. However, in $L_2^<[\exists^{TLA}]$, the TLA existential quantifier can be applied to any formula (in particular to the formulas which contain free individual variables). Let us consider L_{rest} the sub-language of $L_2^<[\exists^{TLA}]$ in which we allow to apply $\exists^{TLA}$ only to the formulas without free individual variables. It can be shown that every L_{rest} formula is equivalent to an $L_2^<$ formula (the proof is similar to the proof of theorem 6). Let L_{rest}^{stutt} be the sub-language of L_{rest} in which $\exists^{TLA}$ can be applied only to the formulas which define stuttering closed languages. Theorem 20 states that $\exists^{TLA}$ of L_{rest}^{stutt} is not definable by any $L_2^<$ context. In particular this theorem implies that there exists no compositional translation from L_{rest}^{stutt} into $L_2^<$ in the structure ω.

9.3 Extension of $L_2^<$ with Stuttering

By adding the following rules one can extend $L_2^<$ by stutterings predicate:

Syntax: If ψ is a formula without free individual variables then $stut(\psi)$ is a formula.

Semantics: $\alpha, \eta \models stut(\psi)$ if there exists η' such that η and η' are stuttering equivalent and $\alpha, \eta' \models \psi$

Note that if ψ defines a regular ω-language then, by lemma 9, the formula $stut(\psi)$ also defines a regular ω-language. Hence, for any $L_2^<$ formula ψ the formula $stut(\psi)$ is equivalent to a $L_2^<$ formula.

However, note that there is no $L_2^<$ formula $STUT(x_1,\ x_2,\ y_1,\ y_2)$ such that $\alpha, \eta \models STUT((x_1,\ x_2,\ y_1,\ y_2)$ if and only if the restriction of η to $\{x_1,\ x_2\}$ is stuttering equivalent to the restriction of η to $\{y_1,\ y_2\}$. (This follows from observation that the ω-language over $\{0,\ 1\}^4$ defined by $STUT$ is not ω-regular.)

Note also that by extending $L_2^{<}$ by this third order predicate $STUT$ we can express the predicate $TWICE$. (The proof of this is similar to the proof of lemma 1 and is omitted here.)

Acknowledgments

The author is grateful to Yoram Hirshfeld, Albert R. Meyer and Boris A. Trakhtenbrot for helpful discussions and comments. Many thanks to the anonymous referees for the suggestions on the exposition of the material.

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, Standford University Press, pp 1-11, 1960.

[2] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Lamguages and Computations*, Addison-Wesley, Reading, MA, 1979.

[3] 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.

[4] L. Lamport. The Temporal Logic of Actions. ACM Transactions on Programming Languages and Systems, 16(3), pp. 872-923, 1994.

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

[6] M. O. Rabin. Decidable theories. In J. Barwise editor *Handbook of Mathematical Logic*, North-Holland, 1977.

[7] A. Rabinovich. On Expressive Completeness of Temporal Logic of Action. (In preparation.)

[8] R. M. Robinson, Restricted set-theoretical definitions in arithmetic. In *Proc. Amer. Math. Soc.*,**9**,pp 238-242, 1958.

[9] S. Shelah. The monadic theory of order. *Ann. of Math.*, **102**, pp 349-419, 1975.

[10] W. Thomas. Automata on Infinite Objects. In J. van Leeuwen editor *Handbook of Theoretical Computer Science*, The MIT Press, 1990.

[11] B. A. Trakhtenbrot. Some constructionsin the monadic predicate calculus, DAN SSSR, 1961, 140, No 2, pp 320-321.

[12] B. A. Trakhtenbrot and Y. M. Barzdin. Finite Automata, North Holland Amsterdam, 1973.