Local Model Checking Games (Extended Abstract)

Colin Stirling*
Department of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ, UK
email: cps@uk.ac.ed.dcs

1 Introduction

Model checking is a very successful technique for verifying temporal properties of finite state concurrent systems. It is standard to view this method as essentially algorithmic, and consequently a very fruitful relationship between temporal logics and automata has been developed. In the case of branching time logics the connection has not been quite so tight as tree automata are not naturally the correct semantics of programs. Hence the introduction of amorphous automata with varying branching degrees, and the use of alternating automata.

Local model checking was proposed as a proof system approach to verification which also applies to infinite-state concurrent systems. In part this was because it predominantly uses the process algebra model of concurrency (such as CCS) where a concurrent system is presented as an expression of the calculus. The question of verification is then whether a particular expression has a temporal property (rather than all the states of a transition system which have a property). In local model checking the proof system is developed in a goal directed fashion, that is a top down approach. When a property holds, there is a proof tree which witnesses this truth. It also allows there to be proofs for infinite-state systems. Moreover local model checking permits compositional reasoning, when a proof tree may be guided by the algebraic structure of the system as well as the logical structure of the formula expressing the property.

In this talk we show that in the finite-state case these two approaches, model checking as essentially algorithmic and model checking as a proof system, can be combined using *games* as an underlying conceptual framework that can enjoy the best of both worlds. The automata theoretic approach is captured via the resulting game graph (which is an alternating automaton), and on the other hand a witness for non-emptiness of a game graph is just a proof tree. Alternatively a game graph can be translated into a formula of boolean fixed point logic, which has provided a variant framework for model checking. Game playing also provides a very perspicuous basis for understanding branching time temporal logics and model checking of finite and infinite-state systems.

An important question is whether finite-state model checking of modal mu-calculus properties can be done in polynomial time. Emerson showed that this

* Research in part supported by ESPRIT BRA project Concur2.

problem belongs to NP ∩ co-NP. Games provide a very direct proof of this result. It appears that finer structure needs to be exposed to improve upon this. We believe that new insights may come from the relationship between these games and other graph games. For example model checking games can be reduced to simple stochastic games, an observation due to Mark Jerrum, whose decision procedure also belongs to NP ∩ co-NP.

2 Modal Mu-Calculus

Modal mu-calculus, modal logic with extremal fixed points, was introduced by Kozen [13]. Formulas of the logic given in positive form are defined by

$$\Phi ::= Z \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [K]\Phi \mid \langle K\rangle\Phi \mid \nu Z.\Phi \mid \mu Z.\Phi$$

where Z ranges over a family of propositional variables, and K over subsets[2] of an action set $\mathcal{A}$. The binder νZ is the greatest whereas μZ is the least fixed point operator.

Modal mu-calculus with action labels drawn from $\mathcal{A}$ is interpreted on labelled transition systems $(\mathcal{P}, \{\xrightarrow{a} : a \in \mathcal{A}\})$ where $\mathcal{P}$ is a countable but non-empty set of processes (or states), and each $\xrightarrow{a}$ is a binary transition relation on $\mathcal{P}$. Labelled transition systems are popular structures for modelling concurrent systems especially process calculi such as CCS [17]: $\mathcal{P}$ is then a transition closed set[3] of process expressions and $E \xrightarrow{a} F$ means that E may evolve to F by performing the action a.

Assume a fixed transition system $(\mathcal{P}, \{\xrightarrow{a} : a \in \mathcal{A}\})$, and let $\mathcal{V}$ be a valuation which assigns to each variable Z a subset $\mathcal{V}(Z)$ of processes in $\mathcal{P}$. Let $\mathcal{V}[\mathcal{E}/Z]$ be the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ everywhere except possibly Z when $\mathcal{V}'(Z) = \mathcal{E}$. The subset of processes in $\mathcal{P}$ satisfying an arbitrary formula Ψ under the valuation $\mathcal{V}$ is inductively defined as the set $\|\Psi\|^{\mathcal{P}}_{\mathcal{V}}$ where for ease of notation we drop the superscript $\mathcal{P}$ which is assumed fixed throughout:

$$\|Z\|_{\mathcal{V}} = \mathcal{V}(Z) \qquad \|\Phi \wedge \Psi\|_{\mathcal{V}} = \|\Phi\|_{\mathcal{V}} \cap \|\Psi\|_{\mathcal{V}} \qquad \|\Phi \vee \Psi\|_{\mathcal{V}} = \|\Phi\|_{\mathcal{V}} \cup \|\Psi\|_{\mathcal{V}}$$
$$\|[K]\Phi\|_{\mathcal{V}} = \{E \in \mathcal{P} : \text{if } a \in K \text{ and } E \xrightarrow{a} F \text{ then } F \in \|\Phi\|_{\mathcal{V}}\}$$
$$\|\langle K\rangle\Phi\|_{\mathcal{V}} = \{E \in \mathcal{P} : E \xrightarrow{a} F \text{ for some } a \in K \text{ and } F \in \|\Phi\|_{\mathcal{V}}\}$$
$$\|\nu Z.\Phi\|_{\mathcal{V}} = \bigcup\{\mathcal{E} \subseteq \mathcal{P} : \mathcal{E} \subseteq \|\Phi\|_{\mathcal{V}[\mathcal{E}/Z]}\}$$
$$\|\mu Z.\Phi\|_{\mathcal{V}} = \bigcap\{\mathcal{E} \subseteq \mathcal{P} : \|\Phi\|_{\mathcal{V}[\mathcal{E}/Z]} \subseteq \mathcal{E}\}$$

Any formula Φ determines the monotonic function $\lambda\mathcal{E} \subseteq \mathcal{P}.\ \|\Phi\|^{\mathcal{P}}_{\mathcal{V}[\mathcal{E}/Z]}$ with respect to the variable Z, the valuation $\mathcal{V}$, and the set $\mathcal{P}$. Hence the meaning of the greatest fixed point is the union of all postfixed points, and it is the intersection of all prefixed points in the case of the least fixed point. One consequence is that the meaning of $\sigma Z.\Phi$ is the same as its *unfolding* $\Phi\{\sigma Z.\Phi/Z\}$ (where

[2] It is very convenient to allow sets of labels to appear in modalities instead of the usual single labels.

[3] If $E \in \mathcal{P}$ and $E \xrightarrow{a} F$ then $F \in P$.

$\Phi\{\Psi/Z\}$ is the substitution of Ψ for free occurrences of Z in Φ). Assume that **tt** (true) abbreviates $\nu Z.\, Z$, and **ff** (false) is $\mu Z.\, Z$. We use $E \models_{\mathcal{V}} \Phi$ as an abbreviation for $E \in \|\Phi\|_{\mathcal{V}}^{\mathcal{P}}$ when $\mathcal{P}$ contains E.

An alternative but equivalent interpretation of extremal fixed points is in terms of approximants. When $\sigma \in \{\nu, \mu\}$, and α is an ordinal let $\sigma Z^{\alpha}.\Phi$ be the α-unfolding with the following interpretation, where λ is a limit ordinal:

$$\begin{array}{llll} \|\nu Z^0.\Phi\|_{\mathcal{V}} & = \mathcal{P} & \|\mu Z^0.\Phi\|_{\mathcal{V}} & = \emptyset \\ \|\nu Z^{\alpha+1}.\Phi\|_{\mathcal{V}} & = \|\Phi\|_{\mathcal{V}[\|\nu Z^{\alpha}.\Phi\|_{\mathcal{V}}/Z]} & \|\mu Z^{\alpha+1}.\Phi\|_{\mathcal{V}} & = \|\Phi\|_{\mathcal{V}[\|\mu Z^{\alpha}.\Phi\|_{\mathcal{V}}/Z]} \\ \|\nu Z^{\lambda}.\Phi\|_{\mathcal{V}} & = \bigcap\{\|\nu Z^{\alpha}.\Phi\|_{\mathcal{V}} : \alpha < \lambda\} & \|\mu Z^{\lambda}.\Phi\|_{\mathcal{V}} & = \bigcup\{\|\mu Z^{\alpha}.\Phi\|_{\mathcal{V}} : \alpha < \lambda\} \end{array}$$

The set $\|\nu Z.\Phi\|_{\mathcal{V}}$ equals $\bigcap\{\|\nu Z^{\alpha}.\Phi\|_{\mathcal{V}} : \alpha \text{ is an ordinal}\}$ and $\|\mu Z.\Phi\|_{\mathcal{V}}$ is the same as $\bigcup\{\|\mu Z^{\alpha}.\Phi\|_{\mathcal{V}} : \alpha \text{ is an ordinal}\}$.

Modal mu-calculus is a very expressive propositional temporal logic with the ability to describe liveness, safety, fairness, and cyclic properties. It has been shown that it is as expressive as finite-state automata on infinite trees, and hence is as powerful as the monadic second-order theory of n successors [10]. This is a very general and fundamental decidable theory to which many other decidability results in logic can be reduced. Most propositional temporal and modal logics used in computer science are sublogics of mu-calculus. Consequently various subcalculi can be defined which contain other well known program logics. An important example is CTL which is contained within the *alternation free* fragment. This is the sublogic when the following pair of conditions are imposed on fixed point formulas:

if $\mu Z.\Phi$ ($\nu Z.\Phi$) is a subformula of $\nu Y.\Psi$ ($\mu Y.\Psi$) then Y is not free in Φ

This fragment of modal mu-calculus turns out to be very natural, for it is precisely the sublogic which is equi-expressive to *weak* monadic second-order theory of n successors[4]. This result follows from [3] where it is proved for Niwinski's tree mu-calculus. Other fragments of modal mu-calculus studied include those of k–alternation depth for any k, and those presented in [12].

3 Verification and model checking

Modal mu-calculus is a very rich temporal logic for describing properties of systems. Here are two examples presented purely for illustration:

Example 1 A finite-state CCS description of a level crossing from [5] consists of three components in parallel, the road, rail, and signal where the first two are simple cyclers.

$$\begin{array}{ll} \mathit{Road} & \stackrel{\text{def}}{=} \mathtt{car.up}.\overline{\mathtt{ccross}}.\overline{\mathtt{down}}.\mathit{Road} \\ \mathit{Rail} & \stackrel{\text{def}}{=} \mathtt{train.green}.\overline{\mathtt{tcross}}.\overline{\mathtt{red}}.\mathit{Rail} \\ \mathit{Signal} & \stackrel{\text{def}}{=} \overline{\mathtt{green}}.\mathtt{red}.\mathit{Signal} + \overline{\mathtt{up}}.\mathtt{down}.\mathit{Signal} \\ \mathit{Crossing} & \stackrel{\text{def}}{=} (\mathit{Road} \mid \mathit{Rail} \mid \mathit{Signal}) \backslash \{\mathtt{green}, \mathtt{red}, \mathtt{up}, \mathtt{down}\} \end{array}$$

[4] when the second-order quantifiers range over finite sets.

The actions **car** and **train** represent the approach of a car and a train, **up** is the gates opening for the car, and $\overline{\texttt{ccross}}$ is the car crossing, and **down** closes the gates, and **green** is the receipt of a green signal by the train, $\overline{\texttt{tcross}}$ is the train crossing, and **red** automatically sets the light red. A safety property for the crossing is that it is never possible to reach a state where a train and a car are both able to cross, $\nu Z.([\texttt{tcross}]\texttt{ff} \vee [\texttt{ccross}]\texttt{ff}) \wedge [-]Z$[5]. The desirable liveness property for the crossing is that whenever a car approaches then eventually it crosses $\nu Z.[\texttt{car}](\mu Y.\langle - \rangle \texttt{tt} \wedge [-\overline{\texttt{ccross}}]Y) \wedge [-]Z$ (and similarly for a train). But this only holds if we assume that the signal is fair, which is also expressible in the logic.

Example 2 A second (somewhat artificial) example is a description of an arbitrary new clock. Let Cl_i be a clock that ticks i times before terminating, and let *Clock* be $\sum\{Cl_i : i \geq 1\}$. Like all new clocks, *Clock* will eventually break down, so has the property $\mu Z.[\texttt{tick}]Z$. However it fails to have $\mu Z^n.[\texttt{tick}]Z$ for all $n \geq 0$.

A process expression determines a (unique) transition graph according to the rules for transitions. The graph associated with *Crossing* is finite-state whereas it is infinite-state in the case of *Clock*. The verification problem in this framework is to find techniques for determining whether or not a process description E has the property Φ (relative to stipulations as to what the free variables say as summarized by a valuation $\mathcal{V}$). One technique for checking whether process E has Φ is to first *construct* the transition graph for E (or even a larger graph), and then second to calculate $\| \Phi \|_{\mathcal{V}}$ with respect to this structure, possibly using approximants, and then finally check whether E belongs to it. This seems reasonable for determining whether a process such as the crossing has a property, as its transition graph is small. As a general method it is very cumbersome and clearly not directly applicable to infinite-state processes such as the clock. These were grounds for introducing local model checking as in [20, 6]. It was not intended as an algorithmic method, but as a proof system, and this is why it was developed using tableaux, for which in the finite-state one might be able to extract a reasonable algorithm. In fact this is one consequence of using games as an alternative logical basis for local model checking.

4 Simple games

We now present an alternative account of the satisfaction relation between a process and a temporal formula using games which underpins the local model checking proof systems in [20, 6]. A property checking game is a pair (E, Φ) relative to $\mathcal{V}$ for which there are two players, player I and player II. It is the role of player I to try to show that E *fails* to have the property Φ whereas player II attempts to frustrate this. Unlike more standard games, players here do not

[5] To cut down on brackets we write $[a_1, \ldots a_n]$ instead of $[\{a_1, \ldots a_n\}]$, and the same within a diamond operator, and $-K$ abbreviates $\mathcal{A} - K$ (and so $[-]$ is $[\mathcal{A}]$).

necessarily take turns[6].

A play of the *property checking game* (E_0, Φ_0) relative to $\mathcal{V}$ is a finite or infinite sequence of the form, $(E_0, \Phi_0)\ldots(E_n, \Phi_n)\ldots$. The next move in a play, the step from (E_j, Φ_j) to (E_{j+1}, Φ_{j+1}), and which player makes it is determined by the main connective of Φ_j. An essential ingredient is the use of auxiliary propositional constants, ranged over by U, which are introduced as fixed point formulas are met and can be thought of as colours. Suppose an initial part of a play is $(E_0, \Phi_0)\ldots(E_j, \Phi_j)$. The next step (E_{j+1}, Φ_{j+1}) is as below:

- if $\Phi_j = \Psi_1 \wedge \Psi_2$ ($\Psi_1 \vee \Psi_2$) then player I (player II) chooses one of the conjuncts (disjuncts) Ψ_i: the process E_{j+1} is E_j and Φ_{j+1} is Ψ_i.
- if $\Phi_j = [K]\Psi$ ($\langle K\rangle\Psi$) then player I (player II) chooses a transition $E_j \xrightarrow{a} E_{j+1}$ with $a \in K$ and Φ_{j+1} is Ψ.
- if $\Phi_j = \nu Z.\Psi$ ($\mu Z.\Psi$) then player I (player II) chooses a *new* constant U and sets $U \stackrel{\text{def}}{=} \nu Z.\Psi$ ($U \stackrel{\text{def}}{=} \mu Z.\Psi$): the process E_{j+1} is E_j and Φ_{j+1} is U.
- if $\Phi_j = U$ and $U \stackrel{\text{def}}{=} \nu Z.\Psi$ ($U \stackrel{\text{def}}{=} \mu Z.\Psi$) then player I (player II) unfolds the fixed point so Φ_{j+1} is $\Psi\{U/Z\}$ and E_{j+1} is E_j.

There is a strong duality between the rules for $\wedge$ and $\vee$, $[K]$ and $\langle K\rangle$, $\nu Z.\Psi$ and $\mu Z.\Psi$. In the case of the last pair new constants are introduced as abbreviations for those fixed points. Rules also govern occurrences of constants as the formula of the next step is the body of the defined fixed point when all the occurrences of the fixed point variable are replaced with the constant.

The rules for the next move in a play are backwards sound with respect to the intentions of the players. If player I makes the move (E_{j+1}, Φ_{j+1}) from (E_j, Φ_j) and E_{j+1} fails to satisfy Φ_{j+1} relative to $\mathcal{V}$ then also E_j fails to have Φ_j. In contrast when player II makes this move and $E_{j+1} \models_{\mathcal{V}} \Phi_{j+1}$ then also $E_j \models_{\mathcal{V}} \Phi_j$. In the case of a fixed point formula this is clear provided we understand the presence of a constant to be its defined equivalent. Formulas are no longer "pure" as they may contain constants. However we can recover a pure formula from an impure formula by replacing constants with their defined fixed points in reverse order of introduction: assuming that $U_1 \stackrel{\text{def}}{=} \Psi_1 \ldots U_n \stackrel{\text{def}}{=} \Psi_n$ is the sequence of declarations in order of introduction, the meaning of Ψ is just $\Psi\{\Psi_n/U_n\}\ldots\{\Psi_1/U_1\}$. Consequently the fixed point unfolding principle justifies the backwards soundness of the moves determined by the constants.

A player *wins* a play of a game in the circumstances depicted in figure 1. If the configuration $(E, \langle K\rangle\Phi)$ is reached and there is no available transition then player II can not establish that E has $\langle K\rangle\Phi$. Similarly if the configuration is $(E, [K]\Phi)$ and there is no available transition then player I cannot refute that E has $[K]\Phi$. Similar comments apply to the case when the configuration is (E, Z). The other circumstances when a player is said to win a play concern repetition. If the configuration reached is (E, U) when U abbreviates a maximal fixed point formula, and this same configuration occurs earlier in the game play then player II wins. Dually if U abbreviates a least fixed point it is player I

[6] It is straightforward to reformulate the definition so that players must take turns.

Player II wins	**Player I wins**
1. The play is $(E_0, \Phi_0) \ldots (E_n, \Phi_n)$ and $\Phi_n = Z$ and $E \in \mathcal{V}(Z)$.	1′. The play is $(E_0, \Phi_0) \ldots (E_n, \Phi_n)$ and $\Phi_n = Z$ and $E \notin \mathcal{V}(Z)$.
2. The play is $(E_0, \Phi_0) \ldots (E_n, \Phi_n)$ and $\Phi_n = [K]\Psi$ and the set $\{F : E_n \xrightarrow{a} F \text{ and } a \in K\} = \emptyset$.	2′. The play is $(E_0, \Phi_0) \ldots (E_n, \Phi_n)$ and $\Phi_n = \langle K \rangle \Psi$ and the set $\{F : E_n \xrightarrow{a} F \text{ and } a \in K\} = \emptyset$.
3. The play is $(E_0, \Phi_0) \ldots (E_n, \Phi_n)$ and $\Phi_n = U$ and $U \stackrel{\text{def}}{=} \nu Z.\Phi$ and $E_i = E_n$ and $\Phi_i = \Phi_n$ for $i < n$.	3′. The play is $(E_0, \Phi_0) \ldots (E_n, \Phi_n)$ and $\Phi_n = U$ and $U \stackrel{\text{def}}{=} \mu Z.\Phi$ and $E_i = E_n$ and $\Phi_i = \Phi_n$ for $i < n$.
4. The play $(E_0, \Phi_0) \ldots (E_i, \Phi_i) \ldots$ is infinite length and there is a constant $U \stackrel{\text{def}}{=} \nu Z.\Phi$ such that for infinitely many j, $\Phi_j = U$.	4′. The play $(E_0, \Phi_0) \ldots (E_i, \Phi_i) \ldots$ is infinite length and there is a constant $U \stackrel{\text{def}}{=} \mu Z.\Phi$ such that for infinitely many j, $\Phi_j = U$.

Fig. 1. Winning conditions

that wins[7]. More generally as a play can have infinite length (but only when the initial process is infinite state) this repeat condition for winning is generalized for these plays. Player I wins if there is a least fixed point constant U which is traversed infinitely often, and player II wins if instead there is a greatest fixed point constant U which occurs infinitely often. In any infinite length play there is only one constant that occurs infinitely often, and therefore just one of the players wins the play.

Lemma 1 *If $(E_0, \Phi_0) \ldots (E_n, \Phi_n) \ldots$ is an infinite length game play then there is exactly one constant U such that for infinitely many j, $\Phi_j = U$.*

A player is said to have a *winning strategy* for a game if she is able to win any play of it. This means that she can always respond effectively to the moves her opponent makes.

Theorem 1 *$E \models_{\mathcal{V}} \Phi$ iff player II has a winning strategy for (E, Φ) under $\mathcal{V}$.*

This provides an alternative characterization of the satisfaction relation between processes and formulas. Game playing does not depend upon explicit calculation of fixed points. It is also open-ended as to knowing only part or all of the transition graph of a process. There is another feature, the possibility of more sophisticated game playing where moves may also be guided by the algebraic structure of a process expression (the raw material in [2] provides a basis for this).

Player II has a winning strategy for the game $(Clock, \mu Z.[\mathtt{tick}]Z)$. Any play has finite length as player I must choose a transition $Clock \xrightarrow{\mathtt{tick}} Cl_i$ and so

[7] These two conditions are redundant, but are included because then any finite-state process can only have finite length game plays.

must end being stuck in a configuration $(Cl_0, [\mathtt{tick}]U)$ when $U \stackrel{\text{def}}{=} \mu Z.[\mathtt{tick}]Z$. Similarly the safety and liveness (under fairness) properties of the crossing can be established using games.

Game playing justifies the tableaux proof systems for verifying temporal properties of finite and infinite-state processes as developed in [6, 20]. A successful tableau for a process E and a property Φ turns out to be a *witness* for player II's successful strategy for (E, Φ). In the case that E is a finite-state process each branch in the tableau is a winning play for player II, and all choices available to player I are contained within it. In the case of an infinite-state system the idea is essentially the same.

Game playing provides a very transparent methodology for property proving. However in the case of a finite-state process it is not very time efficient: the length of a play may be exponential in the number of fixed point subformulas. By refining the definition of game we can dramatically improve efficiency, and yet retain transparency.

5 Refinement of games

In this section we refine the definition of game play to provide a more efficient characterization of the satisfaction relation. Constants are reintroduced when the same fixed point formula is met again. This means that the previous rule for introducing constants for fixed points is divided it into two cases. Recall that we are defining the next pair in the play $(E_0, \Phi_0) \ldots (E_j, \Phi_j)$:

- if $\Phi_j = \nu Z.\Psi$ ($\mu Z.\Psi$) and player I (player II) has not previously introduced a constant $V \stackrel{\text{def}}{=} \nu Z.\Psi$ ($V \stackrel{\text{def}}{=} \mu Z.\Psi$) then player I (player II) chooses a *new* constant U and sets $U \stackrel{\text{def}}{=} \nu Z.\Psi$ ($U \stackrel{\text{def}}{=} \mu Z.\Psi$): the process E_{j+1} is E_j and Φ_{j+1} is U.
- if $\Phi_j = \nu Z.\Psi$ ($\mu Z.\Psi$) and player I (player II) has previously introduced a constant $V \stackrel{\text{def}}{=} \nu Z.\Psi$ ($V \stackrel{\text{def}}{=} \mu Z.\Psi$) then E_{j+1} is E_j and Φ_{j+1} is V.

The other rules are as before. As before a player wins a play in the circumstances $1, 1', 2$ and $2'$ of figure 1. The other conditions for winning, when there is a repeat configuration and those for infinite length plays, need to be redefined because constants are reintroduced: an infinite length play may now contain more than one constant that recurs infinitely often. A little notation:

Definition 1 The constant U is *active* in Φ iff either U occurs in Φ, or some constant V occurs in Φ with $V \stackrel{\text{def}}{=} \sigma Z.\Psi$ and U is active in $\sigma Z.\Psi$.

The constraints on how constants are introduced ensure that being active is well defined. Activity of a constant can be extended to finite or infinite length sequences of formulas, U is *active* throughout $\Phi_0, \ldots, \Phi_n \ldots$ if it is active in each Φ_i.

Lemma 2 i. *If* $(E_0, \Phi_0), \ldots, (E_n, \Phi_n)$ *is an initial part of a game play and* $\Phi_i = \Phi_n$ *when* $i < n$ *then there is a unique constant* U *which is active throughout* $\Phi_i, \ldots, \Phi_n$ *and occurs there,* $\Phi_j = U$ *for some* $j : i \leq j \leq n$.

ii. *If* $(E_0, \Phi_0), \ldots, (E_n, \Phi_n) \ldots$ *is an infinite length game play then there is a unique constant* U *which occurs infinitely often and is active throughout* $\Phi_j, \ldots, \Phi_n \ldots$ *for some* $j \geq 0$.

Lemma 2 governs the remaining winning conditions for game playing, the replacements for 3, 3′, 4 and 4′ of figure 1. A repeat configuration (E, Ψ) when Ψ is *any formula*, and not just a constant, terminates play. Who wins depends on the sequence of formulas between (and including) the identical configurations. There is exactly one constant U which is active in this cycle and which occurs within it: if it abbreviates a maximal fixed point formula then player II wins and otherwise it must abbreviate a least fixed point formula and player I wins. In any infinite length play there is a unique constant which is traversed infinitely often and which is active for all but a finite prefix: if this constant abbreviates a maximal fixed point formula player II wins and otherwise player I wins.

As before a player is said to have a winning strategy for a game if she is able to win any play of it.

Theorem 2 $E \models_{\mathcal{V}} \Phi$ *iff player II has a winning strategy for* (E, Φ) *under* $\mathcal{V}$.

It is straightforward to present tableaux proof systems for verifying temporal properties of finite and infinite-state processes, which are underpinned by these more refined games and where again a successful tableau is a witness for player II's successful strategy.

Theorem 2 offers a different perspective on similar results for the finite-state case presented in [12] whose basis is tree automata, and [4] which is grounded in alternating automata. The proof in the general case, when processes may be infinite-state, is similar to the model construction in [19]. It also follows from Theorem 2 (which utilizes approximants) that a winning strategy for a game is stationary or history free. Hence for a player it is a function from configurations she is able to move from to unique successors.

Assume that E is finite-state. The *game graph* for (E, Φ) relative to $\mathcal{V}$ is the graph representing all possible plays of (E, Φ) modulo a canonical means of choosing constants. The vertices are pairs (F, Ψ), configurations of a possible game play, and there is a directed edge between two vertices $v_1 \longrightarrow v_2$ if a player can make as her next move v_2 from v_1. Let $\mathcal{G}(E, \Phi)$ be the game graph for (E, Φ), and let $|\mathcal{G}(E, \Phi)|$ be its vertex size. It follows that $|\mathcal{G}(E, \Phi)| \leq |E| * |\Phi|$ where $|E|$ is the number of processes in the transition graph for E, and $|\Phi|$ is the size of this formula. This means that any play of (E, Φ) has length at most $1 + (|E| * |\Phi|)$. The proof that model checking belongs to NP $\cap$ co-NP follows from the observation that given a strategy for player II or player I it is straightforward to check in polynomial time whether or not it is successful. (See [10] for a proof which has its roots in [9] using tree automata.) For the alternation free fragment game graphs obey the weak alternating automaton condition for partitioning vertices [18], and hence model checking can be determined in polynomial time.

(See [4] which directly uses alternating automata.)

We can easily ensure that game playing must proceed to infinity by adding extra moves when a player is stuck (and removing the redundant repeat termination condition). The resulting game graph is then an alternating automaton: the and-vertices are the configurations from which player I must proceed and the or-vertices are those from which player II moves, and the acceptance condition is given in terms of active constants.

Alternatively a game graph can be directly translated into a formula of boolean fixed point logic, defined as follows:

$$\Phi ::= Z \mid \mathtt{tt} \mid \mathtt{ff} \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \nu Z.\Phi \mid \mu Z.\Phi$$

Satisfiability (or really truth) checking of closed formulas of this logic is therefore also in NP $\cap$ co-NP. Various authors have, in effect, translated finite-state model checking into this logic, with a preference for a syntax utilizing equations [1, 14, 7]. One can also model check directly using approximants, where a careful utilization of monotonicity provides reasonable exponential algorithms [11, 15].

An important open question is whether model checking modal mu-calculus formulas can be done in polynomial time (with respect to the size of a game graph). One direction for research is to provide a finer analysis of successful strategies, and to be able to describe optimizations of them. New insights may come from the relationship between the games developed here and other graph games where there are such descriptions.

6 Graph games

The model checking game of the previous section can be abstracted into the following graph game. A game is a graph with vertices $\{1, \ldots, n\}$ where each vertex i has two directed edges $i \longrightarrow j_1$ and $i \longrightarrow j_2$, and which obeys the following condition: if $i \longrightarrow j$ and $j \leq i$ then there is a path $j \longrightarrow j_1 \longrightarrow \ldots \longrightarrow j_n = i$ where $j \leq j_1 \leq \ldots \leq j_n$. Each vertex is labelled I or II. A play is an infinite path through the graph starting at vertex 1, and player I moves from vertices labelled I and player II from vertices labelled II. The winner of a play is determined by the label of the *least* vertex i which is traversed infinitely often: if i is labelled I then player I wins, and if II then player II wins. A player wins the game if she is able to win any play. A winning strategy is again stationary.

Simple stochastic games [8] are graph games where the vertices are labelled I, II or A (average), and where there are two special vertices I-sink and II-sink (which have no outgoing edges). As above each I, II (and A) vertex has two outgoing edges. At an average vertex during a game play a coin is tossed to determine which of the two edges is traversed each having probability $\frac{1}{2}$. More generally one can assume that the two edges are labelled with probabilities of the form $\frac{p}{q}$ where $0 \leq p \leq q \leq 2^m$ for some m, as long as their sum is 1. A game play ends when a sink vertex is reached: player II wins if it is the II-sink, and player I otherwise. The decision question is whether the probability that player II wins is greater than $\frac{1}{2}$. It is not known whether this problem can be solved

in polynomial time. In [16] a "subexponential" $(2^{O(\sqrt{n})})$ algorithm is presented, which works by refining optimal strategies. A polynomial time algorithm for simple stochastic games would imply that extending space bounded alternating Turing machines with randomness does not increase the class of languages that they accept.

Mark Jerrum noted that there is a reduction from the graph game to the simple stochastic game. The idea is to add the two sink vertices, and an average vertex $i1$ for each vertex i for which there is an edge $j \longrightarrow i$ with $j \geq i$. Each such edge $j \longrightarrow i$ when $j \geq i$ is changed to $j \longrightarrow i1$. And the vertex $i1$ has an edge to i, and to I-sink if i is labelled I or to II-sink otherwise. With suitable rational probabilities on the edges, player II has a winning strategy for the graph game iff she has one for the simple stochastic game. Another relevant graph game is the mean payoff game for which there is also a reduction from the model checking game.

Acknowledgement: I would like to thank Mark Jerrum for numerous discussions about model checking and games.

References

1. Andersen, H. (1994). Model checking and boolean graphs. *Theoretical Comp. Science*, **126**, 3-30.
2. Andersen, H., Stirling, C., and Winskel, G. (1994) A compositional proof system for the modal mu-calculus. *Procs LICS.*
3. Arnold, A., and Niwinski, D. (1992). Fixed point characterization of weak monadic logic definable sets of trees. In *Tree Automata and Languages*, ed. M. Nivat and A. Podelski, Elsevier, 159-188.
4. Bernholtz, O., Vardi, M. and Wolper, P. (1994). An automata-theoretic approach to branching-time model checking. *Procs. CAV 94.*
5. Bradfield, J. and Stirling, C. (1990). Verifying temporal properties of processes. *Lect. Notes in Comput. Science*, **458**, 115-125.
6. Bradfield, J. and Stirling, C. (1992). Local model checking for infinite state spaces. *Theoret. Comput. Science*, **96**, 157-174.
7. Cleaveland, R. and Steffen, B. (1992). A linear-time model checking algorithm for the alternation-free modal mu-calculus. *Lect. Notes in Comp Science*, **575.**
8. Condon, A. (1992). The complexity of stochastic games. *Inf. and Comp.*, **96**, 203-224.
9. Emerson, E. (1985). Automata, tableaux, and temporal logics. *Lect. Notes in Comput. Science*, **193**, 79-87.
10. Emerson, E., and Jutla, C. (1988). The complexity of tree automata and logics of programs. Extended version from FOCS '88.
11. Emerson, E, and Lei, C. (1986). Efficient model checking in fragments of the propositional mu-calculus. In *Proc. 1st IEEE Symp. on Logic in Comput. Science*, 267-278.
12. Emerson, E., Jutla, C., and Sistla, A. (1993). On model checking for fragments of μ-calculus. *Lect. Notes in Comput. Sci.*, **697**, 385-396.
13. Kozen, D. (1983). Results on the propositional mu-calculus. *Theoret. Comput. Sci* **27**, 333-354.

14. Larsen, K. (1992). Efficient local correctness checking. *Lect. Notes in Comput. Sci.*, **663**, 385-396.
15. Long, D., Browne, A., Clarke, E., Jha, S., and Marrero, W. (1994) An improved algorithm for the evaluation of fixpoint expressions. *Procs. CAV 94.*
16. Ludwig, W. (1995). A subexponential randomized algorithm for the simple stochastic game problem. *Inf. and Comp*, **117**, 151-155.
17. Milner, R. (1989). *Communication and Concurrency.* Prentice Hall.
18. Muller, D., Saoudi, A. and Schupp, P. (1986). Alternating automata, the weak monadic theory of the tree and its complexity. *Lect. Notes in Comput. Sci.*, **225**, 275-283.
19. Streett, R. and Emerson, E. (1989). An automata theoretic decision procedure for the propositional mu-calculus. *Inf. and Comp.*, **81**, 249-264.
20. Stirling, C. and Walker, D. (1991). Local model checking in the modal mu-calculus. *Theoret. Comput. Science*, **89**, 161-177.