Games in TLA

Can BAŞKENT

May 15, 2008
cbaskent@gc.cuny.edu, www.canbaskent.net

1 Prologue

The largest professional organization of computer scientists is Association for Computing Machinery (ACM). When it was established in New York City in 1947, it stated that:

The purpose of this organization would be to advance the science, development, construction, and application of the new machinery for computing, reasoning, and other handling of information.(1)

The so-called “new machinery for computing” was early computers, and ACM considered it a machinery for “reasoning .. of information”.

However, the computers are not the only models for information change. The exciting history of derivations (or so called syllogisms) go back till Aristotle, and gained so much importance in Medieval ages via the rise of logical discourse among religious scholars. The manipulation of information was investigated by numerous philosophers including Aristotle, Boethius, Abelard etc (3). The very same tradition of information manipulation based on some strict rules survived until the ground breaking work of George Boole. When people still utter the fact that the computers depend on Boolean logic, not only a true fact of formal sciences is exposed, but also the significance of the idea formalization of a structure is underlined by praising G. Boole - the logician.

There are several methods and strategies to formalize the intuitions that computers use1. The engineering based approach shaped the mechanism and

1What makes the reasoning of computers an intuition can be verified from many perspective. Lack of a metamathematical theory of algorithms, for instance, show that a precise and mathematically meaningful definition of the term algorithms is still missing (17). In order to clarify this point, consider a very simple problem $P$ and assume that there are two algorithms $σ_1$ and $σ_2$ that solve the given problem. Assume further that both algorithms have the same time and space complexity. Then, what distinguishes these two algorithms? They calculate the same thing in the same time by using the same space. The most naive approach, without doubt, might be to consider $σ_1$ and $σ_2$ as strings and thus to compare the length of the strings as a measure for algorithms.

The point we underlined is called the problem of rigour in philosophy of mathematics and exhibits a very fragile issue in the methodology of mathematics (9; 4). We claim also that the very same issue is extremely visible in the analysis algorithms. Therefore, the paradigm we advocate here, i.e. the paradigm of game theory as an intuitive computation, might as well be useful in the theory of algorithms. Christos H. Papadimitriou seems to agree with that point (12; 13).
the design of computers in the early years for obvious reasons. Fortunately, except from the circuits and cables, there exist several other mathematical models that can precisely establish a model for such computation. We will present the game theoretical semantics as a new and alternative paradigm for computation.

In this paper, we will present some widely known models for machine computations. The first is called Temporal Logic of Actions which is essentially a dynamic modal logic interpreted in Kripke structures with a temporal modality for the time parameter. The dynamic aspect is characterized by the so-called actions which are meant to formalize the changes in the states. Temporal approach, on the other hand, characterizes and sometimes measure the change with respect to time. The second approach is automata based and thus uses rather modern techniques and understanding for computations.

Now, what makes logical models good for distributed systems? Lamport put it as follows: “temporal logic is a good method for specifying and reasoning about concurrent program” (11). In other words, the multi-agent circumstances (in which more than one processor, computer or agent is present so that the computation necessitates a distribution of the computation or an order of the computation among the agents) present a challenging aspect for formal scientists. Yet, the multi-agent version of temporal logic turns out to be a rather quick generalization with a heavy mathematical apparatus. Therefore, in order to avoid the technical mathematics for the sake of making our arguments clearer, we will only focus on the single agent temporal logic of actions. The interested reader is referred to (7) to see how complex multi agent logic can get.

The paper is organized as follows. First, we will provide a basic introduction to logic of actions and then embed it to the temporal setting. Second, we will briefly introduce timed input/output automata which is a general case for timed automata. Then, we will use some game theory to go deeper in temporal logic of actions in such a way that game theoretical analysis will provide us a computable and finite way to understand how the temporal logic of action works. Finally, we will briefly introduce the tableau rules as a simple proof theory for temporal logic of actions.

2 Temporal Logic

Temporal logic of actions (TLA) was motivated to reason about concurrent algorithms (10). Before introducing TLA, let us briefly recall the use of temporal logic in theoretical computer science. As Lamport indicated,

An execution of an algorithm is often thought of as a sequence of steps, each producing a new state by changing the values of one or more variables. We will consider an execution to be the resulting sequence of states, and will take the semantic meaning of an algorithm to be the collection of all it possible executions. Reasoning
about algorithms will therefore require reasoning about sequences of states. (10).

As we remarked earlier, the lack of a very precise definition of an algorithm still challenges computer scientists and mathematicians. Thus, the account of temporal logic might as well be read as an attempt to unravel the intuition of algorithms. Due to similarities between a state diagram and a Kripke structure, the immediate use of modal logic in theoretical computer science has become a prominent approach.

Temporal logic is the propositional logic extended with a unary modal operator □ that reads *always*. A formula in temporal logic is constructed by using Boolean operators and the modal operator by following the familiar syntactic rules. The dual of □ operator is ♦ and defined as follows: ♦F ≡ ¬□¬F. Therefore, □F means “F is always (i.e. at all times in the future) the case” whereas ♦F means “F is eventually (i.e. at some time in the future) the case”. Observe that the modal operators cannot look back. In other words, they cannot talk about the past.

The semantics of the temporal modal formulae is as usual for Boolean cases. For modal cases, we will use behaviors to give the semantic meaning of [□F] for a formula F. A behavior is an infinite sequence of states, and we will use this idea to represent the always truthfulness of □F. We thus define

\[ \langle s_0, s_1, \ldots \rangle[□F] := \langle s_n, s_{n+1}, \ldots \rangle[F], \text{ for all } n \in \mathbb{N} \]

We want to remark that the representation of states in TLA is rather different than that of the possible world semantics that people are familiar from Kripke semantics. In other words, behaviors expose the future states. Therefore, the locality motto of modal logics is rather modified in this representation (16). Moreover, remark that in the aforementioned representation the agent can foresee the future states without even knowing the entire mathematical model and entire structure. Because, the behavior includes the information about all accessible states starting from the actual state. This is very unusual considering the Kripke semantics².

Furthermore, we can express several usual characteristics in this language, too. Given the semantics, we can represent some properties by using this language. Let us reproduce several of them here in order to make the intuition behind the semantics clearer.

**Eventually** The formula [♦F] asserts that F is eventually true. Considering the previously given definition of ♦, we leave it to the reader to verify the following definition.

\[ \langle s_0, s_1, \ldots \rangle[♦F] := \langle s_n, s_{n+1}, \ldots \rangle[F], \text{ for some } n \in \mathbb{N} \]

²However, the taste of tower modal models can be felt in the notion of behavior. See (5) for further information on tower models of modal logics.
Infinitely Often  If $F$ is true at infinitely many times during the behavior, then the behavior satisfies $\square \Diamond F$. By the previous definitions, it can be seen that the definition of $\square \Diamond F$ is as follows.

$$\langle s_0, s_1, \ldots \rangle [\square \Diamond F] := \forall n \in \mathbb{N}, \exists m \in \mathbb{N} \langle s_{n+m}, s_{n+m+1}, \ldots \rangle [F].$$

Leads To  The formula $\Box (F \rightarrow \Diamond G)$ is true if and only if at any time when $F$ is true, $G$ is true at that time or at some later time. We leave it to the reader to convince herself that this formula in fact represents the given statement.

3 Logic of Actions

Logic of actions is intended to formalize the dynamic aspect of modal models. Depending on the context, the actions might differ. In epistemic situations, a public announcement of private announcement can be considered an action as they lead to a change (or update) in model. In propositional dynamic logic, a query (i.e. a Prolog query) can be an action.

In order to introduce the actions in temporal logic of actions, let us start with the fundamental definitions. A predicate is a Boolean expression constructed from variables and constant symbols. For example, $y = x + 3$ is a predicate where $x$ and $y$ are natural numbers. $s[P]$ will denote the validity of the predicate $P$ at state $s$.

The fundamental idea to formalize the programs might include a construction that will represent the updates of variables. Therefore, an action represents a relation between old states and new states. In our context, unprimed (e.g. $x$) variables refer to the old state whereas the primed (e.g. $x'$) variables refer to the new state. Thus, an action is a Boolean valued expression formed from variables, primed variables and constants. $y = x' + 3$ is then an action where $3$ is a constant, $x$ is a primed variable and $y$ is a variable. Therefore, the given example means the value of $y$ in the old state is three greater than the value of $x$ in the new state.

For an action $a$, we define a binary relation $[a]$ between states. If $s$ is an “old state”, and $t$ is a “new state”; then $s[a]t$ will be obtained from $a$ by replacing each unprimed variable $v$ by $s[v]$ and each primed variable $v'$ by $t[v]$. Hence the definition is as follows:

$$s[a]t := a(\forall v : s[v]/v, t[v]/v')$$

In order to illustrate the above concept, let us consider the previous example. For $a \equiv y' = x + 3$, we then have $s[y = x' + 3]t$ which is equivalent to the Boolean valued term $s[y] = t[x] + 3$.

Note that it is easy to see that predicates are also actions although they do not contain primed variables. Let us consider the predicate $P$ in the context of $s[P]t$. As $P$ has no primed variables, there will be no new states. Hence

$^3$This is rather different from the notations used in logic. Hence, $s[P]$ corresponds to $s \models P$. 


Temporal Logic of Actions

the validity of the given statement will depend only on $s$. Thus, $s[P]t$ for a predicate $P$ equals to $s[P]$.

However, we can define a new action $P'$ from $P$ by replacing each unprimed variable $v$ with the primed counterpart $v'$. Hence we define:

$$P' := P(v : v'/v)$$

Therefore, for the action $P'$ obtained from the predicate $P$, we observe that $s[P']t$ equals to $t[P]$.

An action $a$ is valid (write $|= a$) if and only if $s[a]t$ for all $s, t$ in the set of states. Similarly, a predicate $P$ is valid if and only if $s[P]$ for all states $s$.

4 Temporal Logic of Actions

Temporal logic of action is the modal temporal logic extended with actions. Given the temporal semantics, the truth of the modal actions is given as follows. For an action $a$, we have $\langle s_0, s_1, \ldots \rangle [\Box a] = \langle s_n, s_{n+1}, \ldots \rangle [a] \forall n \in \mathbb{N} = s_n[a]s_{n+1} \forall n \in \mathbb{N}$. Similar definitions can also be given for the predicate $P$. For a predicate $P$, we have $\langle s_0, s_1, \ldots \rangle [\Box P] = s_n[P] \forall n \in \mathbb{N}$.

As we pointed out already in the previous sections, temporal logic is a suitable and an intuitive way to represent algorithms. Now let us present a simple example to illustrate how TLA works. The example program starts of by fixing the variables $x$ and $y$ to zero, and increases either $x$ or $y$ repeatedly. The algorithm will choose $x$ or $y$ nondeterministically. The program is given as follows in a pseudo-code in Figure 1.

```
start
var x = 0, y = 0
do
choose
{x = x + 1}
or
{y = y + 1}
esoohc
od
goto start
```

Figure 1: An example program given in pseudo-code
We can formalize the given program as follows. The formula \( I \equiv (x = 0) \land (y = 0) \) will formalize the initial state. The formula \( C_1 \equiv (x' = x + 1) \land (y' = y) \) updates the \( x \) value whereas \( C_2 \equiv (x' = x) \land (y' = y + 1) \) updates the \( y \) value. Then the formula \( F \equiv I \land \Box(C_1 \lor C_2) \) represent the above program (10).

Considering the previous observation, we may, for the sake of the argument, insist that both \( x \) and \( y \) should be incremented infinitely often. Then, the proper representation of the program under that restriction would be \( I \land \Box(C_1 \lor C_2) \land \Box\Diamond C_1 \land \Box\Diamond C_2 \).

We can also formalize the states in which variables are not updated. In order to represent such cases, let us introduce a notation. \( [a]_x \) will denote the cases when either the action \( a \) is performed or we update \( x \) by itself. So, \( [a]_x := a \lor (x' = x) \). Dually, \( \langle a \rangle_x := a \land (x' \neq x) \) represent the case that the action \( a \) is carried out and the value of \( x \) is changed.

### 4.1 Axioms

For the sake of the completeness of the present paper, let us briefly mention the axioms of TLA. The axioms of the temporal logic of actions are given as follows.

1. Axioms of Propositional Logic.
2. \( \Box F \rightarrow F. \)
3. \( \Box\Box F \equiv \Box F. \)
4. \( F \rightarrow G \; \therefore \Box F \rightarrow \Box G. \)
5. \( \Box(F \land G) \equiv \Box F \land \Box G. \)
6. \( \Diamond\Box F \land \Diamond\Box G \equiv \Diamond(\Box F \land G). \)

We will not give the detailed expositions of the axioms as they are rather self-explanatory. The interested reader is referred to (10) for detailed explanations. However, remark that TLA is an S4 logic. Due to the lack of symmetry (as the modal operator cannot access the past), we do not have an S5 logic.

Having discussed the TLA, we will now very briefly mention timed input/output automata to establish the immediate connections between two formalizations.

### 5 Timed Input / Output Automata

Timed Input / Output Automaton (TIOA, afterwards) is a nondeterministic, possibly countably infinite-state state machine which was developed to formalize the distributed systems and multi-agent computational structures (8).

Formally, a TIAO is a tuple \( A = (X, Q, S, E, I, D, T) \) where

- A set \( X \) of (internal) variables.
A set \( Q \subseteq \text{val}(X) \) of states where \( \text{val}(X) \) denotes the valuation of \( X \).

- A nonempty set \( S \subseteq Q \) of start states.

- A set \( E \) of external actions and a set \( I \) of internal actions which are disjoint.

- A set \( D \subseteq Q \times (E \cup I) \times Q \) of discrete transitions.

- A set \( T \) of trajectories. (8)

The relation between automata and Kripke structures has very well been investigated. Especially, the connection between Büchi automata and modal-\( \mu \) calculus exhibits a fascinating field of study. We will not repeat them here. However, the interested reader is referred to (16) for a brief yet involved account.

6 TLA and TIOA

In this section, we will discuss how to obtain a TLA model from a given TIOA. Let \( A = (X, Q, S, E, I, D) \) be a given TIOA. Now, let us construct the TLA model \( M \) of \( A \). Let \( X \) be the set of variables for the TLA model \( M \). Let \( Q \) be the set of states for \( M \). Similarly, we will have \( D \) as the underlying relation of the TLA model \( M \). Recall that \( D \) has members in the form of \((q,a,q')\) where \( q, q' \) are states and \( a \) is an action. Therefore the relation we have in the model is labeled. We will say in the TLA model that \( q \) and \( q' \) are \( a \) related, and will denote this situation as \( qRa q' \). Moreover, the external and internal actions (which can be considered output and input actions respectively) are also included in the TLA model \( M \). Hence, we have \( M = (Q, \{ R_a \}_a, \text{val}(X), E \cup I) \) as the TLA model. If \( S \neq \emptyset \) then the states \( s \) in \( S \) will be the roots of the TLA model.

![Figure 2: A simple TIAO](image)

Let us consider the following simple TIAO. Assume, \( s \) is a starting state. This automaton at the state \( s \) when the input action \( a \) is received, yields the output action \( b \) and moves to the state \( t \). Therefore, we can represent the situation with the following formula

\[ s[\Diamond a][\Diamond b]t \]
Recall that the only behavior we have in the given automaton is \( (s, t) \). Hence the \( \Box \) operator is satisfied. We interpret the given sequence as follows. After the each input of the action \( a \) (we have only one in this situation) to the state, we get only the output actions \( b \) and move to \( t \).

Therefore, we say \( s[\Box a][\Box b]t \) is accepted by the given automaton. Similarly, we will say that the formula \( s[\Box a][\Box b]t \) is satisfied in the corresponding TLA model \( M \). For simplicity we will write \( s[\Box a][\Box b]t \) instead of \( s[\Box a][\Box b]t \); and similarly we will write \( s[ab]t \) instead of \( s[a]t \). Semantically, it is rather clear that \( s[\Box a][\Box b]t \) is satisfied as for all behavior (which is only \( (s, t) \) here) we have \( s[ab]t \). Moreover, as TLA is reflexive, it satisfies the axiomatic schema \( \Box \varphi \rightarrow \varphi \). Use of this schema also simplifies the equations.

Let us now consider a bit more complicated example. By following the same arguments, it is easy to see that

\[
s[\Diamond a][\Box b]t
\]

is satisfied. Observe that there are several options for input, \( a \) or \( c \). Thus, we represent it by the use of \( \Diamond \) operator. However, after, for instance, action \( a \) is chosen, the only output possibility is \( b \), hence we use \( \Box \) operator.

![Figure 3: A rather complicated TIAO](image)

Therefore, given any TIOA model, we can obtain a TLA model. Observe that, this procedure, is basically renaming automaton actions as relations. This is based on the fact that, in automata, the actions are relations between states - this is exactly what we want in modal models.

7 Ehrenfeucht - Fraïssé Games

The game theoretical semantics to establish the equivalence of different structures have been studied extensively. Equivalence between (Buchi) automata and modal fixed point logics; Ehrenfeucht-Fraisse games and automata, and equivalence between geometrical models of some epistemic logics have been presented and welcomed with a great interest (16; 2; 15). In this work, we will first calculate whether a given formula \( \varphi \) holds at a given behavior by TLA evaluation games.
The positions in the game will be of the form \((\varphi, (s_0, \ldots))\) where \(\varphi\) is a well formed formula in the language of TLA and \((s_0, \ldots)\) is a behavior. The player of the game will be called \(\forall\)belard and \(\exists\)loise by following the tradition. \(\forall\) is the adversary player who forces \(\exists\) to fail in such a way that she will not have any move to play. When any player has no move to make, then s/he will lose. As \(\forall\) is the adversary player, when \(\exists\) wins the game, it will mean that the formula is valid at the given behavior.

There is one syntactic restriction. We will work with formulae in the positive normal form. Recall that a formula \(\varphi\) is in positive normal form if \(\varphi\) has no negation symbol, or \(\varphi \equiv \neg \psi\) where \(\psi\) has no negation symbol. Observe that, \(\neg\) symbol changes the roles of \(\forall\) and \(\exists\). In other words, if \(\forall\) wins the game for \(\neg \varphi\), it means that \(\exists\) wins the game for \(\varphi\) starting from the same behavior. This observation is accurate as the game is a zero-sum game (one wins when the other loses). Thus, we fix the roles of the players at the beginning of the game by restricting our attention to the formulae in the positive normal form. However, this restriction does not make us to ignore any formula in the language as each formula has a positive normal form as it is shown in the following lemma.

**Lemma 1.** Any formula in the language of TLA has a positive normal form.

**Proof.** Proof is on the induction of the complexity of the formula \(\varphi\).

If \(\varphi\) is propositional variable, primed variable or constant, then the result is trivial. Likewise if \(\varphi \equiv \neg \psi\) for some positive \(\psi\), the result is then obvious. For the Boolean cases, let us assume \(\varphi \equiv \psi \land \chi\). By induction hypothesis, \(\psi\) and \(\chi\) have positive normal forms, say \(\psi \equiv \neg \psi'\) and \(\chi \equiv \neg \chi'\) where \(\psi', \chi'\) have no negation symbol. Then, by substitution we obtain \(\varphi \equiv \neg \psi' \land \neg \chi'\). By DeMorgan’s Law from propositional logic, we see \(\varphi \equiv \neg (\psi' \lor \chi')\). Hence the result follows.

We leave the actions case to the reader. 

TLA evaluation game \(\mathcal{E}(\varphi, M)\) for the TLA model \(M\) is a board game with players \(\exists\) and \(\forall\) moving a token around the positions of the form \((\psi, (s_0, \ldots))\) where \(\psi\) is a subformula of \(\varphi\) and \((s_0, \ldots)\) is a behavior. The rules of the game is given below.
Observe that, since the language of TLA is an extension of propositional modal logic of actions, evaluation games will be finite, thus will terminate⁴.

Winning conditions can be formulated as follows: ∃ wins if ∀ gets stuck, and dually, ∀ wins if ∃ gets stuck. As a matter of notations, we will denote the winning positions for ∃ by $Win_\exists (E(\varphi, \langle s_0, \ldots \rangle))$. However, one can feel the basic intuition behind the evaluation games by considering the winning positions. We will observe in the next theorem that, if a position $\langle s_0, \ldots \rangle$ is in the set of winning positions for ∃, then it is equivalent to say $\langle s_0, \ldots \rangle$ satisfies $\varphi$ in the model $M$.

**Theorem 1 (Adequacy Theorem for TLA Evaluation Games).**

$$(\varphi, \langle s_0, \ldots \rangle) \in Win_\exists (E(\varphi, \langle s_0, \ldots \rangle)) \text{ if and only if } \langle s_0, \ldots \rangle \text{ satisfies } \varphi.$$  

**Proof.** Proof goes by induction on the length of the formulae. The case for propositional variables and Boolean operators are easy and hence skipped. The case for negation is also easy. These cases are straightforward applications of the rules of the game. As we work with positive normal formula, if the given formula is a negation, then the roles are swapped.

Let us consider the more complex case. For the modal formula $\varphi \equiv \Diamond \psi$, we will have the following. Assume we are in the position $\langle \Diamond \psi, \langle s_0, \ldots \rangle \rangle$, and this is a winning position for ∃. We will show that, the given behavior satisfied the formula $\Diamond \psi$. Then it is ∃’s turn according to the rules of the game. She will pick a point $s_n$ and move to $\langle \psi, \langle s_n, \ldots \rangle \rangle$. As $\langle \Diamond \psi, \langle s_0, \ldots \rangle \rangle$ is a winning position for ∃, then so is $\langle \psi, \langle s_n, \ldots \rangle \rangle$ as it is the next position in the winning strategy of ∃. Remember that, if ∃ plays according to her winning strategy, each move is still in the winning strategy. Then, by induction hypothesis we see $\langle s_n, \ldots \rangle$ satisfies $\psi$. But then, we see $\langle s_0, \ldots \rangle$ satisfies $\Diamond \psi$.

The converse direction for the case that $\varphi \equiv \Diamond \psi$ can be shown by establishing the contrapositive of the claim. So, this is left to the reader.

---

⁴In order to cover the infinite loops, some authors used fixed point logics for infinite and finite loop iterations. For a detailed coverage of the aforementioned aspect, see (16).
The case for $\varphi \equiv \Box \psi$ is similar to the case above and thus we omit it.
Consider the case $\varphi \equiv [a]$. According to the rules, ground actions only unravel the given action formula. The player $\forall$ unravels the formula and moves to the next stage. And in the next stage, the player - whose turn is determined by the rules given in the table - continues. Therefore, this is just a reduction/simplification rule for the game to continue.

The cases for $\varphi \equiv [\Box a]$ or $\varphi \equiv [\Diamond a]$ is exactly similar to the above cases, so we skip them. Hence the theorem is proved.

Above theorem established that we can play a game to determine whether a given formula in TLA holds in a given behavior. In a similar fashion, we can also determine whether given two TLA models satisfy the same formula of a certain depth at given behaviors by playing a game. This game is thus called a bisimulation game. What is then a bisimulation? Bisimulation is a process equivalence. It is a stronger notion that an isomorphism as isomorphisms do not cover accessible states. Two models are bisimular if they satisfy the same formula in the given behaviors and further satisfy the same modal formula in their accessible behaviors. When the two behavior $\langle s_0, \ldots \rangle$ of the TLA model $M$ and $\langle s'_0, \ldots \rangle$ of the TLA model $M'$ are $n$-bisimular (i.e. bisimular up to length $n$), we denote it by $\langle s_0, \ldots \rangle \bisim_n \langle s'_0, \ldots \rangle$.

In bisimulation games, $\forall$ and $\exists$ compare the behaviors across the respective TLA models. $\exists$ wins if the given two behaviors are bisimular in their respective models and, $\forall$ wins otherwise. A TLA bisimulation game of length $n$, then can be defined as a game which can distinguish formulas of depth at most $n$. The game depends on $n$. The reason for that is the fact that, the formulae with depth of $m$ where $m > n$ would require more than $n$ steps to unravel, therefore the length of the game will be greater than $n$. In other words, longer formulae require longer games to compare them. It is then easy to observe that, $\exists$ has a winning strategy in the bisimulation game of length $n$ for two behaviors if and only if these two behaviors are actually bisimular for formulas of depth at most $n$.

It is easy to see that when $\exists$ has a winning strategy in the bisimulation game, then the given behaviors are bisimular. The proof goes by induction on the depth of the formulae and the application of straightforward ideas that we presented in the previous proof, thus we skip it.

**Theorem 2** (Adequacy Theorem for TLA Bisimulation Games).

$\langle s_0, \ldots \rangle \bisim_n \langle s'_0, \ldots \rangle$ if and only if $\exists$ has a winning strategy in the TLA bisimulation game of length $n$.

**Proof.** Left to the reader.

To conclude, our strategy in this paper was to reduce a given TIOA to a TLA model and then play games in the TLA. The reason we played games in TLA structures is the fact that TLA models are Kripke structures and thus exhibit a rather appropriate model for games.
The ideas we presented are not unique. As we underlined in the Introduction, use of game theoretical structures are gaining interest among computer scientists. We followed the current and tried to present a simple application of these constructions in the context of TLA. However, the rule for the action cases \([a]\) is rather different from the rules of normal modal logics. \([a]\) exhibits a rather strange construction. It only replaces some certain variables with some other variables. Therefore, the move it endorses is rather a simplification or reduction. Therefore, we introduced a rule that can express this fact.

8 Tableaus for TLA

Game theoretical semantics implicitly uses a proof tree (or game tree). Therefore, without referring to the games, one can establish a tableau method for TLA. For the completeness of the present paper, we will very briefly mention tableaus. As the tableau rules for normal modal logics are given in (6), we will not repeat them here. Instead, we will provide the missing rules for actions.

Let \(\sigma\) denote a prefix in a tableau, and if \(n\) is an integer, similarly \(\sigma.n\) will be a prefix as well. We give the tableau rules for logic of actions as follows. For an action \(a\),

\[
\frac{\sigma [a]}{\sigma [a']}
\]

where \(a'\) is obtained from \(a\) by replacing each unprimed variable by \(\sigma[v]\) if \(\sigma\) is not an end state. If \(\sigma\) is an end state, then \(a'\) is obtained by replacing each primed variable by \(\sigma[v]\). In other words, we unravel the unprimed variables with respect to the current state \(\sigma\), and once we are done with unprimed once, i.e. once we are at an end-state (i.e. end of a loop in a program), we deal with the primed variables.

The rules for modal cases is as expected.

\[
\frac{\sigma [\Diamond a]}{\sigma.n [a]}
\]

where \(\sigma.n\) is a new prefix.

\[
\frac{\sigma [\Box a]}{\sigma.n [a]}
\]

where \(\sigma.n\) is any prefix. The rules for negated modalities can be constructed by duality identities.

9 Epilogue

In this paper, our main aim was to provide a game theoretical semantics of TLA models. Our implicit aim was, however, to provide a game theoretical understanding of actions. Clearly, the topic is deep, we only covered a very small
portion of it. The further research areas in this topic include the connection between TIOA and modal fixed point logics.

Yet another research direction would be to use non-temporal (i.e. alethic) modalities in some extended dynamic modal languages. For instance, instead of using a transition mapping, one can use $E$ modality which reads $E\varphi$ as $\varphi$ is true somewhere in the model. Therefore, instead of specifying the transition, one can only state that the given state is transmitted to some other without explicitly stating which one. This would provide, not only extreme nondeterminism but also a full insight how a decision theoretical move can be made under uncertainty.

The literature also provides rather mathematical results on the connection between TLA and second order theories. Rabinovich, investigates and compares TLA with monadic second order logic which is in essence complex and complicated already (14). We will conclude this work with a brief remark on the literature.

A Note on the Literature  A significant amount of the research on the relation between TLA and concurrency and distributed programming have been carried out by L. Lamport. However, some of the remarks he made in his papers are simply misleading if not wrong. Let us mention only one of these misleading remarks here for illustrative purposes. For instance in (11), he says, while try to argue on the use of TLA, that

However, a formal description does not tell us what temporal logic is really about. Just as Peano’s axioms would be meaningless if we didn’t know they were about the integers, temporal logic is meaningless without an understanding of the models that underlie it.

These unfortunate lines are far from reflecting the formal semantics of any logical and axiomatic systems. First, a logic is just a class of formulae whereas a model is a structure that realizes this logic. Therefore, calling a logic meaningless simply sounds like expecting too much from a formal structure and consequently get disappointed. What Lamport tried to remark is the interpretation of the logical symbols (i.e. variables, constants, relation and function symbols) in a specific set. But, this is not the logic, this is just a model of the underlined logic. Lamport seems to confuse them.

Moreover, the remark on Peano axioms simply reflects the missing point. Natural numbers (not integers) is a model for Peano axioms. $(\mathbb{N}, 0, +, *, <)$ is called the standard model of Peano arithmetic. However, there are continuum many other nonstandard models of Peano arithmetics which have simply nothing to do with natural numbers in a formal sense. Therefore, Peano axioms are validated in infinitely many structures and natural numbers is only one of them. Natural number system does not necessarily justify or heuristically motivate the understanding and comprehension of other models of Peano arithmetic.

In conclusion, it appears that Lamport’s claims and remarks sometime go beyond the initial intentions. Therefore, we needed to ignore several comments
of Lamport in order to provide an understandable presentation of the topics discussed in this paper.

Bibliography


