A Game Based Approach to CTL* Model Checking

Martin Lange

LFCS, Division of Informatics
The University of Edinburgh
King's Buildings, Mayfield Road
Edinburgh EH9 3JZ
Scotland, UK
martin@dcs.ed.ac.uk

Supervisor Colin Stirling
LFCS, Division of Informatics
The University of Edinburgh

Keywords: model checking, verification, parallel processes, concurrent systems, games, temporal logic, full branching time, CTL*, interactive plays

Abstract. We introduce a definition of model checking games for the full branching time logic CTL* and sketch a proof of their correctness. These games are a helpful technique for using model checking in the verification of concurrent systems, because they may not only show that a specified property is violated, but also why it is.

1 Verification of Concurrent Systems

Mathematically, model checking is the process of deciding whether a structure as an interpretation satisfies a formula. This makes it a useful and broadly accepted technique for verifying parallel processes. However, verification of concurrent systems is often done in combination with specification in the framework of developing them. For such a process a simple yes/no answer, that a model checking algorithm usually yields, is not sufficient. Moreover, techniques that show why or where a certain property is violated, are required.

2 The Game Based Approach

Model checking games being played by two players on the structure and the formula provide the feature that has been outlined in the first section. Answering the question about the property being fulfilled turns out to be equivalent to finding a winning strategy for one of the players. Once such a strategy is found, i.e. computed by a verification tool for example, it can be used to enable an interactive play between the tool and the developer.

In this case we will deal with transition systems and the full branching time logic CTL* [2] only, although there are other models for concurrency, too. CTL* subsumes other temporal logics like LTL [5] and CTL [1] syntactically, and thus the results on the games for CTL* easily carry over to these other logics as well.

On the other hand, CTL* can be translated into the modal μ-calculus [3], for which such model checking games already exist [6]. However, using those games for checking properties being specified in CTL* exhibits a major disadvantage. The play does not proceed on subformulas of the original formula. That makes it hard to understand why a certain property is violated.

After recalling the syntax and semantics of CTL*, we will give a definition of how model checking games for CTL* may look like. The first goal is to prove their correctness, i.e. to show that the mentioned equivalence between winning strategies on the games' side and the satisfaction on the logical side really holds. The proof of this theorem is expected to be constructive, so that it already sketches an algorithm for deciding the model checking problem. The algorithm ought to be local, i.e. it must be possible to expand the transition system step by step throughout the model checking process. Its time complexity is expected to be exponential, for the problem was shown to be PSPACE-complete [4].

The desired interactive games that can be played by a verification tool and its user usually rely on the data structures that were used by the model checking algorithm. In this case playing means following a path through a graph, following the game rules and a winning strategy. Yet it is not known whether the strategies for these games are history free or not.

There is a fairly simple way of defining games for CTL* following exactly the semantics. This would involve players to choose whole paths in the transition system which contradicts the locality and would result in a horrendous space complexity as well. Hence, another requirement to the games is the ability to choose paths stepwise. Indeed, we will call this the main requirement.

3 Model Checking Games for CTL*

Let $Prop = \{\mathtt{tt}, \mathtt{ff}, Q_1, \overline{Q_1}, \ldots\}$ be a set of propositional constants. A *transition system* $\mathcal{T}$ is a triple (S, T, L) with (S, T) being a directed graph. $L : S \to 2^{Prop}$ labels the states, such that for all $s \in S$: $\mathtt{tt} \in L(s), \mathtt{ff} \notin L(s)$ and $Q \in L(s)$ iff $\overline{Q} \notin L(s)$. We assume that every state in the graph has a successor. The syntax of CTL* is defined by

$$\varphi \quad ::= \quad Q \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid E\varphi \mid A\varphi$$

where Q ranges over $Prop$. The set of subformulas $Sub(\varphi)$ for a given φ is defined in the usual way. Additionally, $X(\varphi U \psi), \varphi \wedge X(\varphi U \psi), \psi \vee (\varphi \wedge X(\varphi U \psi))$ are subformulas of $\varphi U \psi$, too. Similarily, $X(\varphi R \psi), \varphi \vee X(\varphi R \psi), \psi \wedge (\varphi \vee X(\varphi R \psi))$ are subformulas of $\varphi R \psi$ as well.

The semantics of a CTL* formula is explained using paths $\pi = s_0 s_1 \ldots s_n \ldots$ of a transition system. With $\pi^{(i)}$ we denote the suffix of π beginning with the the state s_i.

- $\mathcal{T}, \pi \models Q$ iff $Q \in L(s_0)$
- $\mathcal{T}, \pi \models \varphi \wedge \psi$ iff $\mathcal{T}, \pi \models \varphi$ and $\mathcal{T}, \pi \models \psi$
- $\mathcal{T}, \pi \models \varphi \vee \psi$ iff $\mathcal{T}, \pi \models \varphi$ or $\mathcal{T}, \pi \models \psi$
- $\mathcal{T}, \pi \models A\varphi$ iff for all paths $\sigma = s_0\sigma'$ implies $\mathcal{T}, \sigma \models \varphi$
- $\mathcal{T}, \pi \models E\varphi$ iff there exists a path $\sigma = s_0\sigma'$ and $\mathcal{T}, \sigma \models \varphi$
- $\mathcal{T}, \pi \models X\varphi$ iff $\mathcal{T}, \pi^{(1)} \models \varphi$
- $\mathcal{T}, \pi \models \varphi U \psi$ iff there exists $i \in \mathbb{N}$ s.t. $\mathcal{T}, \pi^{(i)} \models \psi$ and for all $j < i$: $\mathcal{T}, \pi^{(j)} \models \varphi$
- $\mathcal{T}, \pi \models \varphi R \psi$ iff for all $i \in \mathbb{N}$: $\mathcal{T}, \pi^{(i)} \models \psi$ or there exists a $j \leq i$ s.t. $\mathcal{T}, \pi^{(j)} \models \varphi$

A formula φ is called a *state formula* if $\varphi \equiv A\varphi$.[1] In the following we will deal with state formulas only. Also, we will write $\mathcal{T}, s_0 \models \varphi$ in these cases. Formulas not being state formulas are called *path formulas*. They will still occur as subformulas of state formulas.

The games are played by two *players*, namely I and II.[2] It is player II's task to show that a formula is fulfilled whereas player I tries to show the opposite. A *game* may consist of several plays, and a *play* is a finite sequence of configurations following certain rules. A *configuration* is an element of $\{\mathrm{I}, \mathrm{II}\} \times S \times Sub(\varphi) \times 2^{Sub(\varphi)}$.

Take $\varphi := A(XQ \vee X\overline{Q})$. In a naive version of games one might keep record of the A-quantifier, making player I to choose the next state whenever the play reaches an X-formula because it is a universal move and thus has to be done by player I. The next playing position would be the $\vee$, so that player II, doing existential moves, selects XQ for example. Now player I might be able to choose a next state in which Q might not be valid. Indeed, player I would always win starting from a state that has two distinct successors, of which exactly one satisfies Q. However, the formula is a tautology, so player II should be able to win every game regardless of how the transition system looks like. This ambiguity arises from the fact that following the semantics the path must have been chosen by player I already when player II chooses at the $\vee$ position. We overcome this by allowing player II in this case to redo her move after player I has done his.

Given a transition system $\mathcal{T}$ with starting state s and a state formula φ every play of the game $\Gamma_{\mathcal{T}}(s, \varphi)$ begins with the configuration $\mathrm{I}, s \vdash [\varphi]$. In general, a configuration looks like $p, t \vdash [\psi], \Psi$, where p is a player called the *pathplayer*, $t \in S$, $\{\psi\} \cup \Psi \subseteq Sub(\varphi)$. In this case, ψ is called to be *in focus*. The game rules are given in figure 1 and are to be read downwards. For example,

$$\frac{\mathrm{I}, s \vdash [\varphi_0 \vee \varphi_1], \Phi}{\mathrm{I}, s \vdash [\varphi_i], \varphi_{1-i}, \Phi} \ \mathrm{II}$$

means: If the current configuration is the upper one then player II performs a choice on the formula and the next configuration will be the lower one.

[1] Hence, it is always possible to bring a state formula into the form $A\varphi$.

[2] In the following "he" will denote player I whereas "she" will stand for either player II or a general case.

(1) $\dfrac{\mathrm{I}, s \vdash [\varphi_0 \wedge \varphi_1], \Phi}{\mathrm{I}, s \vdash [\varphi_i], \Phi}$ I

(2) $\dfrac{\mathrm{II}, s \vdash [\varphi_0 \vee \varphi_1], \Phi}{\mathrm{II}, s \vdash [\varphi_i], \Phi}$ II

(3) $\dfrac{\mathrm{I}, s \vdash [\varphi_0 \vee \varphi_1], \Phi}{\mathrm{I}, s \vdash [\varphi_i], \varphi_{1-i}, \Phi}$ II

(4) $\dfrac{\mathrm{II}, s \vdash [\varphi_0 \wedge \varphi_1], \Phi}{\mathrm{II}, s \vdash [\varphi_i], \varphi_{1-i}, \Phi}$ I

(5) $\dfrac{p, s \vdash [\varphi U \psi], \Phi}{p, s \vdash [\psi \vee (\varphi \wedge X(\varphi U \psi))], \Phi}$

(6) $\dfrac{p, s \vdash [\varphi R \psi], \Phi}{p, s \vdash [\psi \wedge (\varphi \vee X(\varphi R \psi))], \Phi}$

(7) $\dfrac{\mathrm{I}, s \vdash [X\psi], \varphi_0 \wedge \varphi_1, \Phi}{\mathrm{I}, s \vdash [X\psi], \varphi_i, \Phi}$ I

(8) $\dfrac{\mathrm{II}, s \vdash [X\psi], \varphi_0 \vee \varphi_1, \Phi}{\mathrm{II}, s \vdash [X\psi], \varphi_i, \Phi}$ II

(9) $\dfrac{\mathrm{I}, s \vdash [X\psi], \varphi_0 \vee \varphi_1, \Phi}{\mathrm{I}, s \vdash [X\psi], \varphi_0, \varphi_1, \Phi}$

(10) $\dfrac{\mathrm{II}, s \vdash [X\psi], \varphi_0 \wedge \varphi_1, \Phi}{\mathrm{II}, s \vdash [X\psi], \varphi_0, \varphi_1, \Phi}$

(11) $\dfrac{p, s \vdash [X\chi], \varphi U \psi, \Phi}{p, s \vdash [X\chi], \psi \vee (\varphi \wedge X(\varphi U \psi)), \Phi}$

(12) $\dfrac{p, s \vdash [X\chi], \varphi R \psi, \Phi}{p, s \vdash [X\chi], \psi \wedge (\varphi \vee X(\varphi R \psi)), \Phi}$

(13) $\dfrac{p, s \vdash [A\varphi], \Phi}{\mathrm{I}, s \vdash [\varphi]}$

(14) $\dfrac{p, s \vdash [E\varphi], \Phi}{\mathrm{II}, s \vdash [\varphi]}$

(15) $\dfrac{p, s \vdash [\varphi], A\psi, \Phi}{p, s \vdash [\varphi], \Phi}\ \overline{p}$

(16) $\dfrac{p, s \vdash [\varphi], E\psi, \Phi}{p, s \vdash [\varphi], \Phi}\ \overline{p}$

(17) $\dfrac{p, s \vdash [\varphi], Q, \Phi}{p, s \vdash [\varphi], \Phi}\ \overline{p}$

(18) $\dfrac{p, s \vdash [X\varphi_0], X\varphi_1, \ldots, X\varphi_k}{p, t \vdash [\varphi_0], \varphi_1, \ldots, \varphi_k}\ p$

(19) $\dfrac{p, s \vdash [\varphi], \psi, \Phi}{p, s \vdash [\psi], \varphi, \Phi}\ \overline{p}$

Fig. 1. The game rules.

The player to do the next choice with one of the rules (1) – (18) is determined by the pathplayer and the focus. After each move the pathplayer's opponent is allowed to reset the focus with rule (19). A play is finished if it has reached a configuration

1. $p, t \vdash [Q], \Phi$, or
2. $\mathrm{II}, t \vdash [\chi U \psi], \Phi$ (resp. $\mathrm{I}, t \vdash [\chi R \psi], \Phi$) after the play already went through the same configuration and player I (resp. II) never applied rule (19) in between, or
3. $p, s \vdash [\varphi], \Phi$ for the second time possibly using rule (19) in between.

In the first case player II wins if $Q \in L(s)$, otherwise player I wins. In the second case player I wins if the formula in focus is $\chi U \psi$, and player II if it is $\chi R \psi$. In the third case p wins.

The main purpose of the focus is to pick out a particular formula from the set of formulas that are examined on a path, and to determine the winner. The set of formulas can be seen as a kind of insurance for the pathplayer's opponent and

are to be understood disjunctively if player I is the pathplayer, or conjunctively if it is player II. By setting the focus to a specific formula a player indicates which formula she wants to prove, respectively he wants to refute. Thus a player wins if she is able to stick to a particular formula and loses if she uses the focus resetting rule too excessively.

A player has a *winning strategy* for, or simply *wins*, a game if she can force every play into a winning position for herself. A *successful gametree* for player p is a tree in which all the paths are plays of a certain game. Moreover, whenever a configuration requires p to make a choice then there is only one successor in the tree unless it is a leaf. All choices of p's opponent are preserved in the tree.

The main result is the following:

Theorem 1. *Let $\mathcal{T}$ be a transition system with a state s and $\varphi \in \mathrm{CTL}^*$. Then $\mathcal{T}, s \models \varphi$ iff player* II *wins the game $\Gamma_{\mathcal{T}}(s, \varphi)$.*

The completeness proof uses a nested induction. The outer one is on the pathquantifier depths of the formula and reduces the original goal to formulas with one path quantifier only. It is justified by the following lemma which is not hard to prove.

Lemma 1. *In a game subformulas of the form $A\psi$ and $E\psi$ represent seperate games and, thus, can be considered as atomic propositions.*

The inner induction is on the syntactical structure of the remaining path-formula being guarded by one quantifier. This splits up into the two distinguishable cases $\varphi = A\psi$ and $\varphi = E\psi$. In any case, a successful gametree for player II is constructed from the fact that the transition system models the formula. The $E\psi$ case is less problematic because one can assume the existence of one good path only. However, in the $A\psi$ case one has to deal with a set of paths that may change throughout the induction step cases.

The soundness proof consists of a copy of the completeness proof with the roles of the players and boolean connectives, etc. being switched, if determinancy and duality of the games has been shown. Indeed, it is easy to prove the following lemmas.

Lemma 2. *Every play has a uniquely determined winner.*

Lemma 3.
a) Player II *wins the game $\Gamma_{\mathcal{T}}(s, \varphi)$ iff player* I *does not win $\Gamma_{\mathcal{T}}(s, \varphi)$.*
b) For every game one of the players has a winning strategy.

Example 1. Take a transition system $\mathcal{T}$ with just one state s satisfying the proposition Q and an edge from s to itself. The formula to be examined is $\varphi = E(\overline{Q}U(\mathtt{ff}RQ))$. $\mathcal{T}$ with state s satisfies φ. The successful gametree with annotated rule numbers is given in fig. 2.

Player II wins the play of the leftmost branch because of winning condition three, and the one right beside it because of condition two.

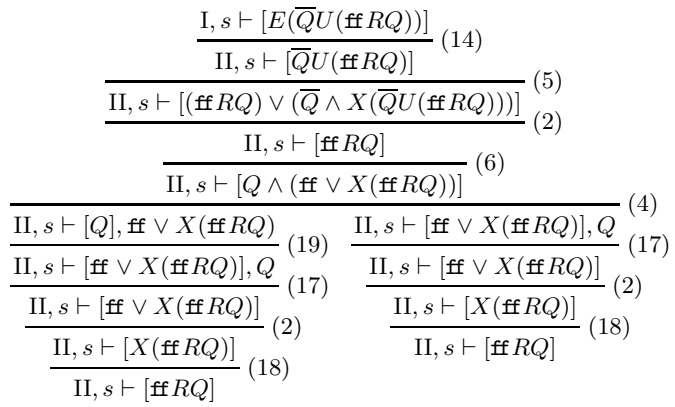

Fig. 2. A successful gametree.

It remains to examine the nature of winning strategies, as to say whether they are *history-free* or not. If they are then the winning strategy simply is the set of all edges (C, C') in the tree, such that the winner of the play has to do a choice in the configuration C. If they are not, meaning that certain choices done by the winner depend on recent choices of her own, then the strategy might be representable by a set of finite parts of paths in the gametree only.

References

1. E.M. Clarke and E.A. Emerson. Synthesis of Synchronization Skeletons for Branching Time Temporal Logic. In *Logics of Programs: Workshop*, volume 131 of *Lecture Notes in Computer Science*, Yorktown Heights, New York, May 1981. Springer-Verlag.
2. E. Allen Emerson and A. Prasad Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, June 1984.
3. Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, December 1983.
4. Faron Moller and Graham M. Birtwistle. *Logics for concurrency: structure versus automata*, volume 1043 of *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1996.
5. A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57, Providence, Rhode Island, October 31–November 2 1977. IEEE, IEEE Computer Society Press.
6. C. Stirling. Games for bisimulation and model checking, June 1997. Notes for Mathfit instructional meeting on games and computation, Edinburgh http://www.dcs.ed.ac.uk/home/cps/mathfit.ps.