Bisimulation, Modal Logic and Model Checking Games

Colin Stirling, *Division of Informatics, University of Edinburgh, Edinburgh EH9 3JZ, cps@dcs.ed.ac.uk*

Abstract

We give a very brief introduction to how concurrent systems can be modelled within process calculi, as terms of an algebraic language whose behaviours are described using transitions. Reasoning has centred on two kinds of questions. One is relationships between descriptions of concurrent systems. The other is appropriate logics for describing crucial properties of concurrent systems. Bisimulation equivalence is briefly described. It can also be characterised in terms of modal logic (Hennessy-Milner logic). However as a logic it is not very expressive. So we also describe modal mu-calculus which is a very expressive temporal logic. In the main part of the paper we show that property checking can be understood in terms of game playing. In the finite state case, games underpin efficient model checking algorithms. The games are also definable independently of property checking as graph games which can be reduced to other combinatorial games.

Keywords: Bisimulation, modal logic, fixed point, games

1 Introduction

Concurrency theory is concerned with formal notations and techniques for modelling and reasoning about concurrent systems such as protocols and safety critical control systems. In Section 2 we give a very brief introduction to how concurrent systems can be modelled within process calculi, as terms of an algebraic language. Their behaviours are described using transitions. Reasoning has centred on two kinds of questions. One is relationships between descriptions of concurrent systems. For instance, when are two descriptions equivalent? The second is appropriate logics for describing crucial properties of concurrent systems. Temporal logics have been found to be very useful. In Section 3 bisimulation equivalence is described. It can also be characterised in terms of modal logic (Hennessy-Milner logic). However as a logic it is not very expressive. So we also describe modal mu-calculus, modal logic with fixed points, and show that it is a very expressive temporal logic for describing properties of processes. However it is also very important to be able to verify that processes have temporal properties. This is the topic of Section 4. First we show that property checking can be understood in terms of game playing. In the finite state case, games underpin efficient model checking algorithms. Second the games are definable independently of property checking as graph games which can be reduced to other combinatorial games (and in particular to simple stochastic games). An important open question is whether finite state property checking of modal mu-calculus properties can be done in polynomial time.

Techniques (such as model checking games) discussed in this paper have been implemented in the Edinburgh Concurrency Workbench, which is a freely available tool for manipulating and analysing concurrent systems, see http/www.dcs.ed.ac.uk/home/cwb.

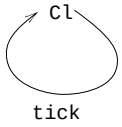


FIG. 1. The transition system for Cl

2 Process Calculi

Process calculi (as developed by Milner, Hoare and others) model concurrent systems as terms of an algebraic language comprising a few basic operators. Transitions of the form $E \xrightarrow{a} F$, that process E may become F by performing the action a, feature prominently. Structured rules guide their derivation, as the transitions of a compound process are determined by those of its components. Families of transitions can be arranged as labelled graphs, concrete summaries of the behaviour of processes. Here we give a very brief introduction to some of the operators of CCS (Calculus of Communicating Systems [27]).

A simple process is a clock that perpetually ticks:

$$\texttt{Cl} \stackrel{\text{def}}{=} \texttt{tick}.\texttt{Cl}$$

Names of actions, such as tick, are in lower case whereas names of processes, such as Cl, have an initial capital letter. The behaviour of Cl is very elementary, as it can only perform the action tick and in so doing becomes Cl again. This follows from the rules for transitions. First is the axiom for the prefix operator when a is an action and E a process: $a.E \xrightarrow{a} E$. Next is the transition rule for the operator $\stackrel{\text{def}}{=}$ which associates a process name P with a process expression E.

$$\text{if } E \xrightarrow{a} F \text{ and } P \stackrel{\text{def}}{=} E \text{ then } P \xrightarrow{a} F$$

¿From these two rules it follows that $\texttt{Cl} \xrightarrow{\texttt{tick}} \texttt{Cl}$. The behaviour of Cl is visualised in Figure 1. Ingredients of this graph (called a *labelled transition system*) are process expressions and binary transition relations between them. Each vertex is a process expression, and one of them is the initial vertex Cl. Each transition of a vertex which is derivable is depicted.

The binary choice operator + (which has wider scope than the . operator) has two transition rules:

$$\begin{array}{l} \text{if } \; E_1 \xrightarrow{a} F \; \text{ then } \; E_1 + E_2 \xrightarrow{a} F \\ \text{if } \; E_2 \xrightarrow{a} F \; \text{ then } \; E_1 + E_2 \xrightarrow{a} F \end{array}$$

The central feature of process calculi is modelling concurrent interaction. A prevalent approach is to appeal to handshake communication as primitive. At any one time only two processes may communicate. The resultant communication is a *completed* internal action τ. Each incomplete action a has a partner $\overline{a}$, its co-action. Moreover the action $\overline{\overline{a}}$ is a which means that a is also the co-action of $\overline{a}$. The joint activity of a and $\overline{a}$ is the communication.

Concurrent composition of E and F is expressed as the process $E \mid F$. Transition rules for $\mid$ are:

$$\begin{array}{l}\text{if } E \xrightarrow{a} E' \text{ and } F \xrightarrow{\overline{a}} F' \text{ then } E \mid F \xrightarrow{\tau} E' \mid F' \\ \text{if } E \xrightarrow{a} E' \text{ then } E \mid F \xrightarrow{a} E' \mid F \\ \text{if } F \xrightarrow{a} F' \text{ then } E \mid F \xrightarrow{a} E \mid F'\end{array}$$

The first of these conveys communication.

There is also an abstraction or encapsulation operator $\backslash J$ where J ranges over families of incomplete actions (thereby excluding τ). Let $\overline{J}$ be the set $\{\overline{a} : a \in J\}$.

$$\text{if } E \xrightarrow{a} F \text{ and } a \notin J \cup \overline{J} \text{ then } E\backslash J \xrightarrow{a} F\backslash J$$

The behaviour of $E\backslash J$ is part of that of E. In $(E \mid F)\backslash J$ the presence of $\backslash J$ prevents E from ever doing a J transition except in the context of a communication with F. In this way communication between E and F can be enforced.

The mesh of abstraction and concurrency is revealed in the following finite-state example of a level crossing from [7] consisting of three components.

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

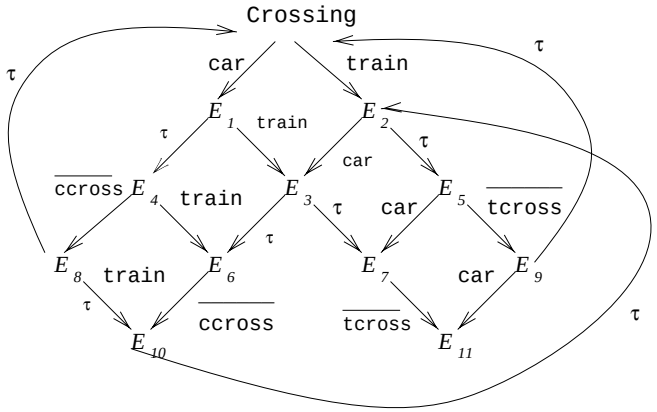
The relation $\equiv$ when $P \equiv F$ means that P abbreviates F. The actions `car` and `train` represent the approach of a car and a train, `up` is the gates opening for the car, $\overline{\texttt{ccross}}$ is the car crossing, `down` closes the gates, `green` is the receipt of a green signal by the train, $\overline{\texttt{tcross}}$ is the train crossing, and `red` automatically sets the light red. Its transition graph is depicted in Figure 2.

The crossing offers a flavour of how systems are modelled in CCS. It is usual to abstract from silent activity. One defines observable transitions $E \stackrel{a}{\Longrightarrow} F$ for $a \neq \tau$ iff $E \xrightarrow{\tau^*} F_1 \xrightarrow{a} F_2 \xrightarrow{\tau^*} F$. This means that there are two transition systems associated with any process expression. There is a large literature on modelling systems using process calculi, see [27] for an elegant introduction. There are many extensions of the model which encompass real time, probabilities and locations.

Transition graphs, such as in Figure 2, are labelled Kripke structures: the accessibility relations are indexed by actions and the possible worlds are process expressions.

3 Equivalences, Modal and Temporal Logics

An important issue is when two process descriptions count as equivalent. There is a variety of equivalences in the literature. In the case of CCS, the definition of equivalence begins with the simple idea that an observer can repeatedly interact with a process by choosing an available transition from it. Equivalence of processes is then defined in terms of the ability for these observers to match their selections so that they can proceed with further corresponding choices. This equivalence is defined in terms of bisimulation relations which capture precisely what it is for observers to match their selections. However we proceed with the well known alternative exposition using



$$
\begin{array}{lcl}
K & = & \{\mathtt{green}, \mathtt{red}, \mathtt{up}, \mathtt{down}\} \\
 & & \\
E_1 & \equiv & (\mathtt{up}.\overline{\mathtt{ccross}}.\overline{\mathtt{down}}.\mathtt{Road} \mid \mathtt{Rail} \mid \mathtt{Signal})\backslash K \\
E_2 & \equiv & (\mathtt{Road} \mid \mathtt{green}.\overline{\mathtt{tcross}}.\overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{Signal})\backslash K \\
E_3 & \equiv & (\mathtt{up}.\overline{\mathtt{ccross}}.\overline{\mathtt{down}}.\mathtt{Road} \mid \mathtt{green}.\overline{\mathtt{tcross}}.\overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{Signal})\backslash K \\
E_4 & \equiv & (\overline{\mathtt{ccross}}.\overline{\mathtt{down}}.\mathtt{Road} \mid \mathtt{Rail} \mid \mathtt{down}.\mathtt{Signal})\backslash K \\
E_5 & \equiv & (\mathtt{Road} \mid \overline{\mathtt{tcross}}.\overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{red}.\mathtt{Signal})\backslash K \\
E_6 & \equiv & (\overline{\mathtt{ccross}}.\overline{\mathtt{down}}.\mathtt{Road} \mid \mathtt{green}.\overline{\mathtt{tcross}}.\overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{down}.\mathtt{Signal})\backslash K \\
E_7 & \equiv & (\mathtt{up}.\overline{\mathtt{ccross}}.\overline{\mathtt{down}}.\mathtt{Road} \mid \overline{\mathtt{tcross}}.\overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{red}.\mathtt{Signal})\backslash K \\
E_8 & \equiv & (\overline{\mathtt{down}}.\mathtt{Road} \mid \mathtt{Rail} \mid \mathtt{down}.\mathtt{Signal})\backslash K \\
E_9 & \equiv & (\mathtt{Road} \mid \overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{red}.\mathtt{Signal})\backslash K \\
E_{10} & \equiv & (\overline{\mathtt{down}}.\mathtt{Road} \mid \mathtt{green}.\overline{\mathtt{tcross}}.\overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{down}.\mathtt{Signal})\backslash K \\
E_{11} & \equiv & (\mathtt{up}.\overline{\mathtt{ccross}}.\overline{\mathtt{down}}.\mathtt{Road} \mid \overline{\mathtt{red}}.\mathtt{Rail} \mid \mathtt{red}.\mathtt{Signal})\backslash K
\end{array}
$$

FIG. 2. The transition system for `Crossing`

Ehrenfeucht-Fraïssé games. Another characterisation of bisimulation equivalence uses modal logic which we also describe. However as a logic it is not very expressive. So we also describe modal mu-calculus, modal logic with fixed points, and show that it is a very expressive temporal logic for describing properties of processes. We also define second-order propositional modal logic to contrast fixed points and second-order quantifiers.

3.1 Interactive games and bisimulations

The equivalence game $\mathcal{G}(E_0, F_0)$ on the pair of processes E_0 and F_0 is played by two participants, players I and II, who are the observers who make choices of transitions. A *play* of the game $\mathcal{G}(E_0, F_0)$ is a finite or infinite length sequence of the form $(E_0, F_0) \ldots (E_i, F_i) \ldots$. Player I attempts to show that the initial processes are different whereas player II wishes to establish that they are equivalent. Suppose an initial part of a play is $(E_0, F_0) \ldots (E_j, F_j)$. The next pair (E_{j+1}, F_{j+1}) is determined by one of the following two moves:

- Player I chooses a transition $E_j \stackrel{a}{\longrightarrow} E_{j+1}$ and then player II chooses a transition with the same label $F_j \stackrel{a}{\longrightarrow} F_{j+1}$,
- Player I chooses a transition $F_j \stackrel{a}{\longrightarrow} F_{j+1}$ and then player II chooses a transition with the same label $E_j \stackrel{a}{\longrightarrow} E_{j+1}$.

The play continues with further moves. Player I always chooses first, and then player II, with full knowledge of player I's selection, must choose a corresponding transition from the other process. (Here we build the games from the transitions $\stackrel{a}{\longrightarrow}$: instead we could build them from the observable transitions $\stackrel{a}{\Longrightarrow}$.)

A play of a game continues until one of the players wins. If two processes have different initial capabilities then they are clearly distinguishable. Consequently any position (E_n, F_n) where one of these processes is able to perform an initial action which the other can not counts as a win for player I. Let us call such positions, I-wins. A play is won by player I if the play reaches a I-win position. Any play that fails to reach such a position counts as a win for player II. Consequently player II wins if the play is infinite, or if the play reaches the position (E_n, F_n) and neither process has an available transition.

Different plays of a game can have different winners. Nevertheless for each game one of the players is able to win any play irrespective of what moves her opponent makes. To make this precise, the notion of strategy is essential. A strategy for a player is a family of rules which tell the player how to move. However it turns out that we only need to consider *history-free* strategies whose rules do not depend on what happened previously in the play. For player I a rule is therefore of the form

$$\text{at} \quad \text{position } (E, F) \text{ choose} \quad \text{transition } t$$

where t is $E \stackrel{a}{\longrightarrow} E'$ or $F \stackrel{a}{\longrightarrow} F'$. A rule for player II is

$$\text{at} \quad \text{position } (E, F) \text{ when} \quad \text{player I has chosen } t \text{ choose } t'$$

where t is either $E \stackrel{a}{\longrightarrow} E'$ or $F \stackrel{a}{\longrightarrow} F'$ and t' is a corresponding transition of the other process. A player uses the strategy π in a play if all her moves obey the rules in π. The strategy π is a *winning strategy* if the player wins every play in which she uses π.

Example 1 Consider the two similar vending machines

$$\begin{aligned} \mathtt{U} &\stackrel{\text{def}}{=} \mathtt{1p.(1p.tea.U + 1p.coffee.U)} \\ \mathtt{V} &\stackrel{\text{def}}{=} \mathtt{1p.1p.tea.V + 1p.1p.coffee.V} \end{aligned}$$

Player I has a winning strategy for the game $\mathcal{G}(\mathtt{U}, \mathtt{V})$: if the position is $(\mathtt{U}, \mathtt{V})$ then choose $\mathtt{V} \stackrel{\mathtt{1p}}{\longrightarrow} \mathtt{1p.tea.V}$, and if $(\mathtt{1p.tea.U + 1p.coffee.U}, \mathtt{1p.tea.V})$ choose $\mathtt{1p.tea.U + 1p.coffee.U} \stackrel{\mathtt{1p}}{\longrightarrow} \mathtt{coffee.U}$. □

Proposition 1 *For any game $\mathcal{G}(E, F)$ either player I or player II has a history-free winning strategy.*

In Section 4 we describe how this result can be proved. If player II has a winning strategy for $\mathcal{G}(E, F)$ then E is *game equivalent* to process F. Game equivalence is

indeed an equivalence: for instance, Player II's winning strategy for $\mathcal{G}(E, E)$ is the copy-cat strategy (always choose the same transition as player I).

When two processes E and F are game equivalent, player II can always match player I's moves: if $E \xrightarrow{a} E'$ (or $F \xrightarrow{a} F'$) then there is a corresponding transition $F \xrightarrow{a} F'$ (or $E \xrightarrow{a} E'$) and E' and F' are also game equivalent. This is precisely the criterion for being a *bisimulation* relation. Bisimulations were introduced by Park [30] as a small refinement of the equivalence defined by Hennessy and Milner in [16].

Definition 1 A binary relation $\mathcal{R}$ between processes is a *bisimulation* just in case whenever $(E, F) \in \mathcal{R}$ and $a \in \mathcal{A}$,

1. if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $(E', F') \in \mathcal{R}$ and
2. if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' such that $(E', F') \in \mathcal{R}$.

Simple examples of bisimulations are the identity relation and the empty relation. Two processes E and F are *bisimulation equivalent* (or *bisimilar*), written $E \sim F$, if there is a bisimulation relation $\mathcal{R}$ with $(E, F) \in \mathcal{R}$. Clearly, bisimulation and game equivalence coincide.

Parallel composition is both commutative and associative with respect to bisimulation equivalence (as is $+$): this permits us to drop bracketing in the case of a process description with multiple parallel components (as we did for `Crossing`). Bisimulation equivalence is also a *congruence* with respect to the process combinators of CCS and other process calculi.

To show that two processes are bisimilar it is sufficient to exhibit a bisimulation relation which contains them, thereby giving a very straightforward proof technique for bisimilarity.

Example 2 The following two processes $\mathtt{Cnt}$ and $\mathtt{Ct}'_0$ are bisimilar (where $\mathtt{0}$ is the process which has no available transitions)

$$\begin{array}{lcl} \mathtt{Cnt} & \stackrel{\text{def}}{=} & \mathtt{up}.(\mathtt{Cnt} \mid \mathtt{down}.\mathtt{0}) \\ \mathtt{Ct}'_0 & \stackrel{\text{def}}{=} & \mathtt{up}.\mathtt{Ct}'_1 \\ \mathtt{Ct}'_{i+1} & \stackrel{\text{def}}{=} & \mathtt{up}.\mathtt{Ct}'_{i+2} + \mathtt{down}.\mathtt{Ct}'_i \end{array}$$

Let $\mathcal{C}_i$ be the following families of processes for $i \geq 0$ (when brackets are dropped between parallel components):

$$\begin{array}{lcl} \mathcal{C}_0 & = & \{\mathtt{Cnt} \mid \mathtt{0}^j \ : \ j \geq 0\} \\ \mathcal{C}_{i+1} & = & \{E \mid \mathtt{0}^j \mid \mathtt{down}.\mathtt{0} \mid \mathtt{0}^k \ : \ E \in \mathcal{C}_i \text{ and } j \geq 0 \text{ and } k \geq 0\} \end{array}$$

where $F \mid \mathtt{0}^0 = F$ and $F \mid \mathtt{0}^{i+1} = F \mid \mathtt{0}^i \mid \mathtt{0}$. The relation $\{(E, \mathtt{Ct}'_i) \ : \ i \geq 0 \text{ and } E \in \mathcal{C}_i\}$ is a bisimulation which contains $(\mathtt{Cnt}, \mathtt{Ct}'_0)$. □

In the case of finite-state systems there are fast algorithms for deciding bisimulation equivalence which have been incorporated in tools for analysing concurrent systems.

Bisimulation equivalence is a very fine equivalence between processes, reflecting the fact that in the presence of concurrency a more intensional description of process behaviour is needed than for instance sets of traces or words. For full CCS the question whether two processes are bisimilar is undecidable. Turing machines can

be modelled in CCS. Let $\mathtt{TM}_n$ be this coding of the n-th Turing machine when all incomplete actions are abstracted. The undecidable Turing machine halting problem is equivalent to whether or not $\mathtt{TM}_n \sim \mathtt{Div}$ where $\mathtt{Div} \stackrel{\text{def}}{=} \tau.\mathtt{Div}$. However an interesting question is for which subclasses of processes it is decidable. Clearly this is the case for finite-state processes. Surprisingly it is also decidable for families of infinite-state processes including context-free processes, pushdown processes and basic parallel processes [10, 34, 9].

3.2 Modal logic and bisimulation equivalence

Another way of understanding bisimulation equivalence uses multimodal logic. Let M be the following family of modal formulas where K ranges over subsets of $\mathcal{A}$:

$$\Phi ::= \mathtt{tt} \mid \mathtt{ff} \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [K]\Phi \mid \langle K\rangle\Phi$$

The inductive stipulation below defines when a process E has a modal property Φ, written $E \models \Phi$. If E fails to satisfy Φ then this is written $E \not\models \Phi$.

$E \models \mathtt{tt}$		$E \not\models \mathtt{ff}$
$E \models \Phi \wedge \Psi$	iff	$E \models \Phi$ and $E \models \Psi$
$E \models \Phi \vee \Psi$	iff	$E \models \Phi$ or $E \models \Psi$
$E \models [K]\Phi$	iff	$\forall F.\, \forall a \in K.$ if $E \stackrel{a}{\longrightarrow} F$ then $F \models \Phi$
$E \models \langle K\rangle\Phi$	iff	$\exists F.\, \exists a \in K.\, E \stackrel{a}{\longrightarrow} F$ and $F \models \Phi$

This modal logic slightly generalises Hennessy-Milner logic [16] as sets of actions instead of single actions appear in the modalities.

Bisimilar processes have the same modal properties. Let $E \equiv_M F$ just in case E and F have the same modal properties.

Proposition 2 *If* $E \sim F$ *then* $E \equiv_M F$.

The converse of Proposition 2 holds for a restricted set of processes. A process E is immediately image-finite if for each $a \in \mathcal{A}$ the set $\{F : E \stackrel{a}{\longrightarrow} F\}$ is finite. And E is *image-finite* if every member of $\{F : \exists w \in \mathcal{A}^*.\, E \stackrel{w}{\longrightarrow} F\}$ is immediately image-finite (where $E \stackrel{w}{\longrightarrow} F$ is defined in the obvious way).

Proposition 3 *If* E *and* F *are image-finite and* $E \equiv_M F$ *then* $E \sim F$.

These two results are known as the modal *characterization* of bisimulation equivalence, due to Hennessy and Milner [16]. (There is also an unrestricted characterisation result for infinitary modal logic.)

Bisimulations were also discovered independently and earlier in the model theory of modal logic as zig-zag relations, see [3]: in this context bisimilar states also have to preserve a family of atomic sentences (the atomic formulas of modal logic), and so Definition 1 above includes an extra clause. One of the many interesting results from this line of research concerns bisimulation closed properties: a property Φ is bisimulation closed if whenever E has Φ and $E \sim F$ then F has Φ. Van Benthem showed that a first-order formula with one free variable (over transition systems) is bisimulation closed iff it is equivalent to a modal formula [3].

3.3 Temporal properties and modal mu-calculus

The modal logic M of the previous section is able to express local capabilities and necessities of processes (such as that `tick` is a possible next action or that it must happen next). However it cannot express enduring capabilities (such as `tick` is always possible) or long term inevitabilities (such as `tick` must eventually happen). These features, especially in the guise of safety or liveness properties, have been found to be very useful when analysing the behaviour of concurrent systems. Another abstraction from behaviour is a run of a process which is a finite or infinite length sequence of transitions. Runs provide a basis for understanding longer term capabilities. Logics where properties are primarily ascribed to runs of systems are called temporal logics. An alternative foundation for temporal logic is to view these enduring features as extremal solutions to recursive modal equations.

Modal mu-calculus, modal logic with extremal fixed points, introduced by Kozen [21], is a very expressive propositional temporal logic. Formulas of the logic, μM, given in positive form are defined as follows

$$\Phi ::= \mathtt{tt} \mid \mathtt{ff} \mid Z \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [K]\Phi \mid \langle K\rangle\Phi \mid \nu Z.\,\Phi \mid \mu Z.\,\Phi$$

where Z ranges over a family of propositional variables, and K over subsets[1] of $\mathcal{A}$. The binder νZ is the greatest fixed point operator whereas μZ is the least fixed point operator.

When E is a process let $\mathcal{P}(E)$ be the smallest transition closed set containing E: that is, if $F \in \mathcal{P}(E)$ and $F \xrightarrow{a} F'$ then $F' \in \mathcal{P}(E)$. Let $\mathcal{P}$ range over (non-empty) transition closed sets. We extend the semantics of modal logic of the previous section to encompass fixed points. Because of free variables we employ valuations $\mathcal{V}$ which assign 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 Z when $\mathcal{V}'(Z) = \mathcal{E}$. The inductive definition of satisfaction stipulates when a process E has the property Φ relative to $\mathcal{V}$, written $E \models_{\mathcal{V}} \Phi$, and the semantic clauses for the modal fragment are as before (except for the presence of $\mathcal{V}$).

$$\begin{array}{lll} E \models_{\mathcal{V}} Z & \text{iff} & E \in \mathcal{V}(Z) \\ E \models_{\mathcal{V}} \nu Z.\,\Phi & \text{iff} & \exists \mathcal{E} \subseteq \mathcal{P}(E).\, E \in \mathcal{E} \text{ and } \forall F \in \mathcal{E}.\, F \models_{\mathcal{V}[\mathcal{E}/Z]} \Phi \\ E \models_{\mathcal{V}} \mu Z.\,\Phi & \text{iff} & \forall \mathcal{E} \subseteq \mathcal{P}(E).\, \text{if } E \notin \mathcal{E} \text{ then } \exists F \in \mathcal{P}(E).\, F \notin \mathcal{E} \text{ and } F \models_{\mathcal{V}[\mathcal{E}/Z]} \Phi \end{array}$$

The stipulations for the fixed points follow directly from the Tarski-Knaster theorem, as a greatest fixed point is the union of all postfixed points and a least fixed point is the intersection of all prefixed points[2].

One consequence is that $E \models_{\mathcal{V}} \sigma Z.\,\Phi$ iff E has the property expressed by the *unfolding* of the fixed point, $E \models_{\mathcal{V}} \Phi\{\sigma Z.\Phi/Z\}$ (where $\Phi\{\Psi/Z\}$ is the substitution of Ψ for free occurrences of Z in Φ and σ is either μ or ν).

When Φ is a closed formula (without free variables) we often abbreviate $E \models_{\mathcal{V}} \Phi$ to $E \models \Phi$. An important feature of modal mu-calculus is that it has the *finite model property* [36]: if $E \models \Phi$ then $F \models \Phi$ for some finite-state process F

[1] μM is a slight generalisation of Kozen's logic as sets of actions instead of single actions appear in modalities.

[2] The clause above for the least fixed point is a slightly simplified (but equivalent) version of: $E \models_{\mathcal{V}} \mu Z.\,\Phi$ iff $\forall \mathcal{E} \subseteq \mathcal{P}(E)$. if $(\forall F \in \mathcal{P}(E).\, F \models_{\mathcal{V}[\mathcal{E}/Z]} \Phi$ implies $F \in \mathcal{E})$ then $E \in \mathcal{E}$.

Modal mu-calculus is a very powerful temporal logic which permits expression of a very rich class of properties. Informally a safety property states that some bad feature is always precluded. Safety can either be ascribed to *states*, that bad states can never be reached, or to *actions*, that bad actions never happen. In the former case if the formula Φ captures the complement of those bad states then $\nu Z.\, \Phi \wedge [-]Z$[3] expresses safety.

Example 3 The safety property for `Crossing` of Section 2 is that it is never possible to reach a state where a train and a car are both able to cross: the formula $(\langle \overline{\texttt{tcross}} \rangle \texttt{tt} \wedge \langle \overline{\texttt{ccross}} \rangle \texttt{tt})$ captures the bad states. Therefore the required safety property is $\nu Z.([\overline{\texttt{tcross}}]\texttt{ff} \vee [\overline{\texttt{ccross}}]\texttt{ff}) \wedge [-]Z$. □

Safety can also be ascribed to actions, that no action in K ever happens, which is expressed by the formula $\nu Z.\, [K]\texttt{ff} \wedge [-]Z$.

A liveness property states that some good feature is eventually fulfilled. Again it can either be ascribed to states, that a good state is eventually reached, or to actions, that a good action eventually happens. If Φ captures the good states then $\mu Z.\, \Phi \vee (\langle - \rangle \texttt{tt} \wedge [-]Z)$ expresses liveness with respect to state. In contrast that eventually some action in K happens is expressed by the formula $\mu Z.\, \langle - \rangle \texttt{tt} \wedge [-K]Z$.

Liveness and safety may relate to subsets of runs. For instance they may be triggered by particular actions or states. A simple case is that if action a ever happens then eventually b happens, so any run with an a action must contain a later b action. More intricate is the expression of liveness properties under fairness.

Example 4 A desirable liveness property for `Crossing` that whenever a car approaches the crossing eventually it crosses:

$$\nu Z.\, [\texttt{car}](\mu Y.\, \langle - \rangle \texttt{tt} \wedge [-\overline{\texttt{ccross}}]Y) \wedge [-]Z$$

However this only holds if we assume that the signal is fair. Let Q and R be variables and $\mathcal{V}$ a valuation such that Q is true when the crossing is in any state where `Rail` has the form $\texttt{green}.\overline{\texttt{tcross}}.\overline{\texttt{red}}.\texttt{Rail}$ and R holds when it is in any state where `Road` has the form $\texttt{up}.\overline{\texttt{ccross}}.\overline{\texttt{down}}.\texttt{Road}$. The liveness property is: for any run if Q is false infinitely often and R is also false infinitely often then whenever a car approaches eventually it crosses. This is expressed by the open formula $\nu Y.\, [\texttt{car}](\Phi_1 \wedge [-]Y)$ relative to $\mathcal{V}$ when Φ_1 is $\mu X.\nu Y_1.(Q \vee [-\overline{\texttt{ccross}}](\nu Y_2.(R \vee X) \wedge [-\overline{\texttt{ccross}}]Y_2)) \wedge [-\overline{\texttt{ccross}}]Y_1$. □

Another class of properties is until properties of the form Φ remains true until Ψ becomes true, or K actions happen until a J action occurs (or a mixture of state and action). Again they can be viewed as holding of all runs, or some runs, or of a particular family of runs which obey a condition. Cyclic properties can also be described in the logic. A simple example is that each even action is `tock`, $\nu Z.\, [-]([-\texttt{tock}]\texttt{ff} \wedge [-]Z)$. These properties can also be weakened to some family of runs. Cyclic properties that allow other actions to intervene within a cycle can also be expressed. Another class of properties is given by counting. An instance is that in each run there are exactly two a actions, given by:

[3] To reduce the number of brackets in modalities we write $a_1, \ldots, a_n$ instead of $\{a_1, \ldots, a_n\}$ inside modalities, and we let $-K$ abbreviate $\mathcal{A} - K$, and we use $-a_1, \ldots, a_n$ for $-\{a_1, \ldots, a_n\}$. Consequently $-$ abbreviates the set $-\emptyset$.

$$\mu X.\,[a](\mu Y.\,[a](\nu Z.\,[a]\mathtt{ff} \wedge [-]Z) \wedge \langle -\rangle \mathtt{tt} \wedge [-a]Y) \wedge \langle -\rangle \mathtt{tt} \wedge [-a]X$$

Another example is that in each run a can only happen finitely often, $\mu X.\,\nu Y.\,[a]X \wedge [-a]Y$. However there are also many counting properties that are not expressible in the logic. A notable case is the following (context-free) property of a buffer: the number of `out` actions never exceeds the number of `in` actions.

As shown by Janin and Walukiewicz [18] modal mu-calculus provides a natural generalisation of van Benthem's result above: a formula of monadic second-order logic with one free variable (over transition systems) is bisimulation closed iff it is equivalent to a modal mu-calculus formula. Moreover, with respect to infinite binary trees modal mu-calculus is as expressive as finite-state tree automata, and hence is as powerful as the monadic second-order theory of 2 successors [13]. 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.

One way to define sublogics is in terms of essential fixed point alternation depth when there is feedback between the different kinds of fixed points. For instance, the initial liveness property of Example 4 does not contain essential alternation as the subformula $\mu Y.\,\langle -\rangle \mathtt{tt} \wedge [\overline{\mathtt{ccross}}]Y$ does not contain Z free. However, the full blown liveness property does as within Φ_1 the least fixed point variable X appears within the scope of νY_1. An important sublogic CTL (Computation Tree Logic due to Clarke and Emerson) is contained within the *alternation free* fragment. This is the sublogic when the following pair of conditions are imposed on fixed point formulas:

$$\begin{array}{l}\text{if } \mu Z.\,\Phi \text{ is a subformula of } \nu Y.\,\Psi \text{ then } Y \text{ is not free in } \Phi\\ \text{if } \nu Z.\,\Phi \text{ is a subformula of } \mu Y.\,\Psi \text{ then } Y \text{ is not free in } \Phi\end{array}$$

This fragment of modal mu-calculus turns out to be very natural, for on infinite binary trees it is precisely the sublogic which is equi-expressive to weak monadic second-order theory of 2 successors (when the second-order quantifiers range over finite sets). For any $n \geq 0$ the alternation depth fragment ad_n can be defined as a generalisation of this [14, 19, 5][4]. Bradfield showed that there is a full alternation depth hierarchy of expressiveness using methods from descriptive set theory [5].

An alternative, but equivalent, interpretation of extremal fixed points is in terms of approximants. We provide a syntactic characterization in infinitary modal logic. 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:

$$\begin{array}{llll} \nu Z^0.\,\Phi & = & \mathtt{tt} & \mu Z^0.\,\Phi = \mathtt{ff} \\ \nu Z^{\alpha+1}.\,\Phi & = & \Phi\{\nu Z^{\alpha}.\,\Phi/Z\} & \mu Z^{\alpha+1}.\,\Phi = \Phi\{\mu Z^{\alpha}.\,\Phi/Z\} \\ \nu Z^{\lambda}.\,\Phi & = & \bigwedge\{\nu Z^{\alpha}.\,\Phi : \alpha < \lambda\} & \mu Z^{\lambda}.\,\Phi = \bigvee\{\mu Z^{\alpha}.\,\Phi : \alpha < \lambda\} \end{array}$$

A simple consequence is the following pair

[4] If Φ contains no fixed point operators then $\Phi \in \Sigma_0 \cup \Pi_0$. If $\Phi \in \Sigma_n \cup \Pi_n$ then $\Phi \in \Sigma_{n+1} \cup \Pi_{n+1}$. If $\Phi, \Psi \in \Sigma_n(\Pi_n)$ then $[K]\Phi$, $\langle K\rangle\Phi$, $\Phi \wedge \Psi$, $\Phi \vee \Psi \in \Sigma_n(\Pi_n)$. If $\Phi \in \Sigma_n$ then $\mu Z.\,\Phi \in \Sigma_n$. If $\Phi \in \Pi_n$ then $\nu Z.\,\Phi \in \Pi_n$. If $\Phi, \Psi \in \Sigma_n(\Pi_n)$ then $\Phi\{\Psi/Z\} \in \Sigma_n(\Pi_n)$. Let ad_n be $\{\Phi : \Phi \in \Sigma_{n+1} \cap \Pi_{n+1}\}$. Thus the alternation free fragment is ad_1.

$$E \models_{\mathcal{V}} \nu Z.\,\Phi \quad \text{iff} \quad E \models_{\mathcal{V}} \nu Z^{\alpha}.\,\Phi \ \text{ for all } \alpha.$$
$$E \models_{\mathcal{V}} \mu Z.\,\Phi \quad \text{iff} \quad E \models_{\mathcal{V}} \mu Z^{\alpha}.\,\Phi \ \text{ for some } \alpha.$$

As μM contains M and is contained in infinitary modal logic it follows that if $E \sim F$ then for all closed $\Phi \in \mu M$. $E \models \Phi$ iff $F \models \Phi$.

3.4 *Second-order propositional modal logic*

We can also contrast second-order propositional modal logic, $2M$, with modal mu-calculus. $2M$ is defined as an extension of μM as follows

$$\Phi \ ::= \ \mathtt{tt} \mid Z \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid [K]\Phi \mid \Box\,\Phi \mid \forall Z.\Phi$$

The modality $\Box$ is the reflexive and transitive closure of $[-]$, and is included so that fixed points are definable within $2M$. Negation is included explicitly, and we assume the expected derived operators. As with modal mu-calculus we define when a process E has a property Φ relative to $\mathcal{V}$, written $E \models_{\mathcal{V}} \Phi$. The new clauses are:

$$E \models_{\mathcal{V}} \neg\Phi \quad \text{iff} \quad E \not\models_{\mathcal{V}} \Phi$$
$$E \models_{\mathcal{V}} \Box\Phi \quad \text{iff} \quad \forall F \in \mathcal{P}(E).\ F \models_{\mathcal{V}} \Phi$$
$$E \models_{\mathcal{V}} \forall Z.\Phi \quad \text{iff} \quad \forall \mathcal{E} \subseteq \mathcal{P}(E).\ E \models_{\mathcal{V}[\mathcal{E}/Z]} \Phi$$

Notice that $\Box$ is definable in μM: assuming Z is not free in Φ, the formula $\Box\Phi$ is $\nu Z.\,\Phi \wedge [-]Z$. The operator $\forall Z$ is a set quantifier, ranging over subsets of $\mathcal{P}(E)$. There is a straightforward translation of μM into $2M$. Let Tr be this translation. The important cases are the fixed points: $\mathrm{Tr}(\nu Z.\,\Phi) = \exists Z.(Z \wedge \Box(Z \to \mathrm{Tr}(\Phi)))$ and $\mathrm{Tr}(\mu Z.\,\Phi) = \forall Z.(\Box(\mathrm{Tr}(\Phi) \to Z) \to Z)$.

It follows from Janin and Walukiewicz's result that modal mu-calculus is equi-expressive to the bisimulation closed subset of $2M$. An open question is whether this remains true when we restrict to finite-state transition graphs. Van Benthem's result remains true under this restriction (as shown by Rosen). Part of the interest in the relationship between μM and $2M$ on finite models is that within $2M$ one can define NP-complete problems, an example is 3-colourability on finite connected undirected graphs. Consider such a graph. If there is an edge between two vertices E and F let $E \xrightarrow{a} F$ and $F \xrightarrow{a} E$. So in this case $\mathcal{A} = \{a\}$, and 3-colourability is given by:

$$\exists X.\,\exists Y.\,\exists Z.\,(\Phi \wedge \Box((X \to [a]\neg X) \wedge (Y \to [a]\neg Y) \wedge (Z \to [a]\neg Z)))$$

where Φ, which says that every vertex has a unique colour, is

$$\Box((X \wedge \neg Y \wedge \neg Z) \vee (Y \wedge \neg Z \wedge \neg X) \vee (Z \wedge \neg X \wedge \neg Y))$$

In contrast, modal mu-calculus can only express PTIME graph properties (this follows from [17]). Recently Otto showed that an extension of modal mu-calculus (with increased fixed point arities) exactly captures the PTIME bisimulation closed graph properties [29].

We can also contrast μM and $2M$ using games that are extensions of the bisimulation game defined earlier. These extended games include moves for colouring processes [33].

4 Property Checking and Games

Modal mu-calculus is a very rich temporal logic which is able to describe a range of useful properties of processes. The next step is to provide techniques for verification, for showing when processes have, or fail to have, these features.

In the case of finite-state systems a popular approach is to use automatic methods, to build an algorithm called a model checker. The first model checkers were for CTL and used depth first search. The size of the problem is the size of the process (the number of states in its transition system) times the size of the formula (the number of subformulas). For CTL there is a straightforward linear time algorithm. As more complex temporal logics were used, more sophisticated algorithms were developed. One general approach is the use of automata. One can often reduce a model checking problem to a nonemptiness problem for a class of automata [12, 13, 15, 4]. In the case of modal mu-calculus the first algorithms appealed to approximants [14]. For each subformula $\sigma Z.\Phi$ there is the formula $\sigma Z^n\Phi$ where n is at most the size of the state space[5]. The question then is how many of these approximants need to be calculated, and to what extent (because of monotonicity) one can reuse information. In general the algorithm is exponential in the alternation depth of the formula. The most far reaching is [23] (but at the expense of space efficiency). These methods tend to be "global": to show if $E \models_{\mathcal{V}} \Phi$ one constructs the sets $\{F \in \mathcal{P}(E).\ F \models_{\mathcal{V}} \Psi\}$ for each subformula Ψ of Φ. Local techniques, in contrast, try and directly solve whether $E \models_V \Phi$. Local methods also apply to infinite state systems, and are often presented using tableaux [35, 7]. Another general question is to what extent property checking can be guided by the algebraic structure of the process, for instance see [2].

Discovering fixed point sets in general is not easy, and is therefore liable to lead to errors. Instead we would like simpler, and consequently safer, methods for checking whether temporal properties hold. Towards this end we first provide a different characterisation of the satisfaction relation between a process and a formula in terms of game playing. We then look at consequences for the finite state case including connections with other interesting combinatorial problems. First we define some preliminary notions.

The size of Φ, denoted by $|\Phi|$, is the number of "symbols" in Φ (and so, for instance, $|\Phi_1 \wedge \Phi_2|$ is $1 + |\Phi_1| + |\Phi_2|$). Let $\mathrm{Sub}(\Phi)$ be the set of subformulas of Φ (where, for example, $\mathrm{Sub}(\nu Z.\,\Phi)$ is $\{\nu Z.\,\Phi\} \cup \mathrm{Sub}(\Phi)$). We say that a formula Φ is *normal* if every occurrence of a binder σZ in Φ binds a distinct variable, and also no free variable Z in Φ is also used in a binder σZ. Every formula can be converted into a normal formula of the same size by renaming bound variables. If $\sigma Z.\,\Psi$ is a subformula of a normal formula Φ then we can use the binding variable Z to uniquely identify this subformula. If Φ is normal and $\sigma X.\Psi, \sigma Z.\Psi' \in \mathrm{Sub}(\Phi)$ then we say that X *subsumes* Z if $\sigma Z.\Psi' \in \mathrm{Sub}(\sigma X.\Psi)$. Notice that subsumes is reflexive and transitive, and if $\sigma X.\Psi$ and $\sigma Z.\Psi' \in \mathrm{Sub}(\Phi)$ and X subsumes Z and $X \neq Z$ then $|\sigma X.\Psi| > |\sigma Z.\Psi'|$.

[5] Notice that if $|\mathcal{P}(E)| = n$ then $E \models_{\mathcal{V}} \nu Z.\,\Phi$ iff for all $k \leq n$, $E \models_{\mathcal{V}} \nu Z^k.\,\Phi$ (and $E \models_{\mathcal{V}} \mu Z.\,\Phi$ iff for some $k \leq n$, $E \models_{\mathcal{V}} \mu Z^k.\,\Phi$).

- if $\Phi_j = \Psi_1 \wedge \Psi_2$ then player I chooses one of the conjuncts Ψ_i, $i \in \{1,2\}$: the process E_{j+1} is E_j and Φ_{j+1} is Ψ_i.
- if $\Phi_j = \Psi_1 \vee \Psi_2$ then player II chooses one of the disjuncts Ψ_i,$i \in \{1,2\}$: the process E_{j+1} is E_j and Φ_{j+1} is Ψ_i.
- if $\Phi_j = [K]\Psi$ then player I chooses a transition $E_j \xrightarrow{a} E_{j+1}$ with $a \in K$ and Φ_{j+1} is Ψ.
- if $\Phi_j = \langle K\rangle\Psi$ then player II chooses a transition $E_j \xrightarrow{a} E_{j+1}$ with $a \in K$ and Φ_{j+1} is Ψ.
- if $\Phi_j = \sigma Z.\,\Psi$ then Φ_{j+1} is Z and E_{j+1} is E_j.
- if $\Phi_j = Z$ and the subformula of Φ_0 identified by Z is $\sigma Z.\,\Psi$ then Φ_{j+1} is Ψ and E_{j+1} is E_j.

FIG. 3. Rules for the next move in a game play

4.1 *Property checking as a game*

The *property checking game* $\mathcal{G}_{\mathcal{V}}(E,\Phi)$, when $\mathcal{V}$ is a valuation, E a process and Φ a *normal* formula, is played by two participants, players I and II. Player I attempts to show that E fails to have the property Φ relative to $\mathcal{V}$ whereas player II wishes to establish that E does have Φ (relative to $\mathcal{V}$). Unlike the earlier bisimulation game, the two players do not necessarily move in turn[6].

A play of $\mathcal{G}_{\mathcal{V}}(E_0,\Phi_0)$ is a finite or infinite length sequence of the following form $(E_0,\Phi_0)\ldots(E_n,\Phi_n)\ldots$ where each $\Phi_i \in \mathrm{Sub}(\Phi_0)$ and each $E_i \in \mathcal{P}(E_0)$. Suppose part of a play is $(E_0,\Phi_0)\ldots(E_j,\Phi_j)$. The next move and which player makes it depends on the formula Φ_j: the moves are given in Figure 3. In the rules for fixed point formulas we use the fact that the starting formula Φ_0 is normal. Each time the current game configuration is $(E,\sigma Z.\Psi)$, at the next step this fixed point is abbreviated to Z, and each time the configuration is (F,Z) the fixed point subformula Z identifies is, in effect, unfolded once as the formula becomes Ψ[7].

The conditions for winning a play are given in Figure 4. A player wins if her opponent is stuck (condition 2 for both players). Player I wins if a blatantly false configuration is reached. and player II wins if a blatantly true configuration is reached. The remaining condition identifies who wins an infinite length play. In the case of the bisimulation game any infinite length play is won by player II. For property checking the winner depends on the "outermost fixed point" subformula that is unfolded infinitely often: if it is a least fixed point subformula player I wins and if it is a greatest fixed point subformula player II is the winner. The notion of outermost fixed point is given in terms of subsumption. For example, in the case of $\sigma X_1\,\sigma X_2\ldots\sigma X_n.\Phi(X_1,\ldots,X_n)$, any of the X_i may occur infinitely often in an infinite length play. However there is just one X_j which occurs infinitely often and which subsumes any other X_k which also occurs infinitely often: this X_j identifies the outermost fixed point subformula which decides who wins the play. Lemma 1 generalises this observation.

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

[7] As there are no choices here neither player is responsible for these moves.

Player I wins

1. The play is $(E_0, \Phi_0) \dots (E_n, \Phi_n)$ and either $\Phi_n = \mathtt{ff}$ or $\Phi_n = Z$ and Z is free in Φ_0 and $E_n \notin \mathcal{V}(Z)$.
2. The play is $(E_0, \Phi_0) \dots (E_n, \Phi_n)$ and $\Phi_n = \langle K \rangle \Psi$ and $\{F : E \xrightarrow{a} F \text{ and } a \in K\} = \emptyset$.
3. The play $(E_0, \Phi_0) \dots (E_n, \Phi_n) \dots$ has infinite length and the unique variable X which occurs infinitely often and which subsumes all other variables that occur infinitely often identifies a least fixed point subformula $\mu X.\, \Psi$.

Player II wins

1. The play is $(E_0, \Phi_0) \dots (E_n, \Phi_n)$ and either $\Phi_n = \mathtt{tt}$ or $\Phi_n = Z$ and Z is free in Φ_0 and $E_n \in \mathcal{V}(Z)$.
2. The play is $(E_0, \Phi_0) \dots (E_n, \Phi_n)$ and $\Phi_n = [K]\Psi$ and $\{F : E \xrightarrow{a} F \text{ and } a \in K\} = \emptyset$.
3. The play $(E_0, \Phi_0) \dots (E_n, \Phi_n) \dots$ has infinite length and the unique variable X which occurs infinitely often and which subsumes all other variables that occur infinitely often identifies a greatest fixed point subformula $\nu X.\, \Psi$.

FIG. 4. Winning conditions

Lemma 1 *If $(E_0, \Phi_0) \dots (E_n, \Phi_n) \dots$ is an infinite length play of $\mathcal{G}_\mathcal{V}(E_0, \Phi_0)$ then there is a unique variable X which occurs infinitely often (for infinitely many j, $X = \Phi_j$) and, if Y also occurs infinitely often then X subsumes Y.*

As with the bisimulation game one of the players is able to win any play of a property checking game irrespective of which moves her opponent makes. A strategy for a player is, as previously, a family of rules which tell the player how to move. Again we only need consider history-free strategies. For player I the rules are of two kinds:

$$\begin{array}{ll} \text{at} & \text{position } (E, \Phi_1 \wedge \Phi_2) \text{ choose } (E, \Phi_i) \\ \text{at} & \text{position } (E, [K]\Phi) \text{ choose } (F, \Phi) \end{array}$$

where Φ_1 is either Φ_1 or Φ_2 and where $E \xrightarrow{a} F$ for some $a \in K$. For player II they are similar

$$\begin{array}{ll} \text{at} & \text{position } (E, \Phi_1 \vee \Phi_2) \text{ choose } (E, \Phi_i) \\ \text{at} & \text{position } (E, \langle K \rangle \Phi) \text{ choose } (F, \Phi) \end{array}$$

A player uses the strategy π in a play if all her moves obey the rules in π, and π is winning if the player wins every play in which she uses π.

Proposition 1 *$E \models_\mathcal{V} \Phi$ iff player II has a history-free winning strategy for $\mathcal{G}_\mathcal{V}(E, \Phi)$.*

This result follows either by using approximants or by using techniques that are discussed later.

Example 1 Let $\mathtt{D} \xrightarrow{a} \mathtt{D}'$, $\mathtt{D}' \xrightarrow{a} \mathtt{D}$ and $\mathtt{D}' \xrightarrow{b} \mathtt{D}''$, and let Ψ be the following formula $\mu Y.\, \nu Z.\, [a]((\langle b\rangle \mathtt{tt} \vee Y) \wedge Z)$. $\mathtt{D}'$ (and $\mathtt{D}$) fails to have the property Ψ, and so player I has a winning strategy for $\mathcal{G}(\mathtt{D}', \Psi)$[8]. The important rules are: at $(\mathtt{D}, (\langle b\rangle \mathtt{tt} \vee Y) \wedge Z)$ choose $(\mathtt{D}, \langle b\rangle \mathtt{tt} \vee Y)$ and at $(\mathtt{D}', (\langle b\rangle \mathtt{tt} \vee Y) \wedge Z)$ choose $(\mathtt{D}', Z)$. □

Property checking is closed under complement. For each formula Φ define Φ^c as follows:

$$\begin{array}{llll} \mathtt{tt}^c = \mathtt{ff} & \mathtt{ff}^c = \mathtt{tt} & Z^c = Z & \\ (\Phi \wedge \Psi)^c = \Phi^c \vee \Psi^c & (\Phi \vee \Psi)^c = \Phi^c \wedge \Psi^c & & \\ (\nu Z.\, \Phi)^c = \mu Z.\, (\Phi)^c & (\mu Z.\, \Phi)^c = \nu Z.\, (\Phi)^c & & \end{array}$$

For any valuation $\mathcal{V}$ let $\mathcal{V}^c$ be its "complement" (with respect to a fixed E), the valuation such that for any Z the set $\mathcal{V}^c(Z) = \mathcal{P}(E) - \mathcal{V}(Z)$.

Proposition 2 *Player II does not have a winning strategy for $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ iff Player II has a winning stategy for $\mathcal{G}_{\mathcal{V}^c}(E, \Phi^c)$.*

$\mathcal{G}_{\mathcal{V}^c}(E, \Phi^c)$ is the dual of $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ (where the players reverse their role, and the blatantly true and false end positions are interchanged).

4.2 Model checking and MC games

The game view of having a property holds for arbitrary processes whether they be finite or infinite-state. In the general case property checking is undecidable: for instance the halting problem is equivalent to whether $\mathtt{TM}_n \models \nu Z.\, \langle -\rangle \mathtt{tt} \wedge [-]Z$ where $\mathtt{TM}_n$ is the coding of the nth Turing machine. However for classes of infinite-state processes property checking is decidable [8]. We now examine the consequences for finite-state processes.

Assume that E is a finite-state process (that is, $\mathcal{P}(E)$ has finite size). The *game graph* for $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ is the directed graph representing all possible plays of $\mathcal{G}_{\mathcal{V}}(E, \Phi)$. The vertices are configurations of a possible play, and have the form (F, Ψ) where $F \in \mathcal{P}(E)$ and $\Psi \in \mathrm{Sub}(\Phi)$. 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. The size (the number of vertices) in the game graph for $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ is therefore bounded by $|\mathcal{P}(E)| \times |\mathrm{Sub}(\Phi)|$. The model checking decision question (in terms of games) is: does player II have a winning strategy for $\mathcal{G}_{\mathcal{V}}(E, \Phi)$?

Model checking can be abstracted into the following simple graph game, which we call the MC game. An MC game is a directed graph $\mathcal{G} = (V, Ed, L)$ whose vertices $V = \{1, \ldots, n\}$ and whose edges $Ed \subseteq V \times V$ and where $L : V \longrightarrow \{\mathrm{I}, \mathrm{II}\}$ labels each vertex with I or with II. Each vertex $i \in V$ has at least one edge $i \longrightarrow j \in Ed$, writing $i \longrightarrow j$ for (i, j). The game is a contest between player I and II. It begins with a token on the initial vertex 1. When the token is on vertex i and $L(i) = I$ player I moves it along one of the outgoing edges of i. When it is on a II vertex player II moves it instead. A play therefore consists of an infinite path through the graph along which the token passes. The winner of a play is determined by the label of the *least* vertex i which is traversed infinitely often: if $L(i) = I$ then player I wins, and otherwise $L(i) = II$ and player II wins.

[8] If Φ is closed we omit the valuation $\mathcal{V}$ in $\mathcal{G}_{\mathcal{V}}$.

Proposition 3 *Every model checking game determines an MC game.*

Proof: Let $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ be the model checking game for the finite-state process E and the normal formula Φ. Let $E_1, \ldots, E_m$ be an enumeration of all processes in $\mathcal{P}(E)$ with $E = E_1$. Assume that $Z_1, \ldots, Z_k$ are all the bound variables in Φ (and so for each Z_i there is the subformula $\sigma Z_i.\Psi_i$). Let $\Phi_1, \ldots, \Phi_l$ be an enumeration of all formulas in $\mathrm{Sub}(\Phi) - \{Z_1, \ldots, Z_k\}$ in decreasing order of size (and so $\Phi = \Phi_1$). Insert each Z_i directly after the fixed point it identifies. The result is a sequence of formulas $\Phi_1, \ldots, \Phi_n$ in decreasing order of size, except that Z_i counts as bigger than Ψ_i. The vertices of the MC game in order are

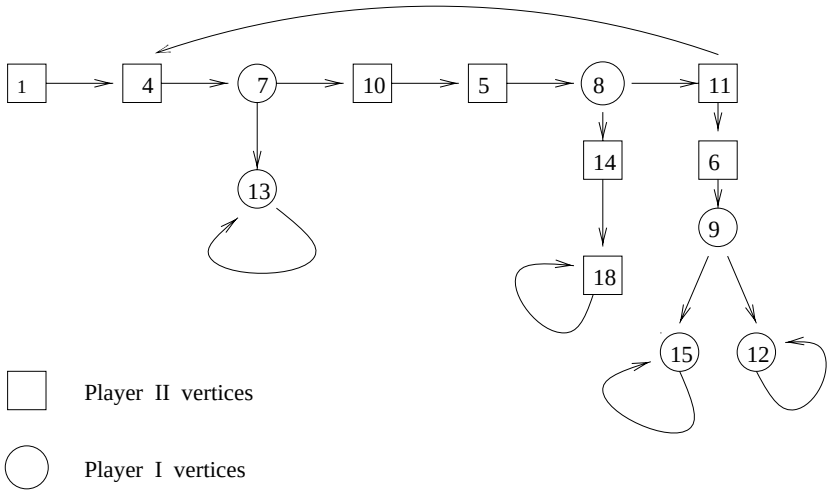
$$(E_1, \Phi_1), \ldots, (E_m, \Phi_1), \ldots, (E_1, \Phi_n), \ldots, (E_m, \Phi_n)$$

and so $V = \{1, \ldots, nm\}$ with vertex i describing (E_j, Φ_k) if $i = m(k-1) + j$. We now define the labelling of a vertex $i = (F, \Psi)$ and the edges from it, by case analysis on Ψ. If Ψ is Z and Z is free in the starting formula Φ and $F \in \mathcal{V}(Z)$ then $L(i) = II$ and there is the edge (i, i). If instead $F \notin \mathcal{V}(Z)$ then $L(i) = I$ and there is the edge (i, i). If Ψ is $\mathtt{tt}$ then $L(i) = II$ and there is the edge (i, i). If Ψ is $\mathtt{ff}$ then $L(i) = I$ and there is the edge (i, i). If Ψ is $\Psi_1 \wedge \Psi_2$ then $L(i) = I$ and there are the edges $(i, j1)$ and $(i, j2)$ where $j1$ describes (F, Ψ_1) and $j2$ describes (F, Ψ_2). Dually, if Ψ is $\Psi_1 \vee \Psi_2$ then $L(i) = II$ and there are the edges $(i, j1)$ and $(i, j2)$ where $j1$ describes (F, Ψ_1) and $j2$ describes (F, Ψ_2). If Ψ is $[K]\Psi'$ and there are no K transitions from F then $L(i) = II$ and there is the edge (i, i). Dually, if Ψ is $\langle K\rangle\Psi'$ and there are no K transitions from F then $L(i) = I$ and there is the edge (i, i). If Ψ is $[K]\Psi'$ and there are K transitions from E then $L(i) = I$ and there is an edge (i, j) for each j describing (F', Ψ') when F' is such that $F \xrightarrow{a} F'$ for $a \in K$. Dually if Ψ is $\langle K\rangle\Psi'$ and there are K transitions from E then $L(i) = II$ and there is an edge (i, j) for each j describing (F', Ψ') when F' is such that $F \xrightarrow{a} F'$ for $a \in K$. If $\Psi = \mu Z_j.\, \Psi_j$ or Z_j (when it identifies a least fixed point) then $L(i) = I$ and there is the edge (i, j) when j identifies (F, Ψ_j). Finally, if $\Psi = \nu Z_j.\, \Psi_j$ or Z_j (when it identifies a greatest fixed point) then $L(i) = II$ and there is the edge (i, j) when j identifies (F, Ψ_j). Delete any vertices which are not reachable from the initial vertex (E, Φ), and renumber remaining vertices but preserve the original ordering. Clearly player II has a winning strategy for $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ iff player II has a winning strategy for the determined MC game. □

Example 2 The game $\mathcal{G}(\mathtt{D}, \nu Z.\, \langle b\rangle\mathtt{tt} \wedge \langle -\rangle Z)$, where $\mathtt{D}$ is from Example 1 determines the MC game of Figure 5. The MC representation of positions from the model checking game are also presented. To make game playing perpetual note the loops on 11, 14, 13 and 10. □

There is a converse to Proposition 3: any MC game can be transformed into a model checking game whose size is polynomially bounded by the MC game. (See [15, 25] who show a similar result but for alternating automata and for boolean equation systems.)

In the following we consider subgames of an MC game $\mathcal{G}$. First let $\mathcal{G}(i)$ be the game $\mathcal{G}$ except that the starting vertex is i instead of 1 (and so $\mathcal{G}$ itself is $\mathcal{G}(1)$). More generally, if $\mathcal{G} = (V, Ed, L)$ and $X \subseteq V$ we let $\mathcal{G} - X$ be the structure $\mathcal{G}' = (V', Ed', L')$ where $V' = V - X$, Ed' is the set of edges $Ed \cap (V' \times V')$ and $L'(j) = L(j)$ for each $j \in V'$. In the circumstance that V' is nonempty and for each $i \in V'$ there is an edge



1	$(\mathtt{D}, \nu Z.\, \langle b\rangle \mathtt{tt} \wedge \langle -\rangle Z)$	8	$(\mathtt{D}, \langle -\rangle Z)$
2	$(\mathtt{D}, Z)$	9	$(\mathtt{D}', \langle -\rangle Z)$
3	$(\mathtt{D}', Z)$	10	$(\mathtt{D}'', \langle -\rangle Z)$
4	$(\mathtt{D}'', Z)$	11	$(\mathtt{D}, \langle b\rangle \mathtt{tt})$
5	$(\mathtt{D}, \langle b\rangle \mathtt{tt} \wedge \langle -\rangle Z)$	12	$(\mathtt{D}', \langle b\rangle \mathtt{tt})$
6	$(\mathtt{D}', \langle b\rangle \mathtt{tt} \wedge \langle -\rangle Z)$	13	$(\mathtt{D}'', \langle b\rangle \mathtt{tt})$
7	$(\mathtt{D}'', \langle b\rangle \mathtt{tt} \wedge \langle -\rangle Z)$	14	$(\mathtt{D}'', \mathtt{tt})$

FIG. 5. Game

$i \longrightarrow j \in Ed'$ we say that $\mathcal{G}'$ is a subgame of $\mathcal{G}$ (meaning that $\mathcal{G}'(k)$ is a subgame for any $k \in V'$). In the following we use game and subgame interchangeably.

We define (following [22, 26, 38]) the set of vertices for which a player P can force play to enter a subset X of vertices, as $\mathrm{Force}_P(X)$.

Definition 1 Let $G = (V, Ed, L)$ be a game and $X \subseteq V$, then let

1. $\mathrm{Force}^0_P(X) = X$ for $P \in \{I, II\}$
2. $\mathrm{Force}^{i+1}_I(X) = \mathrm{Force}^i_I(X) \cup \{j \in V : L(j) = I \text{ and } \exists k \in \mathrm{Force}^i_I(X).\, j \longrightarrow k \in Ed\}$ $\cup \{j \in V : L(j) = II \text{ and } \forall k \in V.\text{ if } j \longrightarrow k \in Ed \text{ then } k \in \mathrm{Force}^i_I(X)\}$
3. $\mathrm{Force}^{i+1}_{II}(X) = \mathrm{Force}^i_{II}(X) \cup \{j \in V : L(j) = II \text{ and } \exists k \in \mathrm{Force}^i_{II}(X).j \longrightarrow k \in Ed\} \cup \{j \in V : L(j) = I \text{ and } \forall k \in V.\text{ if } j \longrightarrow k \in Ed \text{ then } k \in \mathrm{Force}^i_{II}(X)\}$
4. $\mathrm{Force}_P(X) = \bigcup \{\mathrm{Force}^i_P(X) : i \geq 0\}$ for $P \in \{I, II\}$.

Consequently, if $j \in \mathrm{Force}_P(X)$ player P can force the play into X irrespective of whatever moves her opponent makes. The *rank* of such a vertex j is the least index i such that $j \in \mathrm{Force}^i_P(X)$: the rank is (an upper bound on) the number of moves it takes to force the play into X.

The following result shows that removing a force set from a game leaves a subgame (or the empty set).

Proposition 4 *If* $\mathcal{G} = (V, Ed, L)$ *and* $X \subseteq V$ *then either* $\text{Force}_P(X) = V$ *or* $\mathcal{G}' = \mathcal{G} - \text{Force}_P(X)$ *is a subgame.*

A history-free strategy for player P consists of a set of rules of the form "at i choose j", where P $= L(i)$ and there is an edge $i \longrightarrow j$.

Proposition 5 *For any* MC *game* $\mathcal{G}$ *and vertex* i *either player I or player II has a history-free winning strategy for* $\mathcal{G}(i)$.

Proof: Let $\mathcal{G} = (V, Ed, L)$ be an MC (sub)game. The proof is by induction on $|V|$. The base case is $|V| = 1$, in which case $V = \{i\}$. As $\mathcal{G}$ is a game it follows that $i \longrightarrow i \in Ed$. Clearly player $L(i)$ has the history free winning strategy given by the (redundant) rule at i choose i. For the inductive step let k be the least vertex in V, and let X be the set $\text{Force}_{L(k)}(\{k\})$. If $X = V$ then player $L(k)$ has a history free winning strategy for $\mathcal{G}(i)$ for each $i \in V$, by forcing play to vertex k and then playing to some $j1$ such that $k \longrightarrow j1 \in Ed$. More precisely, the strategy consists of the rule at k choose $j1$ where $j1$ has a least rank in $\text{Force}_{L(k)}(\{k\})$ among the set $\{j : k \longrightarrow j \in Ed\}$, and for any other $l \in X$ such that $L(l) = L(k)$ choose the edge $l \longrightarrow l1$ such that $l1$ has a least rank. Clearly this is a winning strategy as every play must proceed through the vertex k which is least in V. Otherwise $X \neq V$. Let $\mathcal{G}'$ be the subgame $\mathcal{G} - X$ which is strictly smaller in size than $\mathcal{G}$. By the induction hypothesis, for each $j \in V'$ player P_j has a history free winning strategy σ_j for the game $\mathcal{G}'(j)$. Partition these vertices into W_I, those won by player I, and into W_{II}, those won by player II. (Note that $W_P = \text{Force}_P(W_P)$ in the game $\mathcal{G}'$.) The proof now consists of examining two subcases, depending on the set $Y = \{j : k \longrightarrow j \in Ed\}$.

Case 1: $Y \cap (W_{L(k)} \cup X) \neq \emptyset$. There is an edge $k \longrightarrow j1 \in Ed$ and $j1 \in X$ or $j1 \in W_{L(k)}$. Player $L(k)$ now has a history free winning strategy for $\mathcal{G}(i)$ for each $i \in (X \cup W_{L(k)})$. To see this let σ' be the substrategy for $L(k)$ which forces any play in X to k together with the rule at k choose $j1$ (determined as earlier). If $j1 \in X$ then σ' is a history free winning strategy for $\mathcal{G}(i)$ for any $i \in X$, and $\sigma' \cup \sigma_i$ is a history free winning strategy for $\mathcal{G}(i)$ for any $i \in W_{L(k)}$. If $j1 \in W_{L(k)}$ then the strategy $\sigma' \cup \sigma_{j1}$ is winning for $\mathcal{G}(i)$ for $i \in X \cup U$ where U consists of vertices visited in $W_{L(k)}$ using σ_{j1}. For other vertices j in $W_{L(k)}$ the history free winning strategy gives priority to the rules $\sigma' \cup \sigma_{j1}$ and uses σ_j otherwise. The opponent O of $L(k)$ has the history free winning strategy σ_i for each game $\mathcal{G}(i)$ if $i \in W_O$.

Case 2: $Y \cap (W_{L(k)} \cup X) = \emptyset$. This means that for every $j1$ such that $k \longrightarrow j1$ the opponent O of $L(k)$ has a history free winning strategy for $\mathcal{G}'(j1)$. Let $Z = \text{Force}_O(W_O)$ with respect to the full game $\mathcal{G}$. For each $i \in Z$ player O has a history free winning strategy for $\mathcal{G}(i)$: the strategy consists of forcing play into W_O and then using the winning strategies determined from $\mathcal{G}'$, similar to above. Let σ'_i be the history free strategy for any $i \in Z$. If $Z = V$ then σ'_i is the strategy for $\mathcal{G}(i)$. Otherwise, consider the subgame $\mathcal{G}'' = \mathcal{G} - Z$. As Z is non-empty by the induction hypothesis, for each j in $\mathcal{G}''$ player P_j has a history free winning strategy σ''_j for $\mathcal{G}''(j)$. If $P_j = L(k)$ then σ''_j is a history free winning strategy for $L(k)$ for the game $\mathcal{G}(j)$. Otherwise, $P_j = O$, and player O has a history free winning strategy for $\mathcal{G}(j)$: player O uses the partial strategy $\sigma''(j)$ until, if at all, player $L(k)$ plays into the set Z in which case player O uses the appropriate winning strategy, remaining in Z: we leave the formal details to the reader. □

Implicit in the proof of Proposition 5 is an exponential time algorithm for deciding which player has a winning strategy for an MC game: with respect to a game of size n the proof may call twice subproofs for games of size less than n, once at the beginning and once in case 2. Note that the algorithm also produces a history-free winning strategy. In the case of model checking this strategy can be used interactively with a user to understand why the property holds or fails to hold. It is an open question whether there is a polynomial time algorithm for this problem. For subclasses of MC games there are known polynomial time decision procedures. One family is MC games determined by model checking games $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ when Φ is an alternation free formula. A second family is simple MC games. An MC game is II-simple if every II vertex i (that is, $L(i) = II$) has exactly one edge. The proof of Proposition 5 restricted to II-simple games leads to a polynomial time algorithm, as Case 2 can be solved directly without calling the induction hypothesis on $\mathcal{G}''$. A similar definition and comments apply to I-simple games.

Proposition 6 *The decision problem for MC games belongs to* NP ∩ co-NP.

Proof: Guess a history-free winning strategy π for player II (which is linear in the size of the game). Delete all edges from a player II vertex which are not consistent with π. The result is a II-simple MC game. This shows that the decision problem belongs to NP. MC games are easily complemented by interchanging I and II labels: if $\mathcal{G} = (V, Ed, L)$ let $\mathcal{G}' = (V, Ed, L')$ where $L'(i) = I$ iff $L(i) = II$, and so a player has a winning strategy for $\mathcal{G}(i)$ iff her opponent has a winning strategy for $\mathcal{G}'(i)$. Hence this shows that the decision problem also belongs to co-NP. □

Emerson, Jutla and Sistla showed that the model checking decision problem belongs to NP ∩ co-NP using automata theoretic techniques [15]. An alternative proof of Proposition 6 is sketched below.

The game graph of an MC game is an alternating automaton [4, 28]: 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 dependent loops. Alternatively an MC game can be directly translated into a closed 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 ∩ co-NP. Various authors have, in effect, translated finite state model checking into this logic, with a preference for a syntax utilizing equations [1, 25], and an elegant technique for solving this problem uses Gaussian elimination [25], (and also see [20]).

4.3 Other graph games

MC games are finite graph games. They can be generalised to countably infinite games $\mathcal{G} = (V, Ed, L, \mathcal{C})$ where $V = \mathbb{N}$ and the extra component $\mathcal{C}$ is a finite ordered sequence of colours $C_1, \ldots C_k$. Each vertex $i \in V$ has exactly one colour C_j, and colours respect players: that is, for any i and i' with colour C_j player $L(i) = L(i')$. The winner of a play is given by the player associated with the least colour which occurs infinitely often. Any property checking game $\mathcal{G}_{\mathcal{V}}(E, \Phi)$ determines a generalised MC game (as the colours are just the elements of Sub(Φ)). Proposition 3 of the previous section

generalises to this class of games. The bisimulation game of Section 3 also determines a simple (possibly infinite) graph game $\mathcal{G} = (V, Ed, L, W)$ where W is the set of I-wins: player I wins from a vertex if she can force play into W, otherwise player II wins[9]. By defining graph games with more elaborate winning conditions, winning strategies need no longer be history-free, see [26, 38].

An alternative extension of finite graph games is to enrich their structure with probabilities. A simple stochastic game [11], SSG, is a finite graph game whose 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). Each I and II vertex (other than the sinks) has at least one outgoing edge, and each A vertex has exactly 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 (which includes the case of the game going on forever). 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. 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. Condon showed the decision question belongs to NP $\cap$ co-NP [11]. In [24] a "subexponential" $(2^{O(\sqrt{n})})$ algorithm is presented, which works by refining optimal strategies.

Mark Jerrum showed that there is a polynomial reduction from the MC game to SSG. 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 removed, and the edge $j \longrightarrow i1$ is added. Two new edges are added for each A vertex $i1$: first an edge to i labelled with probability $1 - 8^{-i}$, and second an edge to I-sink if i is labelled I or to II-sink if it is labelled II with probability $\frac{1}{8^i}$. It now follows that player II has a winning strategy for the MC game iff she has one for the resulting SSG.

Proposition 7 *Each MC game can be polynomially reduced to an SSG.*

The converse of this is unlikely to be true. Jerrum also shows that there is a polynomial time reduction from the MC game to the mean payoff game.

One direction for further research for model checking is to provide a finer analysis of winning strategies, and to be able to describe optimizations of them as with SSG. In general research in graph games is an exciting area, which may lead to solutions of important open problems.

Acknowledgement: Many thanks to Olaf Burkart and Mark Jerrum for discussions about MC 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

[9] When games may have countably infinite vertices the definition of $\text{Force}_P(X)$ appeals to ordinals.

mu-calculus. *Procs 9th IEEE Symposium on Logic in Computer Science*, 144-153.

[3] van Benthem, J. (1984). Correspondence theory. In *Handbook of Philosophical Logic*, Vol. II, ed. Gabbay, D. and Guenthner, F., 167-248, Reidel.

[4] Bernholtz, O., Vardi, M. and Wolper, P. (1994). An automata-theoretic approach to branching-time model checking. *Lecture Notes in Computer Science*, **818**, 142-155.

[5] Bradfield, J. (1996). The modal mu-calculus alternation hierarchy is strict. *Lecture Notes in Computer Science*, **1119**, 233-246.

[6] Bradfield, J. and Stirling, C. (1990). Verifying temporal properties of processes. *Lecture Notes in Computer Science*, **458**, 115-125.

[7] Bradfield, J. and Stirling, C. (1992). Local model checking for infinite state spaces. *Theoretical Computer Science*, **96**, 157-174.

[8] Burkart, O., and Steffen, B.(1995). Composition, decomposition, and model checking of push-down processes. *Nordic Journal of Computing*, **2**, 89-125.

[9] Christensen, S., Hirshfeld, Y., and Moller, F. (1993). Bisimulation is decidable for basic parallel processes. *Lecture Notes in Computer Science*, **715**, 143-157.

[10] Christensen, S., Hüttel, H., and Stirling, C. (1995). Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, **121**, 143-148.

[11] Condon, A. (1992). The complexity of stochastic games. *Information and Computation*, **96**, 203-224.

[12] Emerson, E. (1985). Automata, tableaux, and temporal logics. *Lecture Notes in Computer Science*, **193**, 79-87.

[13] Emerson, E., and Jutla, C. (1988). The complexity of tree automata and logics of programs. *Extended version from FOCS '88.*

[14] Emerson, E, and Lei, C. (1986). Efficient model checking in fragments of the propositional mu-calculus. In *Procs 1st IEEE Symposium on Logic in Computer Science*, 267-278.

[15] Emerson, E., Jutla, C., and Sistla, A. (1993). On model checking for fragments of μ-calculus. *Lecture Notes in Computer Science*, **697**, 385-396.

[16] Hennessy, M. and Milner, R. (1985). Algebraic laws for nondeterminism and concurrency. *Journal of Association of Computer Machinery*, **32**, 137-162.

[17] Immermann, N. (1986). Relational queries computable in polynomial time. *Information and Control*, **68**, 86-104.

[18] Janin, D. and Walukiewicz, I (1996). On the expressive completeness of the propositional mu-calculus with respect to the monadic second order logic. *Lecture Notes in Computer Science*, **1119**, 263-277.

[19] Kaivola, R. (1995). On modal mu-calculus and Büchi tree automata. *Information Processing Letters*, **54**, 17-22.

[20] Karlokoti, K. (1996). Model checking in the modal μ-calculus by substitutions. *Submitted for publication.*

[21] Kozen, D. (1983). Results on the propositional mu-calculus. *Theoretical Computer Science*, **27**, 333-354.

[22] Lescow, H. (1995). On polynomial-size programs winning finite-state games. *Lecture Notes in Computer Science*, **939**, 239-252.

[23] Long, D., Browne, A., Clarke, E., Jha, S., and Marrero, W. (1994) An improved algorithm for the evaluation of fixpoint expressions. *Lecture Notes in Computer Science*, **818**, 338-350.

[24] Ludwig, W. (1995). A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, **117**, 151-155.

[25] Mader, A. (1997). *Verification of modal properties using boolean equation systems.* PhD Thesis, Technical University of Munich.

[26] McNaughton, R. (1993). Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, **65**, 149-184.

[27] Milner, R. (1989). *Communication and Concurrency.* Prentice Hall.

[28] Muller, D., Saoudi, A. and Schupp, P. (1986). Alternating automata, the weak monadic theory of the tree and its complexity. *Lecture Notes in Computer Science*, **225**, 275-283.

[29] Otto, M. (1997). Bisimulation-invariant ptime and higher-dimensional μ-calculus. *Preliminary report RWTH Aachen.*

[30] Park, D. (1981). Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science*, **154**, 561-572.

[31] Stirling, C. (1995). Local model checking games. *Lecture Notes in Computer Science*, **962**, 1-11.

[32] Stirling, C. (1996). Modal and temporal logics for processes. *Lecture Notes in Computer Science*, **1043**, 149-237.

[33] Stirling, C. (1996). Games and modal mu-calculus. *Lecture Notes in Computer Science*, **1055**, 298-312.

[34] Stirling, C. (1996). Decidability of bisimulation equivalence for normed pushdown processes. *Lecture Notes in Computer Science*, **1119**, 217-232.

[35] Stirling, C. and Walker, D. (1991). Local model checking in the modal mu-calculus. *Theoretical Computer Science*, **89**, 161-177.

[36] Streett, R. and Emerson, E. (1989). An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, **81**, 249-264.

[37] Thomas, W. (1993). On the Ehrenfeucht-Fraïssé game in theoretical computer science. *Lecture Notes in Computer Science*, **668**.

[38] Thomas, W. (1995). On the synthesis of strategies in infinite games. *Lecture Notes in Computer Science*, **900**, 1-13.

Received August 1998