

Bisimulation, Model Checking and Other Games *

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

Contents

1	Introduction	2
2	Process Calculi	2
3	Equivalences, Modal and Temporal Logics	6
3.1	Interactive games and bisimulations	8
3.2	Modal logic and bisimulations	13
3.3	Temporal properties and modal mu-calculus	15
3.4	Second-order propositional modal logic	21
4	Property Checking and Games	22
4.1	Subformulas and subsumption	24
4.2	Property checking as a game	25
4.3	Model checking and MC games	30
4.4	Other graph games	37
	References	38

*Notes for Mathfit Instructional Meeting on Games and Computation, University of Edinburgh, June 23–24. Thanks to EPSRC and LMS for supporting the meeting.

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 due to Park and Milner is described. It is essentially game theoretic and so we build on this view. 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.

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 la-

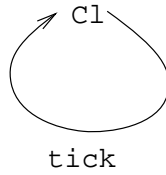


Figure 1: The transition system for $C1$

belled 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:

$$C1 \stackrel{\text{def}}{=} \text{tick}.C1$$

Names of actions, such as `tick`, are in lower case whereas names of processes, such as `C1`, have an initial capital letter. The clock `C1` is defined as `tick.C1`, where both occurrences of `C1` name the same process. The expression `tick.C1` invokes a prefix operator `.` which builds the process $a.E$ from the action a and the process E .

The behaviour of `C1` is very elementary, as it can only perform the action `tick` and in so doing becomes `C1` 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 $C1 \xrightarrow{\text{tick}} C1$. The behaviour of `C1` 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 `C1`. Each transition of a vertex which is derivable is depicted.

An unsophisticated vending machine `Ven` is:

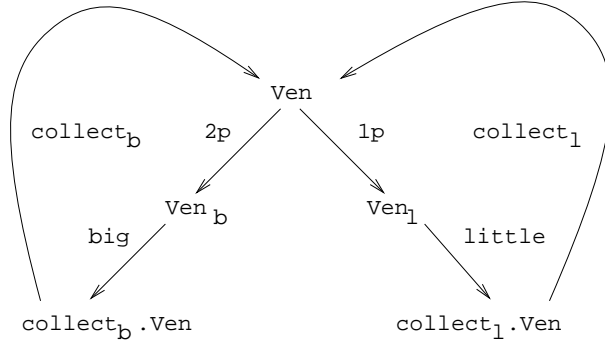


Figure 2: The transition system for Ven

$$\begin{aligned}
 \text{Ven} &\stackrel{\text{def}}{=} 2p.\text{Ven}_b + 1p.\text{Ven}_l \\
 \text{Ven}_b &\stackrel{\text{def}}{=} \text{big}.\text{collect}_b.\text{Ven} \\
 \text{Ven}_l &\stackrel{\text{def}}{=} \text{little}.\text{collect}_l.\text{Ven}
 \end{aligned}$$

Its definition employs the binary choice operator $+$ (which has wider scope than the $.$ operator). The two transition rules for $+$ are

$$\begin{aligned}
 &\text{if } E_1 \xrightarrow{a} F \text{ then } E_1 + E_2 \xrightarrow{a} F \\
 &\text{if } E_2 \xrightarrow{a} F \text{ then } E_1 + E_2 \xrightarrow{a} F
 \end{aligned}$$

The transition system for Ven is depicted in Figure 2. The operator $+$ is frequently extended to indexed families $\sum\{E_i : i \in I\}$ when I is a set of indices: the idea is that $E_1 + E_2$ abbreviates $\sum\{E_i : i \in \{1, 2\}\}$. The single rule for \sum generalizes those for $+$:

$$\text{if } E_j \xrightarrow{a} F \text{ when } j \in I \text{ then } \sum\{E_i : i \in I\} \xrightarrow{a} F$$

A special case of \sum is when the indexing set I is empty. By the transition rule this process has no transitions as the subgoal can never be fulfilled. This nil process $\sum\{E_i : i \in \emptyset\}$ is abbreviated to 0 . Thus the clock $\text{tick}.0$ can only do a single tick before terminating.

Example 1 A 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 $\sum\{\text{Cl}_i : i \geq 1\}$. Like all new clocks, Clock will eventually break down. \square

A 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 \bar{a} , its co-action. Moreover the action $\bar{\bar{a}}$ is a which means that a is also the co-action of \bar{a} . The joint activity of a and \bar{a} is the communication.

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

$$\begin{aligned} &\text{if } E \xrightarrow{a} E' \text{ and } F \xrightarrow{\bar{a}} F' \text{ then } E \mid F \xrightarrow{\tau} E' \mid F' \\ &\text{if } E \xrightarrow{a} E' \text{ then } E \mid F \xrightarrow{a} E' \mid F \\ &\text{if } F \xrightarrow{a} F' \text{ then } E \mid F \xrightarrow{a} E \mid F' \end{aligned}$$

The first of these conveys communication. The parallel operator is expressively powerful. It can be used to describe infinite state systems without invoking infinite indices. A simple example is the following counter

$$\text{Cnt} \stackrel{\text{def}}{=} \text{up}.\text{(Cnt} \mid \text{down}.0)$$

It can perform up and become $\text{Cnt} \mid \text{down}.0$ which can perform down and a further up and become $\text{Cnt} \mid \text{down}.0 \mid \text{down}.0$, and so on.

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

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

The behaviour of $E \backslash J$ is part of that of E . The presence of $\backslash J$ in $(E \mid F) \backslash J$ 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.

Example 2 The mesh of abstraction and concurrency is revealed in the following finite state example of a level crossing from [7] consisting of three components.

$$\begin{aligned}
\text{Road} & \stackrel{\text{def}}{=} \text{car.up}.\overline{\text{ccross}}.\overline{\text{down}}.\text{Road} \\
\text{Rail} & \stackrel{\text{def}}{=} \text{train.green}.\overline{\text{tcross}}.\overline{\text{red}}.\text{Rail} \\
\text{Signal} & \stackrel{\text{def}}{=} \overline{\text{green}}.\overline{\text{red}}.\text{Signal} + \overline{\text{up}}.\overline{\text{down}}.\text{Signal}
\end{aligned}$$

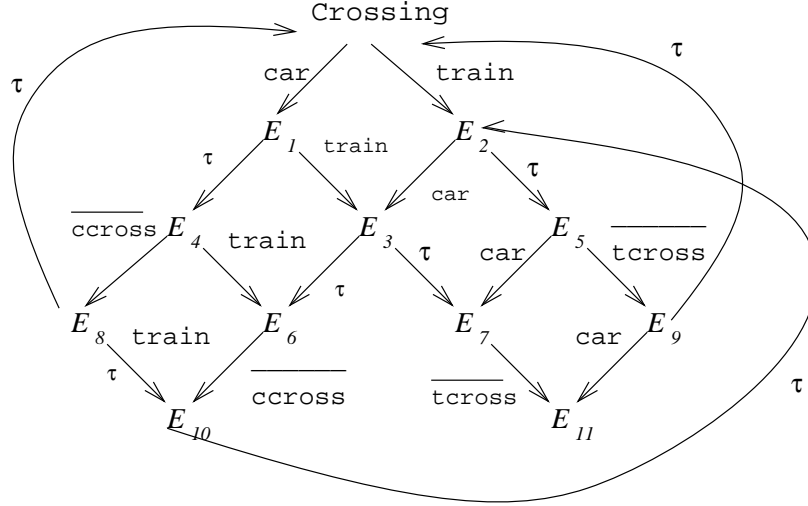
$$\text{Crossing} \equiv (\text{Road} | \text{Rail} | \text{Signal}) \setminus \{\text{green}, \text{red}, \text{up}, \text{down}\}$$

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, $\overline{\text{ccross}}$ is the car crossing, $\overline{\text{down}}$ closes the gates, green is the receipt of a green signal by the train, $\overline{\text{tcross}}$ is the train crossing, and $\overline{\text{red}}$ automatically sets the light red. Unlike most crossings it keeps the barriers down except when a car actually approaches and tries to cross. Its transition graph is depicted in Figure 3. \square

Example 2 offers a flavour of how systems are modelled in CCS. It is usual to abstract from silent activity. One defines observable transitions $E \xrightarrow{a} F$ 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. For more details see [27].

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 an alternative exposition using 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



$K = \{\text{green, red, up, down}\}$

$E_1 \equiv (\text{up}.\overline{\text{ccross}}.\overline{\text{down}}.\text{Road}|\text{Rail}|\text{Signal})\backslash K$

$E_2 \equiv (\text{Road}|\text{green}.\overline{\text{tcross}}.\overline{\text{red}}.\text{Rail}|\text{Signal})\backslash K$

$E_3 \equiv (\text{up}.\overline{\text{ccross}}.\overline{\text{down}}.\text{Road}|\text{green}.\overline{\text{tcross}}.\overline{\text{red}}.\text{Rail}|\text{Signal})\backslash K$

$E_4 \equiv (\overline{\text{ccross}}.\overline{\text{down}}.\text{Road}|\text{Rail}|\text{down}.\text{Signal})\backslash K$

$E_5 \equiv (\text{Road}|\overline{\text{tcross}}.\overline{\text{red}}.\text{Rail}|\text{red}.\text{Signal})\backslash K$

$E_6 \equiv (\overline{\text{ccross}}.\overline{\text{down}}.\text{Road}|\text{green}.\overline{\text{tcross}}.\overline{\text{red}}.\text{Rail}|\text{down}.\text{Signal})\backslash K$

$E_7 \equiv (\text{up}.\overline{\text{ccross}}.\overline{\text{down}}.\text{Road}|\overline{\text{tcross}}.\overline{\text{red}}.\text{Rail}|\text{red}.\text{Signal})\backslash K$

$E_8 \equiv (\overline{\text{down}}.\text{Road}|\text{Rail}|\text{down}.\text{Signal})\backslash K$

$E_9 \equiv (\text{Road}|\overline{\text{red}}.\text{Rail}|\text{red}.\text{Signal})\backslash K$

$E_{10} \equiv (\overline{\text{down}}.\text{Road}|\text{green}.\overline{\text{tcross}}.\overline{\text{red}}.\text{Rail}|\text{down}.\text{Signal})\backslash K$

$E_{11} \equiv (\text{up}.\overline{\text{ccross}}.\overline{\text{down}}.\text{Road}|\overline{\text{red}}.\text{Rail}|\text{red}.\text{Signal})\backslash K$

Figure 3: The transition system for Crossing

processes. We also define second-order propositional modal logic to contrast fixed points and second-order quantifiers.

3.1 Interactive games and bisimulations

Equivalences for CCS processes begin 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, using bisimulations, in terms of the ability for these observers to match their selections so that they can proceed with further corresponding choices. However we proceed with an alternative exposition using *games* which offer a powerful image for interaction.

The equivalence game $\mathcal{G}(E_0, F_0)$ on a pair of processes 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 $(E_0, F_0) \dots (E_i, F_i) \dots$. 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 $(E_0, F_0) \dots (E_j, F_j)$. The next pair (E_{j+1}, F_{j+1}) is determined by one of the following two moves:

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

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 \xrightarrow{a} : instead we could build them from the observable transitions \xRightarrow{a} .)

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: she can then choose a transition and player II will be unable to match it. 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 both processes have no available transitions. In both these circumstances player I has been unable to detect a difference between the starting processes.

Example 1 Let $C1_2 \stackrel{\text{def}}{=} \text{tick}.\text{tick}.C1_2$. Player II wins any play of the game $\mathcal{G}(C1, C1_2)$. Any play

$$(C1, C1_2) (C1, \text{tick}.C1_2) (C1, C1_2) \dots$$

proceeds forever irrespective of from which component player I chooses to make her move. \square

Example 2 In the case of $\mathcal{G}(C1, C1_5)$ when $C1_5 \stackrel{\text{def}}{=} \text{tick}.C1_5 + \text{tick}.0$, there are plays that player I wins and plays that player II wins. If player I initially moves $C1_5 \xrightarrow{\text{tick}} 0$ then after her opponent makes the move $C1 \xrightarrow{\text{tick}} C1$ the resulting position $(C1, 0)$ is a I-win. If player I always chooses transitions $C1_5 \xrightarrow{\text{tick}} C1_5$ or $C1 \xrightarrow{\text{tick}} C1$ then player II can avoid defeat. However player I has the power to win any play of $(C1, C1_5)$ by initially choosing the transition $C1_5 \xrightarrow{\text{tick}} 0$. \square

Example 2 illustrates that 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. For player I a rule has the form: if the play so far is $(E_0, F_0) \dots (E_i, F_i)$ then choose the transition t . For player II it has the form: if the play so far is $(E_0, F_0) \dots (E_i, F_i)$ and player I has chosen the transition t then choose the transition t' . 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

at position (E, F) choose transition t

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

at position (E, F) when player I has chosen t choose t'

where t is either $E \xrightarrow{a} E'$ or $F \xrightarrow{a} F'$ 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 3 Consider the two similar vending machines

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

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

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. Player II's winning strategy for $\mathcal{G}(E, E)$ is the copy-cat strategy (always choose the same transition as player I). If π is a winning strategy for player II for $\mathcal{G}(E, F)$ then π' which changes each rule “at position (G, H) choose ...” to “at position (H, G) choose ...” is a winning strategy for $\mathcal{G}(F, E)$. Finally if π is a winning strategy for player II for $\mathcal{G}(E, F)$ and π' for $\mathcal{G}(F, H)$ then the composition of these strategies is a winning strategy for player II for $\mathcal{G}(E, H)$: composition is defined in such a way that, for instance, the pair of rules “at (E', F') when player I has chosen $E' \xrightarrow{a} E''$ choose t_1 ” in π and “at (F', H') when player I has chosen t_1 choose t_2 ” becomes “at (E', H') when player I has chosen $E' \xrightarrow{a} E''$ choose t_2 ”.

When two processes E and F are game equivalent, player II can always match player I's moves: if $E \xrightarrow{a} E'$ then there is a corresponding transition $F \xrightarrow{a} F'$ and E' and F' are also game equivalent, and if $F \xrightarrow{a} F'$ then there is also a corresponding transition $E \xrightarrow{a} E'$ with E' and F' game equivalent. This is precisely the criterion for being a *bisimulation* relation. Bisimulations were introduced¹ by Park [29] as a small refinement of the equivalence defined by Hennessy and Milner in [16].

¹They also occur in a slightly different form in the theory of modal logic as zig-zag relations, see [3].

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 $(E', F') \in \mathcal{R}$.

A binary relation between processes counts as a bisimulation provided that it obeys the two hereditary conditions in this definition. Simple examples of bisimulations are the identity relation and the empty relation. Not all binary relations between processes are bisimulations.

Example 4 Assume that Cl_1 , Cl_2 , and Cl_5 are the clocks defined previously. The relation $\{(Cl_1, Cl_2), (Cl_1, tick.Cl_2)\}$ is a bisimulation. However the relation $\{(Cl_1, Cl_5)\}$ is *not* a bisimulation because of the transition $Cl_5 \xrightarrow{tick} 0$. Adding the pair $(Cl_1, 0)$ does not rectify this. The transition $Cl_1 \xrightarrow{tick} Cl_1$ can not be matched by the process 0 . \square

Two processes E and F are *bisimulation equivalent* (or *bisimilar*), written $E \sim F$, if there is a bisimulation relation \mathcal{R} with $(E, F) \in \mathcal{R}$. Bisimulation and game equivalence coincide.

Proposition 2 E is game equivalent to F iff $E \sim F$.

Proof: Assume that E is game equivalent to F . We show that $E \sim F$ by establishing that the relation $\mathcal{R} = \{(E, F) : E \text{ and } F \text{ are game equivalent}\}$ is a bisimulation. Suppose $E \xrightarrow{a} E'$, and as this is a possible move by player I we know that player II can respond with $F \xrightarrow{a} F'$ in such a way that $(E', F') \in \mathcal{R}$, and similarly when $F \xrightarrow{a} F'$. For the other direction suppose $E \sim F$, and so there is a bisimulation relation \mathcal{R} such that $(E, F) \in \mathcal{R}$. We construct a winning strategy for player II for the game $\mathcal{G}(E, F)$: in any play, whatever move player I makes player II responds by making sure that the resulting pair of processes remain in the relation \mathcal{R} . Clearly player I cannot then win any play from $\mathcal{G}(E, F)$. \square

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. This offers a very straightforward proof technique for bisimilarity.

Example 5 The two processes Cnt and Ct'_0 are bisimilar where

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

A bisimulation containing the pair $(\text{Cnt}, \text{Ct}'_0)$ has to be infinite because these processes are infinite state. Let \mathcal{C}_i be the following families of processes for $i \geq 0$ (when brackets are dropped between parallel components):

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

where $F \mid 0^0 = F$ and $F \mid 0^{i+1} = F \mid 0^i \mid 0$. The following relation $\mathcal{R} = \{(E, \text{Ct}'_i) : i \geq 0 \text{ and } E \in \mathcal{C}_i\}$ is a bisimulation which contains $(\text{Cnt}, \text{Ct}'_0)$. The proof that \mathcal{R} is a bisimulation proceeds by case analysis. First when $i = 0$, the pair $(\text{Cnt} \mid 0^j, \text{Ct}'_0) \in \mathcal{R}$. As $\text{Cnt} \xrightarrow{\text{up}} \text{Cnt} \mid \text{down}.0$ consequently $\text{Cnt} \mid 0^j \xrightarrow{\text{up}} \text{Cnt} \mid \text{down}.0 \mid 0^j$ and $\text{Ct}'_0 \xrightarrow{\text{up}} \text{Ct}'_1$. However $\text{Cnt} \mid \text{down}.0 \mid 0^j \in \mathcal{C}_1$ and so these resultant processes belong to \mathcal{R} . The case $i > 0$ is left as an exercise. \square

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 its set of traces. For full CCS the question whether two processes are bisimilar is undecidable. Turing machines can be modelled in CCS. Let 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 $\text{TM}_n \sim \text{Div}$ where $\text{Div} \stackrel{\text{def}}{=} \tau.\text{Div}$. However an interesting question is for which subclasses of processes it is decidable. Clearly this is the case for finite state processes, as there are only finitely many candidates for being a bisimulation. Surprisingly it is also decidable for families of infinite state processes including context-free processes, pushdown processes and basic parallel processes [10, 33, 9].

3.2 Modal logic and bisimulations

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

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

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 $E \not\models \Phi$.

$$\begin{array}{ll} E \models \text{tt} & E \not\models \text{ff} \\ E \models \Phi \wedge \Psi & \text{iff } E \models \Phi \text{ and } E \models \Psi \\ E \models \Phi \vee \Psi & \text{iff } E \models \Phi \text{ or } E \models \Psi \\ E \models [K]\Phi & \text{iff } \forall F. \forall a \in K. \text{if } E \xrightarrow{a} F \text{ then } F \models \Phi \\ E \models \langle K \rangle \Phi & \text{iff } \exists F. \exists a \in K. E \xrightarrow{a} F \text{ and } F \models \Phi \end{array}$$

This modal logic slightly generalises Hennessy-Milner logic [16] as sets of actions instead of single actions appear in the modalities. To reduce the number of brackets in modalities we write $[a_1, \dots, a_n]$ and $\langle a_1, \dots, a_n \rangle$ instead of $\{\{a_1, \dots, a_n\}\}$ and $\{\{a_1, \dots, a_n\}\}$.

The simple modal formula $\langle K \rangle \text{tt}$ expresses a capability for performing an action from K . In contrast, $[K] \text{ff}$ expresses an inability to initially perform an action in K . In the case of the vending machine Ven of section 2 a button can not be depressed (before money is deposited), so $\text{Ven} \models [\text{big}, \text{little}] \text{ff}$.

Example 1 Other interesting properties of Ven are:

- $\text{Ven} \models [2\text{p}]([\text{little}] \text{ff} \wedge \langle \text{big} \rangle \text{tt})$: after 2p is deposited the little button cannot be depressed whereas the big one can.
- $\text{Ven} \models [1\text{p}, 2\text{p}][1\text{p}, 2\text{p}] \text{ff}$: after a coin is entrusted no other coin (2p or 1p) may be deposited.
- $\text{Ven} \models [1\text{p}, 2\text{p}][\text{big}, \text{little}] \langle \text{collect}_b, \text{collect}_1 \rangle \text{tt}$: after a coin is deposited and a button is depressed, an item can be collected. \square

We let $\neg K$ abbreviate the set $\mathcal{A} - K$, and we write $\neg a_1, \dots, a_n$ within modalities for $\neg\{a_1, \dots, a_n\}$. Moreover we assume that \neg abbreviates the set

$\neg\emptyset$ (which is just \mathcal{A}). Consequently a process E has the property $[-]\Phi$ when each F in the set $\{E' : E \xrightarrow{a} E' \text{ and } a \in \mathcal{A}\}$ has the feature Φ . The modal formula $[-]ff$ expresses deadlock. Within this modal logic we can also express immediate necessity or inevitability. The property that only a can be performed, that it must be the next action, is given by the formula $\langle a \rangle tt \wedge [-a]ff$. After $2p$ is deposited ven must perform big , and so $ven \models [2p](\langle a \rangle tt \wedge [-big]ff)$.

Proposition 1 *If $E \sim F$ then for all $\Phi \in \mathcal{M}$. ($E \models \Phi$ iff $F \models \Phi$).*

Proof: By induction on modal formulas Φ , we show that for any G and H if $G \sim H$ then $G \models \Phi$ iff $H \models \Phi$. The base case, when Φ is either tt or ff , is clear. For the inductive step first suppose Φ is $\Psi_1 \wedge \Psi_2$ and that the result holds for the components Ψ_1 and Ψ_2 . By the definition of the satisfaction relation $G \models \Phi$ iff $G \models \Psi_1$ and $G \models \Psi_2$ iff by the inductive hypothesis $H \models \Psi_1$ and $H \models \Psi_2$ and therefore iff $H \models \Phi$. A similar argument justifies the case when Φ is $\Psi_1 \vee \Psi_2$. Next suppose Φ is the formula $[K]\Psi$ and $G \models \Phi$. Therefore for any G' such that $G \xrightarrow{a} G'$ and $a \in K$ it follows that $G' \models \Psi$. To show that $H \models \Phi$, let $H \xrightarrow{a} H'$ (with $a \in K$). However we know that for some G' there is the transition $G \xrightarrow{a} G'$ and $G' \sim H'$, and so by the induction hypothesis $H' \models \Psi$, and so $H \models \Phi$. The case when H has Φ is symmetric. The final case, when Φ is $\langle K \rangle \Psi$, is similar to the other modal case. \square

The converse of Proposition 1 holds for a restricted set of processes. A process E is immediately image finite if for each $a \in \mathcal{A}$ the set $\{F : E \xrightarrow{a} F\}$ is finite. And E is *image finite* if every member of $\{F : \exists w \in \mathcal{A}^*. E \xrightarrow{w} F\}$ is immediately image finite (where $E \xrightarrow{w} F$ is defined in the obvious way).

Proposition 2 *If E and F are image finite and for all $\Phi \in \mathcal{M}$. ($E \models \Phi$ iff $F \models \Phi$), then $E \sim F$.*

Proof: Let $G \equiv_M H$ if G and H have exactly the same \mathcal{M} properties, $\{\Phi : G \models \Phi\} = \{\Phi : H \models \Phi\}$. The relation $\{(E, F) : E \equiv_M F \text{ and } E, F \text{ are image finite}\}$ is a bisimulation. Suppose not. Without loss of generality this means that $G \equiv_M H$ for some G and H , and $G \xrightarrow{a} G'$ for some a and G' , but $G' \not\equiv_M H'$ for all H' such that $H \xrightarrow{a} H'$. There are two possibilities. First the set $\{H' : H \xrightarrow{a} H'\}$ is empty. However then $G \models \langle a \rangle tt$ (because $G \xrightarrow{a} G'$) and $H \not\models \langle a \rangle tt$ and this contradicts that $G \equiv_M H$. Otherwise the set

$\{H' : H \xrightarrow{a} H'\}$ is $\{H_1, \dots, H_n\}$ which is finite by the image finiteness condition. Therefore $G' \not\equiv_M H_i$ for each $i : 1 \leq i \leq n$, and so there are formulas Φ_1, \dots, Φ_n such that $G' \models \Phi_i$ and $H_i \not\models \Phi_i$. (Here we are utilizing the fact that M is closed under complement, see Section 4.2.) Let Ψ be the formula $\Phi_1 \wedge \dots \wedge \Phi_n$. Consequently $G' \models \Psi$, but $H_i \not\models \Psi$ for each i , as it fails the i th component of the conjunction. However $G \models \langle a \rangle \Psi$ but H can not have this property, which contradicts that $G \equiv_M H$. \square

These two results are known as the modal *characterization* of bisimulation equivalence, due to Hennessy and Milner [16].

Example 2 The need for the image finiteness condition in Proposition 2 is illustrated by the following example. Consider the family of clocks Cl_i described earlier, and $clock$. Let $clock'$ be $clock + Cl$. The processes $clock$ and $clock'$ are not bisimilar because the transition $clock' \xrightarrow{tick} Cl$ would then have to be matched by $clock \xrightarrow{tick} Cl_j$ for some $j \geq 1$, and clearly $Cl \not\sim Cl_j$. On the other hand $clock$ and $clock'$ have the same modal properties. \square

There is an unrestricted characterisation result for infinitary modal logic, M_∞ , given as follows where I ranges over arbitrary finite and infinite indexing families:

$$\Phi ::= \bigwedge\{\Phi_i : i \in I\} \mid \bigvee\{\Phi_i : i \in I\} \mid [K]\Phi \mid \langle K \rangle \Phi$$

The satisfaction relation between processes and \bigwedge and \bigvee formulas is defined as expected

$$\begin{aligned} E \models \bigwedge\{\Phi_i : i \in I\} & \text{ iff } E \models \Phi_j \text{ for every } j \in I \\ E \models \bigvee\{\Phi_i : i \in I\} & \text{ iff } E \models \Phi_j \text{ for some } j \in I \end{aligned}$$

The atomic formula tt is defined as $\bigwedge\{\Phi_i : i \in \emptyset\}$ and ff as $\bigvee\{\Phi_i : i \in \emptyset\}$.

Proposition 3 $E \sim F$ iff for all $\Phi \in M_\infty$. ($E \models \Phi$ iff $F \models \Phi$).

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, $\mu\mathcal{M}$, given in positive form are defined as follows

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

where Z ranges over a family of propositional variables, and K over subsets of \mathcal{A} . (In fact, it is a slight generalisation of Kozen's logic as sets of actions instead of single actions appear in modalities.) 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 below stipulates when a process E has the property Φ relative to \mathcal{V} , written $E \models_{\mathcal{V}} \Phi$.

$$\begin{aligned}
E \models_{\mathcal{V}} \text{tt} & \quad E \not\models_{\mathcal{V}} \text{ff} \\
E \models_{\mathcal{V}} Z & \quad \text{iff } E \in \mathcal{V}(Z) \\
E \models_{\mathcal{V}} \Phi \wedge \Psi & \quad \text{iff } E \models_{\mathcal{V}} \Phi \text{ and } E \models_{\mathcal{V}} \Psi \\
E \models_{\mathcal{V}} \Phi \vee \Psi & \quad \text{iff } E \models_{\mathcal{V}} \Phi \text{ or } E \models_{\mathcal{V}} \Psi \\
E \models_{\mathcal{V}} [K]\Phi & \quad \text{iff } \forall F, \forall a \in K. \text{ if } E \xrightarrow{a} F \text{ then } F \models_{\mathcal{V}} \Phi \\
E \models_{\mathcal{V}} \langle K \rangle \Phi & \quad \text{iff } \exists F, \exists a \in K. E \xrightarrow{a} F \text{ and } F \models_{\mathcal{V}} \Phi \\
E \models_{\mathcal{V}} \nu Z. \Phi & \quad \text{iff } \exists \mathcal{E} \subseteq \mathcal{P}(E), E \in \mathcal{E} \text{ and } \forall F \in \mathcal{E}. F \models_{\mathcal{V}[\mathcal{E}/Z]} \Phi \\
E \models_{\mathcal{V}} \mu Z. \Phi & \quad \text{iff } \forall \mathcal{E} \subseteq \mathcal{P}(E). \text{ if } E \notin \mathcal{E} \text{ then } \exists F \in \mathcal{P}(E), F \notin \mathcal{E} \text{ and } F \models_{\mathcal{V}[\mathcal{E}/Z]} \Phi
\end{aligned}$$

The stipulations for the fixed points follow directly from the Tarski-Knaster theorem, as a greatest fixed point is the union of all postfixes and a least fixed point is the intersection of all prefixes².

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*. A proof of this can be found in [35].

Proposition 1 *If $E \models \Phi$ and Φ is closed then there is a finite state process F such that $F \models \Phi$.*

Modal mu-calculus is a very powerful temporal logic which permits expression of a very rich class of properties. We briefly examine how to express a range of properties that pick out important features of processes.

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 $\nu Z. \Phi \wedge [-]Z$ expresses safety.

Example 1 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 $(\langle \overline{\text{tcross}} \rangle \text{tt} \wedge \langle \overline{\text{ccross}} \rangle \text{tt})$ captures the bad states.

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

Therefore the required safety property is $\forall Z. (\overline{tcross}ff \vee \overline{ccross}ff) \wedge [-]Z$. \square

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

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 $\mu Z. \Phi \vee (\langle - \rangle tt \wedge [-]Z)$ expresses liveness with respect to state. Note the presence of $\langle - \rangle tt$ to ensure that Φ does become true. In contrast that eventually some action in K happens is expressed by the formula $\mu Z. \langle - \rangle tt \wedge [-K]Z$.

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. This is expressed by the formula $\forall Z. [a](\mu Y. \langle - \rangle tt \wedge [-b]Y) \wedge [-]Z$. More complex is the expression of liveness properties under fairness. An example is the property that in any run if the actions b and c happen infinitely often then so does a which is expressed as follows:

$$\forall Z. (\mu X. [b](\forall Y. [c](\forall Y_1. X \wedge [-a]Y_1) \wedge [-a]Y) \wedge [-]Z)$$

Here there is an essential fixed point dependency, as the occurrence of X is free within the fixed point subformula prefaced with $\forall Y$.

Example 2 A desirable liveness property for `Crossing` that whenever a car approaches the crossing eventually it crosses is captured as

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

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 `green. \overline{tcross} . \overline{red} .Rail` (the states E_2, E_3, E_6 and E_{10} of Figure 3) and R holds when it is in any state where `Road` has the form `up. \overline{ccross} . \overline{down} .Road` (the states E_1, E_3, E_7 and E_{11}). 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 $\forall Y. [\text{car}](\Phi_1 \wedge [-]Y)$ relative to \mathcal{V} where Φ_1 is

$$\mu X. \forall Y_1. (Q \vee [\overline{\text{ccross}}](\forall Y_2. (R \vee X) \wedge [\overline{\text{ccross}}]Y_2)) \wedge [\overline{\text{ccross}}]Y_1$$

□

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. The formula $\mu Y. \Psi \vee (\Phi \wedge \langle - \rangle \text{tt} \wedge [-]Y)$ expresses that Φ holds until Ψ in every run. Note here the requirement that Ψ does eventually become true. This commitment can be removed by changing fixed points.

Cyclic properties can also be described in the logic. A simple example is that each even action is `tock`: if $E_0 \xrightarrow{a_1} E_1 \xrightarrow{a_2} \dots$ is a finite or infinite length run then each action a_{2i} is `tock`, $\forall Z. [-]([\text{tock}]ff \wedge [-]Z)$. The clock $\text{Cl}_1 \stackrel{\text{def}}{=} \text{tick.tock.Cl}_1$ has this property. It also has the tighter cyclic property that every run involves the repeated cycling of `tick` and `tock` actions, expressed as $\forall Z. [\text{tick}]ff \wedge [\text{tick}](\text{tock}ff \wedge [-]Z)$ ³. 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:

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

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

As these examples show, 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 on infinite binary trees it 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

³This formula leaves open the possibility that a run has finite length. To preclude it one adds $\langle - \rangle \text{tt}$ at the outer and inner level.

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 2 does not contain essential alternation as the subformula $\mu Y. \langle - \rangle \text{tt} \wedge [\overline{\text{cross}}]Y$ 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:

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

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. Recently Bradfield has shown 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 the extended modal logic M_∞ . 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:

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

A simple consequence is the following pair

⁴If 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 .

$$\begin{aligned} E \models_{\mathcal{V}} \forall Z. \Phi & \text{ iff } E \models_{\mathcal{V}} \forall Z^{\alpha}. \Phi \text{ for all } \alpha. \\ E \models_{\mathcal{V}} \mu Z. \Phi & \text{ iff } E \models_{\mathcal{V}} \mu Z^{\alpha}. \Phi \text{ for some } \alpha. \end{aligned}$$

As μM contains M and is contained in M_{∞} the following is a consequence of the results in the previous section.

Proposition 2 *If $E \sim F$ then for all closed $\Phi \in \mu M$. ($E \models \Phi$ iff $F \models \Phi$).*

Notice the significance of this result. Bisimilar processes not only have the same safety properties but also the same liveness, fairness, and cyclic properties when expressed using closed formulas.

3.4 Second-order propositional modal logic

We contrast second-order propositional modal logic, $2M$, with modal mu-calculus. $2M$, introduced in [32], is defined as an extension of μM as follows

$$\Phi ::= \text{tt} \mid Z \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid [K]\Phi \mid \square\Phi \mid \forall Z.\Phi$$

The modality \square 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: $\text{ff} \stackrel{\text{def}}{=} \neg\text{tt}$, $\Phi_1 \vee \Phi_2 \stackrel{\text{def}}{=} \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, $\Phi_1 \rightarrow \Phi_2 \stackrel{\text{def}}{=} \neg\Phi_1 \vee \Phi_2$, $\langle K \rangle \Phi \stackrel{\text{def}}{=} \neg[K]\neg\Phi$, $\diamond\Phi \stackrel{\text{def}}{=} \neg\square\neg\Phi$, and $\exists Z.\Phi \stackrel{\text{def}}{=} \neg\forall Z.\neg\Phi$.

As with modal mu-calculus we define when a process E has a property Φ relative to \mathcal{V} , written $E \models_{\mathcal{V}} \Phi$, where \mathcal{V} is a valuation. The semantic clauses for tt , Z , \wedge and $[K]$ are as in the previous section. The new clauses are:

$$\begin{aligned} E \models_{\mathcal{V}} \neg\Phi & \text{ iff } E \not\models_{\mathcal{V}} \Phi \\ E \models_{\mathcal{V}} \square\Phi & \text{ iff } \forall F \in \mathcal{P}(E). F \models_{\mathcal{V}} \Phi \\ E \models_{\mathcal{V}} \forall Z.\Phi & \text{ iff } \forall \mathcal{E} \subseteq \mathcal{P}(E). E \models_{\mathcal{V}[\mathcal{E}/Z]} \Phi \end{aligned}$$

Notice that \square is definable in μM : assuming Z is not free in Φ , the formula $\square\Phi$ is $\forall Z. \Phi \wedge [-]Z$. The operator $\forall Z$ is a set quantifier, ranging over subsets of $\mathcal{P}(E)$.

An important issue is the relationship between closed formulas of μM and $2M$, with respect to particular families of models. Within $2M$ we can define 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:

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

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]).

Proposition 1 $\mu\mathcal{M}$ is a sublogic of $2\mathcal{M}$.

Proof: There is a straightforward translation of $\mu\mathcal{M}$ into $2\mathcal{M}$. Let Tr be this translation. The important cases are the fixed points: $\text{Tr}(\nu Z. \Phi) = \exists Z. (Z \wedge \Box(Z \rightarrow \text{Tr}(\Phi)))$ and $\text{Tr}(\mu Z. \Phi) = \forall Z. (\Box(\text{Tr}(\Phi) \rightarrow Z) \rightarrow Z)$. \square

When models are restricted to binary (or n -ary, $n \geq 1$) trees, the closed formulas of $2\mathcal{M}$ are translatable into $\mu\mathcal{M}$. However this is not the case for processes, as $2\mathcal{M}$ formulas can distinguish between bisimilar processes which $\mu\mathcal{M}$ formulas are unable to do. It turns out that this is the only reason for increased expressiveness of $2\mathcal{M}$ over $\mu\mathcal{M}$. A set of processes \mathcal{E} is bisimulation closed provided that if $E \in \mathcal{E}$ and $E \sim F$ then $F \in \mathcal{E}$. Clearly (from Proposition 2 of the previous section) for any closed modal mu-calculus formula Φ the set $\{E : E \models \Phi\}$ is bisimulation closed. The following result is a corollary of [18].

Proposition 2 If $\Phi \in 2\mathcal{M}$ is closed and $\{E : E \models \Phi\}$ is bisimulation closed then there is a closed $\Psi \in \mu\mathcal{M}$ equivalent to Φ (that is, $\{E : E \models \Psi\} = \{E : E \models \Phi\}$).

Another way that we can contrast $\mu\mathcal{M}$ and $2\mathcal{M}$ is to use games that are extensions of the bisimulation game defined earlier. These extended games include moves for colouring processes, see [32].

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 tech-

niques 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 polynomial 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 space⁵. 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 $E \models_{\forall} \Phi$ one constructs the sets $\{F \in \mathcal{P}(E). F \models_{\forall} \Psi\}$ for each subformula Ψ of Φ . Local techniques, in contrast, try and directly solve whether $E \models_{\forall} \Phi$. Local methods also apply to infinite state systems, and are often presented using tableaux [34, 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.

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

4.1 Subformulas and subsumption

Here we present some useful definitions which will be used in the next section when property games are defined.

Definition 1 The size of Φ , denoted by $|\Phi|$, is defined inductively as follows:

$$\begin{aligned} |\text{tt}| &= |\text{ff}| = |Z| = 1 \\ |\Phi_1 \wedge \Phi_2| &= |\Phi_1 \vee \Phi_2| = 1 + |\Phi_1| + |\Phi_2| \\ |[K]\Phi| &= |\langle K \rangle \Phi| = |\sigma Z. \Phi| = 1 + |\Phi| \end{aligned}$$

Thus $|\Phi|$ is the number of “symbols” within Φ .

Definition 2 The set of subformulas of a modal mu-calculus formula Φ is defined inductively as $\text{Sub}(\Phi)$:

$$\begin{aligned} \text{Sub}(\text{tt}) &= \{\text{tt}\} \\ \text{Sub}(\text{ff}) &= \{\text{ff}\} \\ \text{Sub}(X) &= \{X\} \\ \text{Sub}(\Phi_1 \wedge \Phi_2) &= \{\Phi_1 \wedge \Phi_2\} \cup \text{Sub}(\Phi_1) \cup \text{Sub}(\Phi_2) \\ \text{Sub}(\Phi_1 \vee \Phi_2) &= \{\Phi_1 \vee \Phi_2\} \cup \text{Sub}(\Phi_1) \cup \text{Sub}(\Phi_2) \\ \text{Sub}([K]\Phi) &= \{[K]\Phi\} \cup \text{Sub}(\Phi) \\ \text{Sub}(\langle K \rangle \Phi) &= \{\langle K \rangle \Phi\} \cup \text{Sub}(\Phi) \\ \text{Sub}(\nu Z. \Phi) &= \{\nu Z. \Phi\} \cup \text{Sub}(\Phi) \\ \text{Sub}(\mu Z. \Phi) &= \{\mu Z. \Phi\} \cup \text{Sub}(\Phi) \end{aligned}$$

Definition 3 A formula Φ is *normal* if

1. every occurrence of a binder σZ in Φ binds a distinct variable, and
2. 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, for instance $Z \vee \nu Z. ((\mu Z. \Phi) \wedge [-]Z)$ becomes $Z \vee \nu X. ((\mu Y. \Phi\{Y/Z\}) \wedge [-]X)$ where X and Y do not occur in Φ . If $\sigma Z. \Phi$ is a subformula of a normal formula Ψ then we can use the binding variable Z to uniquely identify this subformula.

Definition 4 Assume Φ is normal and that $\sigma X. \Psi, \sigma Z. \Psi' \in \text{Sub}(\Phi)$. The variable X *subsumes* Z iff $\sigma Z. \Psi' \in \text{Sub}(\sigma X. \Psi)$.

Proposition 1

1. X subsumes X , and if X subsumes Z and Z subsumes Y then X subsumes Y ,
2. If $\sigma X.\Psi, \sigma Z.\Psi' \in \text{Sub}(\Phi)$ and X subsumes Z and $X \neq Z$ then $|\sigma X.\Psi| > |\sigma Z.\Psi'|$.

4.2 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 turn⁶.

A play of $\mathcal{G}_{\mathcal{V}}(E_0, \Phi_0)$ is a finite or infinite length sequence of the form $(E_0, \Phi_0) \dots (E_n, \Phi_n) \dots$ where each $\Phi_i \in \text{Sub}(\Phi_0)$ and each $E_i \in \mathcal{P}(E_0)$. Suppose part of a play is $(E_0, \Phi_0) \dots (E_j, \Phi_j)$. The next move and which player makes it depends on the formula Φ_j ; the moves are given in Figure 4. Note the duality between the rules for \wedge and \vee , and $[K]$ and $\langle K \rangle$. In the rules for fixed point formulas we use the fact that the starting formula Φ_0 is normal, and so each fixed point subformula is uniquely identified by its bound variable. 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 it identifies is, in effect, unfolded once as the formula becomes Ψ ⁷.

The conditions for winning a play are given in Figure 5. A player wins if her opponent is stuck (condition 2 for both players). Player I wins if a blatantly false configuration is reached, (E_n, ff) or (E_n, Z) where Z is free in Φ_0 and $E \not\equiv_{\mathcal{V}} Z$. Similarly 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

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

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

- if $\Phi_j = \Psi_1 \wedge \Psi_2$ then player I chooses one of the conjuncts $\Psi_i, i \in \{1, 2\}$: 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 $\Psi_i, i \in \{1, 2\}$: 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 .

Figure 4: Rules for the next move in a game play

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 $\sigma X_1 \sigma X_2 \dots \sigma X_n. \Phi(X_1, \dots, X_n)$, 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.

Lemma 1 *If $(E_0, \Phi_0) \dots (E_n, \Phi_n) \dots$ is an infinite length play of $\mathcal{G}_V(E_0, \Phi_0)$ then there is a unique variable X which*

1. *occurs infinitely often (for infinitely many j , $X = \Phi_j$) and*
2. *if Y also occurs infinitely often then X subsumes Y .*

Proof: Let $\sigma X_1. \Psi_1, \dots, \sigma X_n. \Psi_n$ be all the fixed point subformulas in $\text{Sub}(\Phi_0)$ in decreasing order of size: if $i < j$ then $|\sigma X_i. \Psi_i| \geq |\sigma X_j. \Psi_j|$, and hence X_j does not subsume X_i . Consider the next move in a play, from (E_j, Φ_j) to (E_{j+1}, Φ_{j+1}) : if Φ_j is not a variable then $|\Phi_{j+1}| < |\Phi_j|$. As each subformula has finite size, an infinite length play must proceed infinitely often

Player I wins

1. The play is $(E_0, \Phi_0) \dots (E_n, \Phi_n)$ and either $\Phi_n = \text{ff}$ 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 $\Phi_n = \text{tt}$ 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 $\nu X. \Psi$.

Figure 5: Winning conditions

through variables belonging to $\{X_1, \dots, X_n\}$. Hence there is at least one variable which occurs infinitely often. If a subpart of the play has the form $(E_n, X_i) \dots (E_k, X_j) \dots (E_m, X_i) \dots (E_l, X_j)$ and $X_i \neq X_j$ then either X_i subsumes X_j or X_j subsumes X_i , but not both. Consequently, by transitivity of subsumption, there is exactly one variable X_i which occurs infinitely often and which subsumes any other X_j which also occurs infinitely often. \square

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:

at position $(E, \Phi_1 \wedge \Phi_2)$ choose (E, Φ_i)
 at position $(E, [K]\Phi)$ choose (F, Φ)

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

at position $(E, \Phi_1 \vee \Phi_2)$ choose (E, Φ_i)
 at position $(E, \langle K \rangle \Phi)$ choose (F, Φ)

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 Player II wins the game $\mathcal{G}(Cl, \nu Z. \langle tick \rangle Z)$ ⁸. There is just one play which has infinite length:

$$(Cl, \nu Z. \langle tick \rangle Z) (Cl, Z) (Cl, \langle tick \rangle Z) (Cl, Z) \dots$$

As Z is the only fixed point variable which occurs infinitely often and it identifies a maximal fixed point formula player II wins. The winning strategy is, in effect, the empty set of rules as player II has to make the move (Cl, Z) from position $(Cl, \langle tick \rangle Z)$. \square

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

Example 2 Let $D \xrightarrow{a} D'$, $D' \xrightarrow{a} D$ and $D' \xrightarrow{b} D''$, and let Ψ be the formula:

$$\mu Y. \nu Z. [a](((\langle b \rangle tt \vee Y) \wedge Z))$$

D' (and D) fails to have the property Ψ , and so player I has a winning strategy for $\mathcal{G}(D', \Psi)$. The important rules are:

$$\begin{aligned} &\text{at } (D, ((\langle b \rangle tt \vee Y) \wedge Z)) \text{ choose } (D, \langle b \rangle tt \vee Y) \\ &\text{at } (D', ((\langle b \rangle tt \vee Y) \wedge Z)) \text{ choose } (D', Z) \end{aligned}$$

The play proceeds:

$$\begin{aligned} &(D', \Psi) (D', \nu Z. [a](((\langle b \rangle tt \vee Y) \wedge Z))) (D', [a](((\langle b \rangle tt \vee Y) \wedge Z))) \\ &\quad (D, ((\langle b \rangle tt \vee Y) \wedge Z)) (D, \langle b \rangle tt \vee Y) \end{aligned}$$

Player II has a choice here, however if she chooses $(D, \langle b \rangle tt)$ then she loses immediately. Otherwise the play continues:

$$\begin{aligned} &(D, Y) (D, \nu Z. [a](((\langle b \rangle tt \vee Y) \wedge Z))) (D, [a](((\langle b \rangle tt \vee Y) \wedge Z))) \\ &\quad (D', ((\langle b \rangle tt \vee Y) \wedge Z)) (D', Z) (D', [a](((\langle b \rangle tt \vee Y) \wedge Z))) \dots \end{aligned}$$

There is now a repeating cycle. Let the play cycle forever. Although the two variables Z and Y occur infinitely often in the play, player I wins because Y subsumes Z (but not vice versa), and Y abbreviates a least fixed point formula. \square

Note that the general problem of property checking is closed under complement. For each formula Φ define Φ^c as follows:

$$\begin{aligned} tt^c &= ff \quad ff^c = tt \quad Z^c = Z \\ (\Phi \wedge \Psi)^c &= \Phi^c \vee \Psi^c \quad (\Phi \vee \Psi)^c = \Phi^c \wedge \Psi^c \\ (\nu Z. \Phi)^c &= \mu Z. (\Phi)^c \quad (\mu Z. \Phi)^c = \nu Z. (\Phi)^c \end{aligned}$$

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 strategy 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).

4.3 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 $\text{TM}_n \models \forall Z. \langle - \rangle \text{tt} \wedge [-]Z$ where TM_n is the coding of the n th 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 $\mathcal{G}_V(E, \Phi)$ is the directed graph representing all possible plays of $\mathcal{G}_V(E, \Phi)$. The vertices are configurations of a possible play, and have the form (F, Ψ) where $F \in \mathcal{P}(E)$ and $\Psi \in \text{Sub}(\Phi)$. There is a directed edge between two vertices $v_1 \rightarrow v_2$ 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 $\mathcal{G}_V(E, \Phi)$ is therefore bounded by $|\mathcal{P}(E)| \times |\text{Sub}(\Phi)|$. The model checking decision question (in terms of games) is: does player II have a winning strategy for $\mathcal{G}_V(E, \Phi)$?

Model checking can be abstracted into the following simple graph game, which we call the MC game. An MC game is a directed graph $\mathcal{G} = (V, \text{Ed}, L)$ whose vertices $V = \{1, \dots, n\}$ and whose edges $\text{Ed} \subseteq V \times V$ and where $L : V \rightarrow \{I, II\}$ labels each vertex with I or with II. Each vertex $i \in V$ has at least one edge $i \rightarrow j \in \text{Ed}$, writing $i \rightarrow j$ 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.

Proposition 1 *Every model checking game determines an equivalent MC game.*

Proof: Let $\mathcal{G}_V(E, \Phi)$ be the model checking game for the finite-state process E and the normal formula Φ . Let E_1, \dots, E_m be an enumeration of all processes in $\mathcal{P}(E)$ with $E = E_1$. Assume that Z_1, \dots, 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 Φ_1, \dots, Φ_l be an enumeration of all formulas in $\text{Sub}(\Phi) - \{Z_1, \dots, Z_k\}$ in de-

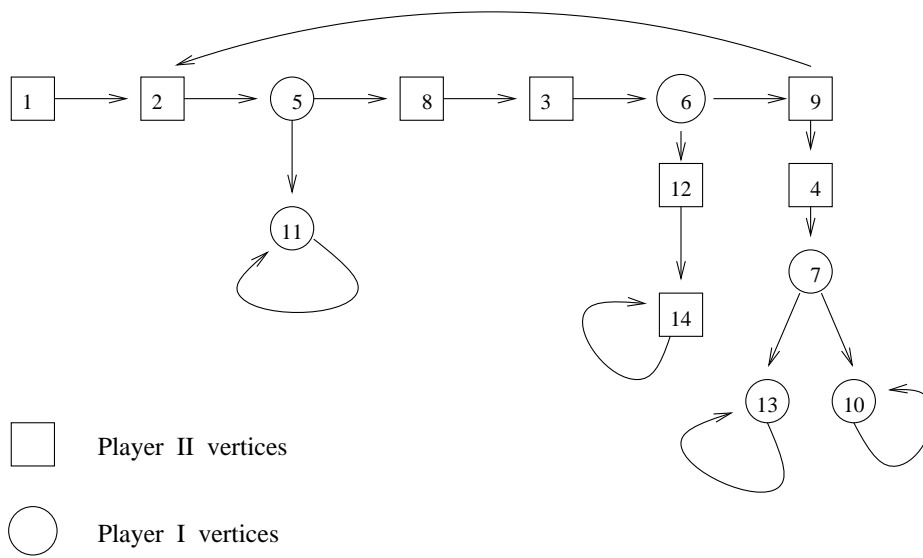
creasing order of size (and so $\Phi = \Phi_1$). Insert each Z_i directly after the fixed point it identifies. The result is a sequence of formulas Φ_1, \dots, Φ_n in decreasing order of size, except that Z_i counts as bigger than Ψ_i . The vertices of the MC game in order are

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

and so $V = \{1, \dots, nm\}$ 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 $L(i) = \text{II}$ and there is the edge (i, i) . If instead $F \notin \mathcal{V}(Z)$ then $L(i) = \text{I}$ and there is the edge (i, i) . If Ψ is tt then $L(i) = \text{II}$ and there is the edge (i, i) . If Ψ is ff then $L(i) = \text{I}$ and there is the edge (i, i) . If Ψ is $\Psi_1 \wedge \Psi_2$ then $L(i) = \text{I}$ and there are the edges (i, j_1) and (i, j_2) where j_1 describes (F, Ψ_1) and j_2 describes (F, Ψ_2) . Dually, if Ψ is $\Psi_1 \vee \Psi_2$ then $L(i) = \text{II}$ and there are the edges (i, j_1) and (i, j_2) where j_1 describes (F, Ψ_1) and j_2 describes (F, Ψ_2) . If Ψ is $[K]\Psi'$ and there are no K transitions from F then $L(i) = \text{II}$ and there is the edge (i, i) . Dually, if Ψ is $\langle K \rangle \Psi'$ and there are no K transitions from F then $L(i) = \text{I}$ and there is the edge (i, i) . If Ψ is $[K]\Psi'$ and there are K transitions from E then $L(i) = \text{I}$ 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 $L(i) = \text{II}$ 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 $L(i) = \text{I}$ 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 $L(i) = \text{II}$ 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.

Having defined the game, we leave it as an exercise that player II has a winning strategy for $\mathcal{G}_V(E, \Phi)$ iff player II has a winning strategy for the determined MC game. (Game positions involving outermost fixed points appear earlier in the ordering of vertices.) \square

Example 1 The game $\mathcal{G}(\text{D}, \nu Z. \langle b \rangle \text{tt} \wedge \langle - \rangle Z)$, where D is from Example 2 of the previous section determines the MC game of Figure 6. The MC representation of positions from the model checking game are also presented.



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

Figure 6: Game

To make game playing perpetual note the loops on 11, 14, 13 and 10. \square

There is a converse to Proposition 1: 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 $\mathcal{G} = (V, \text{Ed}, L)$ and $X \subseteq V$ we let $\mathcal{G} - X$ be the structure $\mathcal{G}' = (V', \text{Ed}', L')$ where $V' = V - X$, Ed' is the set of edges $\text{Ed} \cap (V' \times V')$ 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 $i \rightarrow j \in \text{Ed}'$ 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, 37]) the set of vertices for which a player P can force play to enter a subset X of vertices, as $\text{Force}_P(X)$.

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

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

Consequently, if $j \in \text{Force}_P(X)$ 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 $j \in \text{Force}_P^i(X)$: the rank is (an upper bound on) the number of moves it takes to force the play into X .

Example 2 Consider the following force set, where the vertices are from Example 1:

$$\begin{aligned}
\text{Force}_{\text{II}}^0(\{13, 10\}) &= \{13, 10\} \\
\text{Force}_{\text{II}}^1(\{13, 10\}) &= \{13, 10\} \cup \{7\} \\
\text{Force}_{\text{II}}^2(\{13, 10\}) &= \{13, 10, 7\} \cup \{4\} \\
\text{Force}_{\text{II}}^3(\{13, 10\}) &= \{13, 10, 7, 4\} \cup \{9\} \\
\text{Force}_{\text{II}}^4(\{13, 10\}) &= \{13, 10, 7, 4, 9\} \cup \emptyset
\end{aligned}$$

So $\text{Force}_{\text{II}}(\{13, 10\}) = \{13, 10, 7, 4, 9\}$, and 9 has rank 3. The different set $\text{Force}_{\text{I}}(\{13, 10\}) = \{13, 10, 7, 4\}$. \square

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

Proposition 2 *If $\mathcal{G} = (V, \text{Ed}, L)$ and $X \subseteq V$ then either $\text{Force}_{\text{P}}(X) = V$ or $\mathcal{G}' = \mathcal{G} - \text{Force}_{\text{P}}(X)$ is a subgame.*

Proof: Assume that $X \subseteq V$, and that $\text{Force}_{\text{P}}(X) \neq V$. Consider the structure $\mathcal{G}' = \mathcal{G} - \text{Force}_{\text{P}}(X)$. \mathcal{G}' fails to be a game if there is a vertex j in V' such that there is no edge $j \rightarrow k \in \text{Ed}'$. Consider any such vertex j . Clearly each k such that $j \rightarrow k \in \text{Ed}$ belongs to $\text{Force}_{\text{P}}(X)$, and so there is a least index i such that $k \in \text{Force}_{\text{P}}^i(X)$ for each such k . But then $j \in \text{Force}_{\text{P}}^{i+1}(X)$, and therefore $j \in \text{Force}_{\text{P}}(X)$, which contradicts that it is in \mathcal{G}' . \square

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 $i \rightarrow j$.

Proposition 3 *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 $\mathcal{G} = (V, \text{Ed}, L)$ 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 $i \rightarrow i \in \text{Ed}$. 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 j_1 such that $k \rightarrow j_1 \in \text{Ed}$. More precisely, the strategy consists of the rule at k choose j_1 where j_1 has a least rank in $\text{Force}_{L(k)}(\{k\})$ among the set $\{j : k \rightarrow j \in \text{Ed}\}$, and for any other $l \in X$ such that $L(l) = L(k)$ choose the edge $l \rightarrow l_1$ such that l_1 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 $Y = \{j : k \rightarrow j \in \text{Ed}\}$.

Case 1: $Y \cap (W_{L(k)} \cup X) \neq \emptyset$. There is an edge $k \rightarrow j_1 \in \text{Ed}$ and $j_1 \in X$ or $j_1 \in W_{L(k)}$. Player $L(k)$ now has a history free winning strategy for $\mathcal{G}(i)$ for each $i \in (X \cup W_{L(k)})$. Let σ' be the substrategy for P which forces any play in X to k together with the rule at k choose j_1 (determined as earlier). If $j_1 \in X$ 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 $j_1 \in W_{L(k)}$ then the strategy $\sigma' \cup \sigma_{j_1}$ is winning for $\mathcal{G}(i)$ for $i \in X \cup U$ where U consists of vertices visited in $W_{L(k)}$ using σ_{j_1} . For other vertices j in $W_{L(k)}$ the history free winning strategy gives priority to the rules $\sigma' \cup \sigma_{j_1}$ 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 j_1 such that $k \rightarrow j_1$ the opponent O of $L(k)$ has a history free winning strategy for $\mathcal{G}'(j_1)$. 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 $\mathcal{G}'' = \mathcal{G} - Z$. 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 $\sigma''(j)$ 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. \square

Implicit in the proof of Proposition 3 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 $\mathcal{G}_V(E, \Phi)$ when Φ is an alternation free formula. An MC game is II-simple if every II vertex i (that is, $L(i) = \text{II}$) has exactly one edge. The proof of Proposition 3 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}'' .

Proposition 4 *The decision problem for MC games belongs to $\text{NP} \cap \text{co-NP}$.*

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 $L'(i) = \text{I}$ iff $L(i) = \text{II}$, 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. \square

Emerson, Jutla and Sistla showed that the model checking decision problem belongs to $\text{NP} \cap \text{co-NP}$ using automata theoretic techniques [15]. An alternative proof of Proposition 4 is sketched in the next section.

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:

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

Satisfiability (or really truth) checking of closed formulas of this logic is therefore also in $\text{NP} \cap \text{co-NP}$. 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]).

4.4 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, \dots, 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 $\mathcal{G}_V(E, \Phi)$ determines a generalised MC game (as the colours are just the elements of $\text{Sub}(\Phi)$). 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 wins⁹. By defining graph games with more elaborate winning conditions, winning strategies need no longer be history-free, see [26, 37].

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 im-

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

ply 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 $\text{NP} \cap \text{co-NP}$ [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 $j \rightarrow i$ with $j \geq i$. Each such edge $j \rightarrow i$ when $j \geq i$ is removed, and the edge $j \rightarrow i1$ 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 4 *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 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 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 Guenther, 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 pushdown 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] Park, D. (1981). Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science*, 154, 561-572.
- [30] Stirling, C. (1995). Local model checking games. *Lecture Notes in Computer Science*, **962**, 1-11.
- [31] Stirling, C. (1996). Modal and temporal logics for processes. *Lecture Notes in Computer Science*, **1043**, 149-237.
- [32] Stirling, C. (1996). Games and modal mu-calculus. *Lecture Notes in Computer Science*, **1055**, 298-312.
- [33] Stirling, C. (1996). Decidability of bisimulation equivalence for normed pushdown processes. *Lecture Notes in Computer Science*, **1119**, 217-232.
- [34] Stirling, C. and Walker, D. (1991). Local model checking in the modal mu-calculus. *Theoretical Computer Science*, **89**, 161-177.
- [35] Streett, R. and Emerson, E. (1989). An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, **81**, 249-264.
- [36] Thomas, W. (1993). On the Ehrenfeucht-Fraïssé game in theoretical computer science. *Lecture Notes in Computer Science*, **668**.
- [37] Thomas, W. (1995). On the synthesis of strategies in infinite games. *Lecture Notes in Computer Science*, **900**, 1-13.