Safety for bisimulation in monadic second-order logic

Marco Hollenberg

Department of Philosophy, Utrecht University

Heidelberglaan 8, 3584 CS Utrecht, the Netherlands

`hollenb@phil.ruu.nl`

`http://www.phil.ruu.nl/home/marco/`

November 5, 1996

Abstract

We characterize those formulas of MSO (monadic second-order logic) that are *safe for bisimulation*: formulas defining binary relations such that any bisimulation is also a bisimulation with respect to these defined relations. Every such formula is equivalent to one constructed from μ-calculus tests, atomic actions and the regular operations. The proof uses a characterization of *completely additive* μ-calculus formulas: formulas $\phi(p)$ that distribute over arbitrary unions. It turns out that complete additivity is equivalent to distributivity over countable unions.

For FOL (first-order logic) a similar theorem is shown (giving an alternative proof to the original of [4]). Here though distributivity over finite unions is sufficient. This enables us to show that the characterization of safe FOL-formulas carries over to the setting of finite models.

1 Introduction

The identification of processes with transition systems has been a fruitful one in computer science. In this perspective, states of a process are viewed as nodes of a transition system, and given some atomic nondeterministic action a on states, a transition labeled with a exists between states s_1 and s_2 if it is possible that upon execution of the action a at state s_1 we reach state s_2.

A variety of logical languages have been designed to speak about transition systems ([19, 31]). Examples include finitary and infinitary polymodal logic (the finitary variant of which is known as Hennessy-Milner Logic (HML) in the computer science literature, due to Hennessy and Milner's seminal [20])), Propositional Dynamic Logic (PDL, [29, 13]), Computation Tree Logic (CTL, [8]), CTL* ([10, 12]) and the modal μ-calculus ([24]). More standard formalisms such as first-order and higher-order languages may of course also be used to describe properties of transition systems (see [6]).

The expressive power of a formalism should be restricted by the intended application. We should not be able to express properties which distinguish between objects which, as far as the intended application is concerned, are equivalent. So if the models on which we interpret the formulas of our formalism are equipped with a natural notion of equivalence, no formula should be able to distinguish between models that are equivalent in this given sense. In the present situation a natural candidate for equivalence is *bisimulation* ([2, 28]). There are numerous other notions of equivalence, but most of these are subsumed by bisimulation (see [1, 14, 15]). So logics of transition systems should satisfy *invariance for bisimulation*: formulas should not be able to distinguish states that are bisimilar. In fact, all the systems mentioned, except full first- or higher-order logic, satisfy this constraint.

Suppose that for some reason we wish to work with a certain logic that is *too* expressive (i.e. it can distinguish between bisimilar states), yet we still take the demand of invariance for bisimulation to heart. Then we may ask which formulas of our given logic satisfy the constraint. For a variety of standard logics there is an answer to this question: the first-order logic (FOL) formulas that are invariant for bisimulation are precisely (modulo equivalence) the finitary polymodal ones, for first-order logic with infinite conjunctions and disjunctions we get infinitary polymodal logic (for the result for $\mathcal{L}_{\omega_1\omega}$ (countable disjunctions and conjunctions only) see [6], for $\mathcal{L}_{\infty\omega}$ (arbitrary disjunctions and conjunctions), see [4]), and for monadic second-order logic (MSO) we get the modal μ-calculus ([23, 21]).

Most of the logical systems given concentrate on expressing properties of states. This leaves something to be desired, as the most fundamental ingredient of a transition system consists of a family of *binary relations* between states, one for each atomic action. A formalism that does speak of relations between states is PDL: the definition of its language consists of a simultaneous recursion on *formulas*, which express unary properties of states, and *programs*, which denote binary relations between states.

Suppose we design a logic whose formulas are interpreted as binary relations between states of a transition system. Such formulas we could call *programs*: they nondeterministically take us from one state to another. Again, we may ask which programs we should consider appropriate. We take our cue from invariance for bisimulation. This property is an obvious generalization of the clause in the definition of bisimulation for unary predicates. Likewise, for programs, we could generalize the clause for binary predicates. This gives us the notion of *safety for bisimulation* ([4]). A program π is safe for bisimulation if whenever Z is a bisimulation between two transition systems with respect to the atomic actions, then this same Z is also a bisimulation with respect to the binary relation defined by π. PDL-programs are without exception safe for bisimulation.

Now consider logical languages of binary relations which can express programs which are not safe for bisimulation. Which formulas of such formalisms *are* safe for bisimulation? For FOL it has been shown that the formulas $\phi(x, y)$ that are safe for bisimulation consist precisely of modal tests $\phi?$ (with ϕ a polymodal formula) and atomic actions, closed under the operations union and composition ([4]). For $\mathcal{L}_{\infty\omega}$ we get tests $\phi?$, with ϕ an infinitary polymodal formula, and atomic actions, closed under *arbitrary* unions and composition ([5]). For $\mathcal{L}_{\omega_1\omega}$ the question is still open, although one would expect the answer to consist of restricting to countable unions. In this paper we give an answer for MSO: we prove that an MSO-formula $\phi(x, y)$ is safe for bisimulation iff it is equivalent to a program constructed from μ-calculus formula tests, atomic actions and the regular operations of union, composition and Kleene-iteration.

The paper is organized as follows. The next section introduces the main definitions and tools of this paper: invariance for bisimulation, the modal μ-calculus and μ-automata. The section after that gives the characterization of the bisimulation-safe fragment of MSO. The proof of this fact may itself be of some interest. It is a combination of a number of ideas:

1. The use of the bisimulation-invariance theorem: the fact that the bisimulation-invariant fragment of MSO is precisely the μ-calculus.

2. A model-theoretic characterization of those μ-calculus formulas that are *completely additive*, i.e. distribute over arbitrary unions.

3. The latter is proved using a method taken from the well-known proof that finite state automata correspond to regular expressions.

The first two of these items are to be found in all bisimulation-safety characterizations mentioned above, adapted to the different logics concerned of course. The last item is specific to the proof of this paper.

Next, the methods used for MSO are applied to FOL, giving an alternative proof of the bisimulation-safety theorem for this language. For this we use so-called *modal automata* (introduced in [9]) which are μ-automata that correspond to polymodal formulas (without fixed-points).

2 Invariance and the modal μ-calculus

Let us first introduce the main notions and results on which this paper builds.

Definition 2.1 (MSO) *Let $\mathcal{L}$ be a signature containing relation symbols and their arities. $\mathcal{V}_1$ and $\mathcal{V}_2$ are infinite sets of first-order and unary second-order variables respectively.*

Given these ingredients, Monadic Second-Order logic (MSO) has the following language:

$$\phi ::= \bot \mid R(x_1,\ldots,x_n) \mid x = y \mid X(x) \mid \neg\phi \mid \phi \vee \phi \mid \exists x.\phi \mid \exists X.\phi$$

where R is an n-ary relation symbol in $\mathcal{L}$, $x, y, x_1, \ldots, x_n$ are first-order variables and X is a second-order variable. Other connectives such as $\wedge$, $\rightarrow$, $\forall$ are definable as usual. □

Note that the 'M' in MSO thus indicates that quantification over monadic, or unary, predicates is allowed: formulas may contain binary relation-symbols, but these may not be quantified over. Note also that equality is assumed present in the language. This is just for convenience, as $x = y$ is definable as $\forall X.(X(x) \leftrightarrow X(y))$. Likewise $\bot$ is not necessary as a basic symbol, being definable as $\forall X.X(x)$.

MSO-formulas are interpreted in ordinary models $\mathcal{M}$ of the signature $\mathcal{L}$. We need assignment-functions both for first-order and second-order variables. Let $\sigma : \mathcal{V}_1 \rightarrow \mathcal{M}$ and $\tau : \mathcal{V}_2 \rightarrow \wp(\mathcal{M})$, where we adopt the convention of identifying $\mathcal{M}$ with its domain. The notion $\mathcal{M} \models \phi[\sigma,\tau]$ is then defined as follows:

$$\begin{array}{lll}
\mathcal{M} \models \bot[\sigma,\tau] & & \text{is never the case} \\
\mathcal{M} \models R(x_1,\ldots,x_n)[\sigma,\tau] & \text{iff} & (\sigma(x_1),\ldots,\sigma(x_n)) \in R^{\mathcal{M}} \\
\mathcal{M} \models x = y[\sigma,\tau] & \text{iff} & \sigma(x) = \sigma(y) \\
\mathcal{M} \models X(x)[\sigma,\tau] & \text{iff} & \sigma(x) \in \tau(X) \\
\mathcal{M} \models \neg\phi[\sigma,\tau] & \text{iff} & \mathcal{M} \not\models \phi[\sigma,\tau] \\
\mathcal{M} \models \phi \vee \psi[\sigma,\tau] & \text{iff} & \mathcal{M} \models \phi[\sigma,\tau] \text{ or } \mathcal{M} \models \psi[\sigma,\tau] \\
\mathcal{M} \models \exists x.\phi[\sigma,\tau] & \text{iff} & \text{there is an } s \in \mathcal{M} \text{ with } \mathcal{M} \models \phi[\sigma[x := s],\tau] \\
\mathcal{M} \models \exists X.\phi[\sigma,\tau] & \text{iff} & \text{there is an } S \subseteq \mathcal{M} \text{ with } \mathcal{M} \models \phi[\sigma,\tau[X := S]]
\end{array}$$

Most transition system languages are definable within MSO. An important example, it turns out, is the modal μ-calculus ([24]). Given a set PROP of propositional constants, a set A of atomic actions and an infinite set $\mathcal{V}$ of second-order variables, the language of the modal μ-calculus (i.e. the set of μ-formulas) is defined as the smallest set containing PROP $\cup \mathcal{V} \cup \{\bot\}$, closed under the booleans $\neg$ and $\vee$, and the following two formation-rules:

1. If $a \in A$ and ϕ is a μ-formula, then $\langle a \rangle\phi$ is a μ-formula.
2. If $X \in \mathcal{V}$ and ϕ is a μ-formula positive in X (that is: X only occurs in ϕ under an even number of negations) then $\mu X.\phi$ is a μ-formula.

$\bot$ is again superfluous, as it is definable by $\mu X.X$, as the semantics will show. We leave it in for convenience.

The notion of free and bound variables in μ-formulas should be obvious if we let μX bind free occurrences of X. A μ-formula without free variables is a *μ-sentence.*

μ-formulas are interpreted at states in transition systems, but their meaning may also be given via a *standard translation* into MSO. The following translation sends any μ-formula ϕ to an MSO-formula ϕ° that contains at most one first-order variable x free. This x is intended as the placeholder for the state at which ϕ is evaluated. In the translation, every propositional constant p is considered a unary predicate of $\mathcal{L}$ and every $a \in A$ corresponds to a binary relation R_a, interpreted as the set of all a-transitions. We write R_a in infix notation.

$$\begin{array}{lll}
\bot^\circ & := & \bot \\
p^\circ & := & p(x) \\
X^\circ & := & X(x) \\
(\neg\phi)^\circ & := & \neg\phi^\circ \\
(\phi \vee \psi)^\circ & := & \phi^\circ \vee \psi^\circ \\
(\langle a \rangle\phi)^\circ & := & \exists y.(xR_a y \wedge \phi^\circ[x := y]) \\
(\mu X.\phi)^\circ & := & \forall X.([\forall x.(\phi^\circ \rightarrow X(x))] \rightarrow X(x))
\end{array}$$

In the $\langle a\rangle$-clause y should be a fresh variable. However, if we are careful with our choice of variables we only need two first-order variables in translating a μ-formula into MSO. It is unlikely that there is such a limit on the number of second-order variables needed, due to the results of [7] and [25].

We say that $\mathcal{M}, s \Vdash \phi[\tau]$ (where ϕ is some μ-formula and τ a valuation for the second-order variables) if $\mathcal{M} \models \phi^\circ[\sigma, \tau]$, where σ is any first-order assignment that assigns x to s. If $\mathcal{M}$ is clear from the context, we may simply write $s \Vdash \phi[\tau]$. If furthermore ϕ contains no free variables, we may simply omit τ as well and write $s \Vdash \phi$.

The key operator of the modal μ-calculus is of course the μ-operator. We view $\phi(X)$ as an operation on sets, that to any set S assigns the set of states where $\phi(X)$ holds were we to interpret X by S. If $\phi(X)$ has only positive occurrences of X then the corresponding operator is monotone, which implies the existence of a least fixed-point. This is precisely what $\mu X.\phi(X)$ denotes: it is satisfied at those points that are in the least fixed-point of $\phi(X)$. Monotone operators also have greatest fixed-points: we denote this by $\nu X.\phi(X)$, which is defined as $\neg\mu X.\neg\phi(\neg X)$.

The modal μ-calculus is interpreted on structures of a signature with a unary predicate p for each $p \in$ PROP and a binary relation-symbol R_a for each $a \in A$. In other words: models for the μ-calculus are labeled transition systems: transition systems in which the states may be labeled by propositional constants, depending on whether that state is in the interpretation of these constants. An important notion of equivalence on labeled transition systems is the notion of *bisimulation* ([2, 28]).

Definition 2.2 (Bisimulation) *Let $\mathcal{M}$ and $\mathcal{N}$ be two labeled transition systems. and let $Z \subseteq \mathcal{M} \times \mathcal{N}$. Z is a* **bisimulation** *between $\mathcal{M}$ and $\mathcal{N}$ (notation $Z : \mathcal{M} \underline{\leftrightarrow} \mathcal{N}$) if:*

1. *Z-connected points agree on the propositional constants: if sZt then $s \in p^{\mathcal{M}}$ iff $t \in p^{\mathcal{N}}$ for every $p \in$ PROP.*
2. *$sR_a^{\mathcal{M}}s'$ and sZt implies that there is a t' such that $s'Zt'$ and $tR_a^{\mathcal{N}}t'$;*
3. *Vice versa: if sZt and $tR_a^{\mathcal{N}}t'$ then there is an s' with $sR_a^{\mathcal{M}}s'$ and $s'Zt'$.*

Two nodes s and t are **bisimilar** *iff there is a bisimulation connecting the two.* □

In the computer-science tradition this notion is also known as *strong bisimulation*, to distinguish it from other notions of bisimulation, such as branching bisimulation ([16]).

A transition system $\mathcal{M}$ is *unraveled* if:

1. It has a root from which any other state can be reached via a finite number of transitions;
2. It is well-founded: there is no infinite sequence $s_0, s_1, \ldots$ with transitions from s_{i+1} to s_i for every $i \geq 0$;
3. For every state t that is not the root there exist a unique state s (the predecessor of t) and a unique action $a \in A$ such that $sR_a^{\mathcal{M}}t$

Let κ be some cardinal. A transition system $\mathcal{M}$ is *κ-unraveled* if:

1. It is unraveled;
2. If $sR_a^{\mathcal{M}}t$ then there are at least κ many a-successors t' of s (including t) with the property that t' is bisimilar to t.

Definition 2.3 *Let κ be some cardinal greater than zero, let $\mathcal{M}$ be a labeled transition system and ι some element of $\mathcal{M}$. The* **κ-unraveling** *$(\mathcal{M}^\kappa, \iota^\kappa)$ of $(\mathcal{M}, \iota)$ is defined as follows:*

- *The domain of $\mathcal{M}^\kappa$ consists of all tuples $x = (\iota s_1 \ldots s_n, a_1 \ldots a_n, \alpha_1 \ldots \alpha_n)$, such that $\iota R_{a_1} s_1 \ldots R_{a_n} s_n$ and $\alpha_1, \ldots, \alpha_n$ are ordinals below κ.*
- *$p \in$ PROP is interpreted in $\mathcal{M}^\kappa$ by:*

$$p^\kappa := \{(\vec{s}s, \vec{a}, \vec{\alpha}) \in \mathcal{M}^\kappa \mid s \in p^{\mathcal{M}}\}$$

- *There is an a-transition from x to y in $\mathcal{M}^\kappa$ if x and y are of the form:*

$$\begin{array}{rcl} x & = & (\vec{s}s, \vec{a}, \vec{\alpha}) \\ y & = & (\vec{s}st, \vec{a}a, \vec{\alpha}\alpha) \end{array}$$

  and $sR_a^{\mathcal{M}}t$.

- *$\iota^\kappa := (\iota, \varepsilon, \varepsilon)$, where ε is the empty sequence.* □

The function Z that sends a tuple $(\iota s_1 \dots s_n, a_1 \dots a_n, \alpha_1 \dots \alpha_n)$ to the last element of the nonempty sequence $\iota s_1 \dots s_n$ is a bisimulation between $\mathcal{M}^\kappa$ and $\mathcal{M}$ that moreover connects ι^κ to ι. It is easy to verify that $\mathcal{M}^\kappa$ is κ-unraveled, with ι^κ its root. Thus: any transition system is bisimilar to one that is κ-unraveled.

In fact, it can be shown that two states s and t are bisimilar iff there is some sufficiently large cardinal κ such that the κ-unravelings of s and t are isomorphic.

Let P be some property of states of transition systems. So, given a transition $\mathcal{M}$ and a state s of $\mathcal{M}$, s has the property P or it does not. We call P *invariant for bisimulation* if whenever $Z : \mathcal{M} \underline{\leftrightarrow} \mathcal{N}$ and sZt then s has the property P iff t also has this property.

An MSO-formula $\phi(x)$ (of the right signature of course) that has one first-order variable (x) but no second-order variables free may be considered a property of states, which holds for $\mathcal{M}, s$ iff $\mathcal{M} \models \phi[s]$. So for formulas of this form the invariance-question makes sense. [23] presents a beautiful result, which originally appeared in [21]:

Theorem 2.4 (Janin and Walukiewicz) *An MSO-formula $\phi(x)$ is invariant for bisimulation iff it is equivalent to a (translation of a) modal μ-calculus formula.*

In fact, they show that for every MSO-formula $\phi(x)$ there is an $n \in \mathbb{N}$ such that $\phi(x)$ is equivalent to a μ-formula on n-unraveled structures. This entails that the result also goes through when we restrict to *image-finite models*, models $\mathcal{M}$ such that for any $a \in A$ and any $s \in \mathcal{M}$ there are only finitely many t with sR_at.

The proof of theorem 2.4 depends upon the notion of a μ-automaton:

Definition 2.5 *A* **μ-automaton** *([22]) $\mathcal{A}$ is a tuple $(Q, \Sigma, \Xi, q_0, \delta, \Omega)$ such that:*

1. *Q is a finite set of states;*
2. *Σ is a finite subset of* PROP*;*
3. *Ξ is a finite subset of A;*
4. *$q_0 \in Q$ is the initial state;*
5. *$\delta : Q \times \wp(\Sigma) \to \wp\wp(\Xi \times Q)$ describes the transitions of the automaton ($\wp$ is the powerset constuction);*
6. *$\Omega : Q \to \mathbb{N}$ is the parity condition.* □

We will also refer to such an automaton as a (Σ, Ξ)-automaton, to stress the language of the automaton. Given a transition system $\mathcal{M}$ and a state ι of $\mathcal{M}$ a **μ-game** of $\mathcal{A}$ on $(\mathcal{M}, \iota)$ is played between two players, the Duplicator and the Spoiler, as follows:

1. The starting position is (ι, q_0).
2. If we are in a position (s, q) then the Duplicator has to make a move. Define:

$$\begin{array}{lcl} L(s) & := & \{p \in \Sigma \mid s \Vdash p\} \\ \textsc{succ}_a(s) & := & \{t \mid sR_at\} \\ \textsc{succ}(s) & := & \bigcup_{a \in A} \textsc{succ}_a(s) \end{array}$$

   A legal move for the Duplicator consists of a **marking** $m : \Xi \times Q \to \wp(\textsc{succ}(s))$ such that $m(a, q') \subseteq \textsc{succ}_a(s)$ for every $(a, q') \in \Xi \times Q$, and such that there exists a $D \in \delta(q, L(s))$ with:

   (a) If $(a, q') \in D$ then $m(a, q') \neq \emptyset$.

   (b) For all $a \in \Xi$ and all $t \in \textsc{succ}_a(s)$: there is an $(a, q') \in D$ with $t \in m(a, q')$.

3. If the Duplicator has just made a move, namely a marking m, the Spoiler picks a $t \in m(a, q')$ (for some $a \in \Xi, q' \in Q$). The new position becomes (t, q').

Either player wins the game if the other player cannot make a move. An infinite game $(\iota, q_0), m_0, (s_1, q_1), m_1, (s_2, q_2), m_2 \ldots$ is won by the Duplicator if:

$$min\{\Omega(q) \mid q \text{ appears infinitely often in the sequence}\}$$

is even (hence these automata may be classified as *parity automata*, see [26, 11]). We say that $(\mathcal{M}, s)$ is **accepted** by $\mathcal{A}$ iff the Duplicator has a winning strategy on $(\mathcal{M}, s)$.

In this perspective, a μ-automaton is the arena on which the game is played: enter a state of a labeled transition system, enter the two players and the game can begin! This is the perspective we will adopt in this paper. There is also a more computational perspective, in keeping with why μ-automata are called *automata* in the first place.

A *run* of a μ-automaton $\mathcal{A}$ on a state ι of $\mathcal{M}$ is a rooted intransitive tree $\mathcal{T}$ whose nodes are labeled with elements of $\mathcal{M} \times Q$ such that:

1. The root is labeled with (ι, q_0).

2. If a node n of $\mathcal{T}$ is labeled with (s, q) then there must be a $D \in \delta(q, L(s))$ such that:

   (a) If $(a, q') \in D$ then there is a a-successor t of s and a successor m of n such that m is labeled with (t, q').

   (b) If $sR_a t$ for some $a \in \Xi$ then there is an $(a, q') \in D$ and a successor m of n that is labeled with (t, q').

A run is *accepting* if whenever $(\iota, q_0), (s_1, q_1), (s_2, q_2), \ldots$ is an infinite path through $\mathcal{T}$ from the root onwards (upwards or downwards, depending on how you draw your trees) then:

$$min\{\Omega(q) \mid q \text{ appears infinitely often in the sequence}\}$$

is even.

The idea is that the run embodies a winning strategy for the Duplicator. An automaton thus accepts a state in a labeled transition system iff it has an accepting run on this state.

Janin and Walukiewicz ([22]) prove that μ-automata correspond precisely to μ-sentences: for every μ-automaton $\mathcal{A}$ there is a μ-sentence $\phi_{\mathcal{A}}$ such that $\mathcal{A}$ accepts $(\mathcal{M}, s)$ iff $\mathcal{M}, s \Vdash \phi_{\mathcal{A}}$. Moreover, if $\mathcal{A}$ is a (Σ, Ξ)-automaton then the corresponding formula is also in this language. The converse also holds: for every μ-sentence there is an automaton $\mathcal{A}_\phi$ such that $\mathcal{A}_\phi$ accepts $\mathcal{M}$ iff $\mathcal{M} \Vdash \phi$. If the set of proposition constants in ϕ is Σ and its set of atomic actions is Ξ, then the corresponding automaton may be assumed to be a (Σ, Ξ)-automaton. Our automata differ slightly from those given in [22]: they are more akin to the definition given in [23]. The automata in their various guises are equivalent, however.

The notion of μ-automaton is heavily inspired by the notion of *tableaux* for the μ-calculus ([27]). First of all, we assume that μ-calculus formulas are presented with negation only applied to propositional constants. This necessitates the use of connectives $\wedge$, $[a]$ and νX. Next, we assume that, given a μ-sentence ϕ, each variable X is used by at most one binder. Then if X occurs in ϕ we may refer to its *binding definition* $\sigma X.\psi$ as the unique formula of this form in ϕ (σ is either μ or ν). One of the main tableau-rules is the *regeneration* rule, which expands a variable X to its binding definition. Due to this rule tableaux will in general be infinite: finitely branching but with infinite paths. On such infinite paths there must be variables regenerated infinitely often. If a variable X whose binding definition is of the form $\mu X.\psi$ is regenerated infinitely often then we must be careful. This is only sanctioned if there is another variable Y occurring free in ψ that also occurs infinitely often on the same path. If this is not the case we are dealing with a *globally inconsistent path.* For more details we refer to [27] and to [22]. The point here is that the notion of globally (in)consistent path is a clear correspondent to the parity condition for games or runs on μ-automata. The other rules of the tableau system are basically as in tableau systems for modal logic without fixed points (consult [18] for an overview).

To conclude this section, we give some examples of μ-automata, to acquaint the reader with them somewhat. First, we consider the ultimate valid μ-sentence $\top$ (i.e. $\neg\bot$). The corresponding μ-automaton is $(\{q_0\}, \Sigma, \Xi, q_0, \delta, \Omega)$ where:

- Σ and Ξ are arbitrary subsets of PROP and A respectively.
- $\delta(q_0, L) = \wp(\Xi \times \{q_0\})$ for any $L \subseteq \Sigma$.
- $\Omega(q_0) = 0$.

In a game on this automaton the Duplicator can always respond with a legal marking. For suppose the current position is (s, q_0). Then the Duplicator may respond with the marking m, defined on (a, q_0) as $\textsc{succ}_a(s)$. This marking is a legal one with respect to $D = \{(a, q_0) \mid a \in \Xi, \textsc{succ}_a(s) \neq \emptyset\}$. The parity condition on infinite games is always satisfied as the only state q_0 has as its parity an even number.

Looking carefully at this automaton we see that it actually behaves like the formula:

$$\nu X. \bigvee_{L\subseteq\Sigma} ((\bigwedge L \wedge \bigwedge_{p\in L} \neg p) \wedge (\bigvee_{M\subseteq\Xi} \bigwedge_{a\in M} \langle a\rangle X) \wedge \bigwedge_{a\in\Xi} [a]X)$$

which is of course equivalent to $\top$: $\top$ is a fixed point and hence the maximal one for the matrix of this formula.

The automaton also gives us the useful notion of a *true state.*

Definition 2.6 *A state $q \in Q$ of an automaton $\mathcal{A} = (Q, \Sigma, \Xi, q_0, \delta, \Omega)$ is a* **true state** *if $\Omega(q)$ is even and for all $L \subseteq \Sigma$, $\delta(q, L) = \wp(\Xi \times \{q\})$.*

It is now easy to see that whenever the game arrives at a position (s, q), where q is a true state, then the Duplicator will win this game.

Let us consider a second example of a μ-automaton, one corresponding to the PDL-formula $\langle a^*\rangle p$. Let $\mathcal{A}$ be $(\{q_0, q_1\}, \Sigma, \Xi, q_0, \delta, \Omega)$ with:

- $\Sigma = \{p\}$ and $\Xi = \{a\}$.
- $\delta(q, L) = \begin{cases} \{\{(a, q_0), (a, q_1)\}\} & \text{if } q = q_0 \text{ and } L = \emptyset. \\ \{\{(a, q_1)\}, \emptyset\} & \text{else.} \end{cases}$
- $\Omega(q_0) = 1$ and $\Omega(q_1) = 0$.

Suppose we are given a labeled transition system $\mathcal{M}$ and an $s_0 \in \mathcal{M}$ such that $s_0 \Vdash \langle a^*\rangle p$. Then there is a path $s_0 R_a s_1 R_a \ldots R_a s_n$, where s_n is the first state on this path that satisfies p. A strategy for the Duplicator consists of the following:

- On a position (s_i, q_0), with $i < n$ the Duplicator responds with the following marking:

  $$m(a, q_0) = \{s_{i+1}\} \qquad m(a, q_1) = \textsc{succ}_a(s)$$

  which is legal with respect to $D = \{(a, q_0), (a, q_1)\}$, the only element of $\delta(q_0, L(s_i))$. The duplicator can either respond with the position (s_{i+1}, q_0) in which case we continue with the strategy we describe here, or with a position (t, q_1) from which the Duplicator must win as q_1 is a true state.

- On the position (s_n, q_0) the Duplicator should play the marking:

$$m(a, q_0) = \emptyset \qquad m(a, q_1) = \textsc{succ}_a(s)$$

  which is correct with respect to $\{(a, q_1)\}$ if $\textsc{succ}_a(s) \neq \emptyset$ and $\emptyset$ otherwise, both of which are elements of $\delta(q_0, L(s_n))$, as $L(s_n) = \{p\}$. The Spoiler can now only respond with a position (t, q_1).

- On any position (t, q_1) the Duplicator plays the same markings as in the previous case.

Clearly this gives a winning strategy for the Duplicator. The parity condition is satisfied because after n steps into the game only the state q_1 can be reached, whose parity is even.

Now suppose s is a node in a labeled transition system $\mathcal{M}$ at which $\langle a^* \rangle p$ is *not* true. If the Duplicator has to respond to a position (s, q_0) for such an s then he must give a marking m that is legal with respect to $\{(a, q_0), (a, q_1)\}$, as p does not hold at s. By definition of legal markings $m(a, q_0) \neq \emptyset$, so the Spoiler can simply choose a position (t, q_0) with $sR_a t$. $\langle a^* \rangle p$ will of course also be false in t, so the Spoiler can continue with this strategy indefinitely or until the Duplicator gets stuck. In the latter case the Spoiler wins, while in the former case we get a sequence of play:

$$(s, q_0), m_1, (s_1, q_0), m_2, (s_2, q_0), \ldots$$

on which the Duplicator loses, because q_0 has an odd parity.

Examining the workings of this automaton, we discover that it resembles a procedural interpretation of the formula $\mu X.((\neg p \wedge \langle a \rangle X) \vee p)$, which is equivalent to $\langle a^* \rangle p$.

We conclude this section with one more example: an automaton corresponding to $\nu X.\langle a \rangle X$, which expresses inverse nonwellfoundedness, i.e. the formula is true at states from which an infinite a-path is possible. Let $\mathcal{A}$ be $(\{q_0, q_1\}, \Sigma, \Xi, q_0, \delta, \Omega)$ with:

- $\Sigma = \emptyset$ and $\Xi = \{a\}$.
- $$\begin{array}{lcl} \delta(q_0, \emptyset) & = & \{\ \{(a, q_0), (a, q_1)\}\ \} \\ \delta(q_1, \emptyset) & = & \{\ \{(a, q_1)\}\ , \emptyset\} \end{array}$$
- $\Omega(q_0) = \Omega(q_1) = 0$.

Note that q_1 is again a true state, so that any strategy for the Duplicator can concentrate on positions of the form (s, q_0). Infinite games are always won by the Duplicator as there are no odd parities available. Suppose s is (inversely) nonwellfounded and we have reached a position (s, q_0). As there is an infinite a-path from s there must be an a-successor t of s that is also nonwellfounded. Let the Duplicator choose a marking:

$$m(a, q_0) = \{t\} \qquad m(a, q_1) = \textsc{succ}_a(s).$$

This marking is legal with respect to $\{(a, q_0), (a, q_1)\}$. The Spoiler's only real choice (he has to avoid the true state) is the position (t, q_0), on which the Duplicator can respond with the same strategy. This gives an infinite game, which the Duplicator automatically wins. So the Duplicator has a winning strategy on nonwellfounded states.

If s is wellfounded, i.e. there is no infinite a-path starting in s, then the Spoiler has a winning strategy on (s, q_0). For the Duplicator must respond with a marking m where $m(a, q_0) \neq \emptyset$. The Spoiler can now respond with (t, q_0) where t is any a-successor from $m(a, q_0)$. In fact, it is possible for the Spoiler to continue choosing positions where the second component is q_0. However, as s is wellfounded, at a certain moment the game reaches a position (t, q_0) such that a-transitions are no longer possible. Then the Duplicator is stuck and the Spoiler wins the game.

3 Safety and complete additivity

Invariance for bisimulation is a generalization of the clause of the definition of bisimulation for propositional constants. *Safety for bisimulation* is a generalization of the clause for atomic actions, which deals with binary relations. So let R assign to every transition system $\mathcal{M}$ a binary relation $R^{\mathcal{M}}$ on $\mathcal{M}$. We say that R is safe for bisimulation if whenever $Z : \mathcal{M} \underline{\leftrightarrow} \mathcal{N}$, sZt and $sR^{\mathcal{M}}s'$ then there exists a t' in $\mathcal{N}$ such that $tR^{\mathcal{N}}t'$ and $s'Zt'$. The vice-versa clause follows automatically, because bisimilarity is a symmetric concept.

Again, the safety-question may be specialized to MSO, for if $\phi(x, y)$ is an MSO-formula that has at most x and y free, this may be viewed as giving a binary relation between states on a given transition system. The question this paper will address is: which MSO-formulas $\phi(x, y)$ are safe for bisimulation? A mathematical paper is not a mystery novel, so let us define the fragment of MSO that contains, modulo logical equivalence, precisely the formulas that are safe for bisimulation.

Definition 3.1 *A* **μ-program** *is a formula in the following language:*

$$\pi ::= a \mid \phi? \mid \pi \cup \pi \mid \pi ; \pi \mid \pi^*$$

where $a \in A$ and ϕ is a μ-formula.

μ-programs of the form $\phi?$ are known as *tests*. A μ-program whose only tests are of the form $\phi?$, where ϕ is a *μ-sentence*, is called *closed*. Note that if we restrict the tests to PDL-formulas, the resulting restricted class of μ-programs coincides with that of the PDL-programs. So the only enrichment of this language lies in the tests: we are able to test statements which PDL cannot express.

Again we give a translation into MSO to define the semantics of these formulas:

$$\begin{array}{lcl}
a^\circ & := & R_a(x, y) \\
(\phi?)^\circ & := & \phi^\circ \wedge x = y \\
(\pi_1 \cup \pi_2)^\circ & := & \pi_1^\circ \vee \pi_2^\circ \\
(\pi_1 ; \pi_2)^\circ & := & \exists z.(\pi_1^\circ[y := z] \wedge \pi_2^\circ[x := z]) \\
(\pi^*)^\circ & := & \forall X((X(x) \wedge \forall xy.[X(x) \wedge \pi^\circ \to X(y)]) \to X(y))
\end{array}$$

This time careful variable-management enables us to use only three first-order variables. We will sometimes write $s \xrightarrow{\pi} t$ for $\mathcal{M} \models \pi^\circ[s, t]$ if $\mathcal{M}$ is clear from the context and π is a closed μ-program.

Given a μ-program π and a μ-formula ϕ, we may define a formula $\langle\pi\rangle\phi$ with as interpretation:

$$(\langle\pi\rangle\phi)^\circ := \exists y.(\pi^\circ \wedge \phi^\circ[x := y])$$

Adding such formulas to the language of the modal μ-calculus would not enrich it, however, as $\langle\pi\rangle\phi$ may be defined for complex π within the μ-calculus as follows:

$$\begin{array}{lcl}
\langle\phi?\rangle\psi & := & \phi \wedge \psi \\
\langle\pi_1 \cup \pi_2\rangle\phi & := & \langle\pi_1\rangle\phi \vee \langle\pi_2\rangle\phi \\
\langle\pi_1 ; \pi_2\rangle\phi & := & \langle\pi_1\rangle\langle\pi_2\rangle\phi \\
\langle\pi^*\rangle\phi & := & \mu X.(\phi \vee \langle\pi\rangle X)
\end{array}$$

Invariance and safety are closely connected. For if $\phi(x)$ is invariant for bisimulation, then $\phi(x) \wedge x = y$ is safe for bisimulation. Conversely, if $\phi(x,y)$ is safe for bisimulation and p is some propositional constant that does not occur in ϕ then $\exists y.(\phi(x, y) \wedge p(y))$ is invariant for bisimulation. By theorem 2.4, the latter formula must be equivalent to a closed μ-formula. This formula is of a particular kind: it is *completely additive* in p. To define this we need the following definition:

Definition 3.2 *Let $p \in$ PROP. If $\mathcal{M}$ is a transition system for the language without p then for any $P \subseteq \mathcal{M}$, $(\mathcal{M}, P)$ is a model for the whole language. A μ-sentence $\phi(p)$ can be viewed as a set-operator. If $\mathcal{M}$ is any model for the language without p then $\phi^{\mathcal{M}} : \wp(\mathcal{M}) \to \wp(\mathcal{M})$ is defined as follows:*

$$\phi^{\mathcal{M}}[P] := \{s \in \mathcal{M} \mid (\mathcal{M}, P), s \Vdash \phi(p)\}$$

If $\mathcal{M}$ is clear from the context, the superscript is dropped. □

Definition 3.3 *A μ-sentence $\phi(p)$ is* **completely additive in** *p if it distributes over arbitrary unions:*

$$\phi[\bigcup_{i\in I} P_i] = \bigcup_{i\in I} \phi[P_i]$$

$\square$

In [4] this notion is referred to as *continuity in p*. This nomenclature seems to clash with the more familiar notion of continuity in the literature (see for instance [31]). Continuity in the present context would mean distributivity over unions of *directed sets*. A directed set in a transition system $\mathcal{M}$ is a nonempty set $\mathcal{D} \subseteq \wp(\mathcal{M})$ such that for all $A, B \in \mathcal{D}$ there is a $C \in \mathcal{D}$ with $A \cup B \subseteq C$. A sentence $\phi(p)$ is continuous in p iff for all $\mathcal{M}$ for all directed sets $\mathcal{D}$ in $\mathcal{M}$, $\phi[\bigcup \mathcal{D}] = \bigcup_{A\in\mathcal{D}} \phi[A]$. Clearly complete additivity implies continuity, but not the other way around: $p \wedge \langle a\rangle p$ is continuous in p, but does not even distribute over finite unions.

Additivity is equivalent to the conjunction of the following two properties:

Monotonicity: $P \subseteq Q$ implies $\phi[P] \subseteq \phi[Q]$.

Downward additivity: $\phi[P] \subseteq \bigcup_{t\in P} \phi[\{t\}]$. In other words: if $s \Vdash \phi(p)$ then we may restrict the interpretation of p to a singleton without losing the truth of $\phi(p)$ at s.

Additivity of course implies distributivity over finite unions. We will see later that in the case of ordinary polymodal logic without fixed points distributivity over finite unions implies complete additivity. This is not the case for the μ-calculus, as the counterexample $\nu X.\langle a; a^*; p?\rangle X$ ('there is an infinite a-path that encounters p infinitely often') demonstrates. However, we will see that for μ-calculus formulas, distributivity over *countable* unions suffices for complete additivity to hold.

It is easy to see that if π is a closed μ-program that does not contain p then $\langle\pi\rangle p$ is completely additive in p. It turns out that all completely additive formulas are equivalent to one of this form.

Theorem 3.4 *Suppose the μ-sentence $\phi(p)$ distributes over countable unions. Then $\phi(p)$ is equivalent to $\langle\pi\rangle p$ for some p-free closed μ-program π.*

Proof.
Let $\mathcal{A} = (Q, \Sigma, \Xi, q_0, \delta, \Omega)$ be a μ-automaton corresponding to $\phi(p)$. For $q \in Q$, let ϕ_q be the μ-sentence corresponding to the automaton we get from $\mathcal{A}$ by changing the initial state from q_0 to q.

We will prove that:

$$\phi(p) \equiv \bigvee\{\langle\pi\rangle p \mid \pi \in \Pi \text{ and } \langle\pi\rangle p \models \phi(p)\} \tag{1}$$

where Π is a finite set of closed p-free π-programs, defined as follows:

- Let TYPE_0 be the set of μ-sentences constructed from a subset $L \subseteq \Sigma \setminus \{p\}$ as follows:

$$\bigwedge L \wedge \bigwedge\{\neg p' \mid p' \in \Sigma \setminus (L \cup \{p\})\}$$

- TYPE_1 is the set of all μ-sentences

$$\bigwedge_{q\in S} \phi_q[p := \bot] \wedge \bigwedge_{q\in Q\setminus S} \neg\phi_q[p := \bot]$$

  where S is a subset of Q.

- TYPE_2 contains all μ-sentences

$$\phi \wedge (\bigwedge_{(a,\psi)\in S} \langle a\rangle\psi) \wedge (\bigwedge_{a\in\Xi} [a] \bigvee_{(a,\psi)\in S} \psi)$$

  where $\phi \in \mathsf{TYPE}_0$ and $S \subseteq \Xi \times \mathsf{TYPE}_1$.

- PTYPE_0 ('*path-type zero*') consists of all μ-programs of the form:

$$(\bigcup_{(\phi,a)\in S} \phi?; a) \cup \bigcup_{\phi\in T} \phi?$$

  where $S \subseteq \mathsf{TYPE}_2 \times \Xi$ and $T \subseteq \mathsf{TYPE}_2$.

- Suppose PTYPE_k has been defined. Then PTYPE_{k+1} is defined to contain all μ-programs of the form:

$$\pi_1 \cup (\pi_2; \pi_3^*; \pi_4)$$

  where each $\pi_i \in \mathsf{PTYPE}_k$.

- Finally we can define Π. Suppose Q has m different subsets (i.e. $m = 2^{|Q|}$). Then:

$$\Pi := \{\pi; \phi? \mid \pi \in \mathsf{PTYPE}_m, \phi \in \mathsf{TYPE}_2\}$$

  Note that Π is a finite set of closed p-free μ-programs.

If we can prove statement (1) then:

$$\phi(p) \equiv \langle \bigcup \{\pi \in \Pi \mid \langle\pi\rangle p \models \phi(p)\}\rangle p$$

which suffices for proving of the theorem.

We proceed to prove statement (1). From right to left the statement is trivial. From left to right we need to show that if $\mathcal{M}, s \Vdash \phi(p)$ then there is some $\pi \in \Pi$ such that $s \Vdash \langle\pi\rangle p$ and $\langle\pi\rangle p \models \phi(p)$.

Suppose $(\mathcal{M}, P), s \Vdash \phi(p)$. We may assume that $\mathcal{M}$ is 2-unraveled with s its root. Furthermore, we may assume that states are not labeled with propositional constants not in Σ and that relations R_a do not connect any states if $a \notin \Xi$. For each $x \in \mathcal{M}$, define:

$$\begin{array}{lll}
\mathsf{type}_0(x) & := & \bigwedge\{p' \in \Sigma \setminus \{p\} \mid x \Vdash p'\} \wedge \\
 & & \bigwedge\{\neg p' \mid p' \in \Sigma \setminus \{p\}, x \nVdash p'\} \\
 & & \\
\mathsf{type}_1(x) & := & \bigwedge\{\phi_q[p := \bot] \mid q \in Q, x \Vdash \phi_q[p := \bot]\} \wedge \\
 & & \bigwedge\{\neg\phi_q[p := \bot] \mid q \in Q, x \nVdash \phi_q[p := \bot]\} \\
 & & \\
\mathsf{type}_2(x) & := & \mathsf{type}_0(x) \wedge \\
 & & \bigwedge\{\langle a\rangle \mathsf{type}_1(y) \mid a \in \Xi, xR_a y\} \wedge \\
 & & \bigwedge_{a\in\Xi}[a] \bigvee_{xR_a y} \mathsf{type}_1(y)
\end{array}$$

For $i \leq 2$, we call $\mathsf{type}_i(x)$ the *i-type of x*. Note that these i-types are p-free μ-sentences and that each i-type is an element of TYPE_i. Furthermore, all i-types are disjoint: if $x \Vdash \mathsf{type}_i(y)$ then x and y have the same i-type.

Next, we divide all transitions into left- and right- transitions. That is, for each $x \in \mathcal{M}$ and $a \in A$ consider all a-successors of x, $\textsc{succ}_a(x)$. Partition this set into $\mathcal{L} \setminus \{p\}$-bisimulation classes. By the assumption that $\mathcal{M}$ is 2-unraveled, we know that each part B of the partition contains at least 2 elements. So we can split B into two nonempty nonoverlapping sets L and R. Call each a-transition from x to an element of L a *left*-transition and a-transitions from x to elements of R: *right*-transitions.

Now we can define the *type* of a path $s = s_0 R_{a_1} s_1 \ldots R_{a_n} s_n$ as a tuple:

$$(\mathsf{type}_2(s_0), a_1, d_1, \mathsf{type}_2(s_1), \ldots, a_n, d_n, \mathsf{type}_2(s_n))$$

where $d_{i+1} = l$ if the transition from s_i to s_{i+1} is a left-transition and $d_{i+1} = r$ otherwise. Note that for paths of some fixed length there are only finitely many types. Hence, overall, there are only countably many types.

Define for each path-type τ, P_τ as $\{x \in P \mid \text{the path from } s \text{ to } x \text{ is of type } \tau\}$. This partitions P into countably many sets. Thus $(\mathcal{M}, P_\tau), s \Vdash \phi(p)$ for some type τ, by the assumption that $\phi(p)$

distributes over countable unions. We know that P_τ cannot be empty, because $\phi(p)$ also distributes over empty unions: $\phi[\emptyset] = \emptyset$.

A *p-path* is a path from the root s to a point with a label p. In the current model $(\mathcal{M}, P_\tau)$ all p-paths are of the same type.

Claim 1: For any two nodes x and y on a (possibly distinct) p-path that are at the same depth (the same distance from the root) and any $q \in Q$: $x \Vdash \phi_q$ iff $y \Vdash \phi_q$.

Proof.
Assume $\tau = (\mathsf{type}_0, a_1, d_1, \mathsf{type}_1, \ldots, a_n, d_n, \mathsf{type}_n)$. Occurrences of p are thus only found via paths of this type. We prove the claim by downward induction: we first prove it for depth n, then for $n-1$, etc.

Suppose x and y are on a p-path at depth n. Then p holds at both x and y, p no longer holds anywhere above x and y, and x and y have the same 2-type type_n. Suppose $x \Vdash \phi_q$. Then the Duplicator has a winning strategy for the position (x, q). Let this strategy choose a marking m that is correct w.r.t. $D \in \delta(q, L(x))$. D is also in $\delta(q, L(y))$, because $L(x) = L(y)$: x and y have the same 2-type, hence the same 0-type, and p is true at both.

Define the following marking:

$$\overline{m}(a, q') := \{z \mid yR_a z \Vdash \phi_{q'}\}$$

If this is a legal marking, the Spoiler can only choose a position (z, q') such that $z \Vdash \phi_{q'}$: the Duplicator can obviously win from such a position.

We prove that $\overline{m}$ is legal w.r.t. D. We need to prove two statements:

1. $(a, q') \in D$ implies that there is an a-successor z of y with $z \in \overline{m}(a, q')$. For if $(a, q') \in D$ then for some a-successor x' of x: $x' \in m(a, q')$, hence $x' \Vdash \phi_{q'}$. As p does not occur anywhere above x', nor at x' itself, this implies: $x' \Vdash \phi_{q'}[p := \bot]$. As x and y have the same 2-type: $y \Vdash \langle a \rangle \phi_{q'}[p := \bot]$. There are no p-labels strictly above y either, so $y \Vdash \langle a \rangle \phi_{q'}$.

2. $yR_a z$ and $a \in \Xi$ implies that there is a $q' \in Q$ with $z \in \overline{m}(a, q')$. For suppose $yR_a z$. As x and y have the same 2-type, $x \Vdash \langle a \rangle \mathsf{type}_1(z)$, say $xR_a x' \Vdash \mathsf{type}_1(z)$. By the fact that m is legal, $x' \in m(a, q')$, for some $q' \in Q$. Thus $x' \Vdash \phi_{q'}$, and by the same reasoning as before $x' \Vdash \phi_{q'}[p := \bot]$. But $\mathsf{type}_1(x') = \mathsf{type}_1(z)$, so $z \Vdash \phi_{q'}[p := \bot]$, hence $z \Vdash \phi_{q'}$.

Next, suppose x and y occur at depth $k-1$ and suppose that $x \Vdash \phi_q$. Let m be a marking chosen according to the winning strategy of the Duplicator on (x, q). m is correct w.r.t. some $D \in \delta(q, L(x)) = \delta(q, L(y))$. Define $\overline{m}$ as before.

1. Suppose $(a, q') \in D$. Then for some x', $xR_a x' \Vdash \phi_{q'}$. There are two cases:

   (a) x' is on a p-path. Then a must be a_k. Now choose an a-successor z of y that is also on a p-path (this is possible because there is only one type of path to p: this fixes the action we must take to eventually reach a label p). By the induction hypothesis: $z \Vdash \phi_{q'}$.

   (b) x' is not on a p-path, so $x' \Vdash \phi_{q'}[p := \bot]$. As y has the same 2-type as x, $y \Vdash \langle a \rangle \phi_{q'}[p := \bot]$. Thus for some z: $yR_a z \Vdash \phi_{q'}[p := \bot]$. If z is not on a p-path then $z \Vdash \phi_{q'}$, as desired. Else, switch direction (if $yR_a z$ is a left-transition switch to any corresponding right-transition, and vice versa) to an a-successor z', $\mathcal{L} \setminus \{p\}$-bisimilar to z that is not on a p-path.

2. Suppose $yR_a z$ and $a \in \Xi$. There are two cases to consider:

   (a) z is on a p-path. Choose an a-successor x' of x that is also on a p-path. There must be some $(a, q') \in D$ with $x' \in m(a, q')$ and hence $x' \Vdash \phi_{q'}$. By the induction hypothesis $z \Vdash \phi_{q'}$.

(b) z is not on a p-path. As x and y have the same 2-type, x has an a-successor x' that has the same 1-type as z. By switching direction if necessary we may assume that x' does not lie on a p-path. m is a legal marking w.r.t. D, so we can find $(a, q') \in D$ s.t. $x' \in m(a, q')$, hence $x' \Vdash \phi_{q'}$. x' is not on a p-path, so $x' \Vdash \phi_{q'}[p := \bot]$. As x' and z have the same 1-type, $z \Vdash \phi_{q'}[p := \bot]$, which implies $z \Vdash \phi_{q'}$. □

Define, for any $x \in \mathcal{M}$, the *labeling* of x as:

$$l(x) := \{q \in Q \mid x \Vdash \phi_q\}$$

Note that Claim 1 now says that if x and y are on a p-path and at the same depth, then $l(x) = l(y)$. Also, as $\phi(p)$ is equivalent to ϕ_{q_0} and $s \Vdash \phi(p)$, $q_0 \in l(s)$.

Pick any path $s = s_0 R_{a_1} s_1 \ldots R_{a_n} s_n \Vdash p$. For any $I, O \subseteq Q$, an *(I, O)-program* is a μ-program of one of the following two forms:

- $\mathsf{type}_2(s_i)?; a_{i+1}$, where $i < n$, $l(s_i) = I$ and $l(s_{i+1}) = O$;
- $\mathsf{type}_2(s_i)?$, where $i \leq n$ and $l(s_i) = I = O$.

From these one-step programs we build a larger μ-program. First enumerate $\wp(Q)$ as $Q_1, \ldots, Q_m$. Define programs $\pi_k(Q_i, Q_j)$, for every $1 \leq i, j \leq m$ and $0 \leq k \leq m$, by induction on k:

$$\pi_0(Q_i, Q_j) \quad := \quad \bigcup\{\pi \mid \pi \text{ is a } (Q_i, Q_j)\text{-program}\}$$

$$\pi_{k+1}(Q_i, Q_j) \quad := \quad \pi_k(Q_i, Q_j) \cup (\pi_k(Q_i, Q_{k+1}); \pi_k(Q_{k+1}, Q_{k+1})^*; \pi_k(Q_{k+1}, Q_j))$$

In the definition of $\pi_0(Q_i, Q_j)$ we define the empty union as $\bot?$. Now define $\pi := \pi_m(l(s_0), l(s_n)); \mathsf{type}_2(s_n)?$. It is easily verified that $\pi \in \Pi$.

Claim 2: $(\mathcal{M}, P_\tau), s \Vdash \langle\pi\rangle p$, hence $(\mathcal{M}, P), s \Vdash \langle\pi\rangle p$.

Proof.
Order the subsets of Q according to their enumeration: $Q_1 < \ldots < Q_m$. We will prove that for any $i \leq j \leq n$, and $k \leq m$: if on the path from s_i to s_j there is no node x strictly between s_i and s_j such that $l(x)$ is a subset of Q higher in the ordering than Q_k (assuming 'higher than Q_0' to be always true), then there is a $\pi_k(l(s_i), l(s_j))$-transition from s_i to s_j. We prove it by induction on k:

- If $k = 0$ then there can be *no* nodes strictly between s_i and s_j. We need to show that there is a $\pi_0(l(s_i), l(s_{i+1}))$-transition between s_i and s_j. For that purpose s_i and s_j must be connected via an $(l(s_i), l(s_j))$-program. There are two cases. If $s_i = s_j$ then $\mathsf{type}_2(s_i)?$ is the desired program. If $s_j = s_{i+1}$ then $\mathsf{type}_2(s_i)?; a_{i+1}$ is the desired program.
- Suppose we have proved it for $k < m$ and we now wish to do the same for $k + 1$. If there is no x strictly between s_i and s_j such that $l(x) = Q_{k+1}$ then by the induction hypothesis there is a $\pi_k(l(x), l(y))$-transition from s_i to s_j, which is also a $\pi_{k+1}(l(x), l(y))$-transition.

  If there *are* such intermediate points however, enumerate all of these as $x_0, \ldots, x_l$ in the order of occurrence on the path. Then all the points y on the path from s_i to s_j that are not among $\{s_i, x_0, \ldots, x_l, s_j\}$ are such that $l(y) \leq Q_k$. We can thus use the induction hypothesis to show that there is a $\pi_k(l(s_i), Q_{k+1})$-transition from s_i to x_0, for any $\alpha < l$ there is a $\pi_k(Q_{k+1}, Q_{k+1})$-transition from x_α to $x_{\alpha+1}$ and finally there is a $\pi_k(Q_{k+1}, l(s_j))$-transition from x_l to s_j. So there is a

  $$\pi_k(l(s_i), Q_{k+1}); \pi_k(Q_{k+1}, Q_{k+1})^*; \pi_k(Q_{k+1}, l(s_j))$$

  -transition from s_i to s_j, as desired.

We may deduce that there is a $\pi_m(l(s_0), l(s_n))$-transition from s_0 to s_n, as there are no subsets of Q and hence no labelings above Q_m. Finally, we note that $s_n \Vdash \mathsf{type}_2(s_n)$ and hence we have a π-transition from $s = s_0$ to s_n. □

Claim 3: $\langle\pi\rangle p \models \phi(p)$.

Proof.
Suppose $(\mathcal{N}, P'), s' \Vdash \langle\pi\rangle p$. We may again assume that $(\mathcal{N}, P')$ is 2-unraveled. Then $s' \xrightarrow{\pi} t'$, for some $t' \in P'$. We will prove that $(\mathcal{N}, \{t'\}), s' \Vdash \phi(p)$. By monotonicity it then follows that $(\mathcal{N}, P'), s' \Vdash \phi(p)$.

We give a winning strategy for the $\mathcal{A}$-game on $(\mathcal{N}, \{t'\}), s'$. Let the unique path from s' to t' be $s' = t_0 R_{b_1} t_1 \dots R_{b_k} t_k = t'$.

1. Suppose the Duplicator has to respond to a position (t_i, q), with $i < k$. By definition of π, there is a transition from t_i to t_{i+1} of the form $\mathsf{type}_2(s_j)?; a_{j+1}$ for some $j < n$. Let $I = l(s_j)$, $O = l(s_{j+1})$, making this an (I, O)-program. We may assume that $q \in I$. For if $i = 0$ then $q = q_0$ and any first step of the program π must be an $(l(s_0), O)$-step (for some O), hence $q_0 \in I = l(s_0)$. If $i > 0$ then there must be some (I', O')-step from t_{i-1} to t_i. It may easily be verified that O' must be equal to I. Furthermore, our strategy will ensure that only positions (t_i, q') with $q' \in O'$ can be chosen by the Spoiler. So our assumption is valid.

   If $q \in I = l(s_j)$ then the Duplicator has a winning strategy for the $\mathcal{A}$-game starting from the position (s_j, q). Let m be the marking chosen according to this strategy. Suppose m is legal w.r.t. $D \in \delta(q, L(s_j)) = \delta(q, L(t_i))$. Define:

$$\overline{m}(a, q') := \begin{array}{l} \{x \mid t_i R_a x \neq t_{i+1} \text{ and } x \Vdash \phi_{q'}\} \cup \\ \{t_{i+1} \mid q' \in O \text{ and } a = a_{j+1}\} \end{array}$$

   This is again a legal marking with respect to D.

   For suppose $(a, q') \in D$. Then $m(a, q') \neq \emptyset$, say $x \in m(a, q')$. The Spoiler's next move could be the position (x, q'), so $q' \in l(x)$. There are two cases:

   (a) x is on a p-path. Then by Claim 1, $q' \in O = l(s_{j+1}) = l(x)$, so $t_{i+1} \in \overline{m}(a, q')$.

   (b) x is not on a p-path. But then there is no p above or at x so the interpretation of p from x onwards is equal to that of $\perp$. Hence: $x \Vdash \phi_{q'}[p := \perp]$. t_i has the same 2-type as s_j, so $t_i \Vdash \langle a\rangle\mathsf{type}_1(x)$. $\mathsf{type}_1(x)$ implies $\phi_{q'}[p := \perp]$, so there is an a-successor y of t_i such that $y \Vdash \phi_{q'}[p := \perp]$. As $\mathcal{N}$ is 2-unraveled, we can choose an a-successor z of t_i which is $\mathcal{L} \setminus \{p\}$-bisimilar to y, but which is not on the path to t'. As t' is the only place where p holds, $z \Vdash \phi_{q'}$, hence $z \in \overline{m}(a, q')$.

   Next, suppose $t_i R_a x$ for some $a \in \Xi$. We need to find some $(a, q') \in D$ such that $x \in \overline{m}(a, q')$. There are again two cases:

   (a) $x = t_{i+1}$. Then $a = a_{j+1}$. As $s_j R_a s_{j+1}$ there is some $(a, q') \in D$ with $s_{j+1} \in m(a, q')$. But then $q' \in l(s_{j+1}) = O$, so $t_{i+1} \in \overline{m}(a, q')$.

   (b) $x \neq t_{i+1}$. Then there is no point from x on where p holds. As t_i has the same 2-type as s_j, $t_i \Vdash [a] \bigvee_{s_j R_a y} \mathsf{type}_1(y)$, so $x \Vdash \mathsf{type}_1(y)$ for some a-successor y of s_j. By switching direction if necessary we may even choose y not to be on a p-path. Because m is a legal marking w.r.t. D there must be an $(a, q') \in D$ with $y \in m(a, q')$. But then $y \Vdash \phi_{q'}[p := \perp]$. As x and y have the same 1-type, $x \Vdash \phi_{q'}[p := \perp]$ also, hence $x \Vdash \phi_{q'}$.

   The Spoiler may respond to this marking in two ways.

   (a) He may pick a position (x, q') with $x \neq t_{i+1}$ and $x \Vdash \phi_{q'}$. But then the Duplicator has a winning strategy from this point on. Continue with this strategy.

   (b) He may pick a position (t_{i+1}, q') with $q' \in O$. Continue with the strategy we describe here.

2. Suppose the position (t_k, q) is played. We know that the test $\mathsf{type}_2(s_n)$? succeeds on t_k and that $q \in l(s_n)$. Define the marking:

$$\overline{m}(a, q') := \{x \mid t_k R_a x \Vdash \phi_{q'}\}$$

   We can again prove that this is a legal marking. The proof is simpler than the one above, as no more switching is necessary.

   A response of the Spoiler to this marking leads to a position on which the Duplicator can win. □

This completes the proof of theorem 3.4. □

Corollary 3.5 *An MSO-formula is safe for bisimulation iff it is equivalent to a closed μ-program.*

Proof.
From right to left is just a matter of induction over closed μ-programs, using the fact that μ-sentences are invariant for bisimulation.

From left to right, suppose $\phi(x, y)$ is safe for bisimulation. Then $\exists y.(\phi(x, y) \wedge p(y))$ is invariant for bisimulation, where p is some fresh propositional constant. By the invariance theorem of Janin and Walukiewicz, theorem 2.4, this formula must be equivalent to a μ-sentence $\psi(p)$. This sentence will be completely additive in p, hence by theorem 3.4, $\psi(p)$ will be equivalent to $\langle\pi\rangle p$, where p does not occur in π. It is now easy to show that π is equivalent to $\phi(x, y)$. □

Corollary 3.6 *The same holds when we restrict to the class of image-finite models.*

Proof.
What we mean by restricting to image-finite models should be clear: $\phi(x, y)$ is safe for bisimulation in IMF (the class of image-finite transition systems) iff whenever we have a bisimulation between two image-finite models it is also a bisimulation w.r.t. $\phi(x, y)$, and likewise equivalence is restricted to equivalence in all image-finite models.

Now the proof is the same as in the class of all models, because theorem 2.4 also holds when restricted to image-finite models and the only construction on models that is used in the proof of theorem 3.4 is 2-unraveling and this preserves image-finiteness. □

4 Modal automata and safety in FOL

The original characterizations of fragments of logics that are invariant or safe for bisimulation were investigated in the context of first-order logic (FOL) ([3, 4]). The FOL-formulas that are invariant for bisimulation turn out to be exactly the *modal formulas*: all μ-calculus sentences that do not contain any μ-operators. The FOL-formulas that are safe for bisimulation are exactly those in the following language, with the obvious interpretation:

$$\pi ::= a \mid \phi? \mid \pi; \pi \mid \pi \cup \pi$$

where $a \in A$ and where ϕ ranges over modal formulas. Let us call formulas in this language **modal programs**. So a modal program is just a μ-program with only tests for modal formulas and without iteration. In this section we give a new proof of the safety theorem in FOL, modeling it after the proof we gave in the previous section for MSO.

Definition 4.1 *An automaton $\mathcal{A} = (Q, \Sigma, \Xi, q_0, \delta, \Omega)$ is* **modal** *if:*

1. *If $(a, q') \in D \in \delta(q, L)$ then $\Omega(q') = \Omega(q) = 0$ or $\Omega(q') < \Omega(q)$;*

2. *If $\Omega(q) = 0$ and $\delta(q, L) \neq \emptyset$, then for all $\Xi' \subseteq \Xi$ there exists a true state $\overline{q}$ (see definition 2.6) such that $D = \{(a, \overline{q}) : a \in \Xi'\} \in \delta(q, L)$.*

The definition of modal automata embodies the fact that modal formulas can only *see* up to a fixed finite depth, given by the depth of the nesting of modal operators in the formula. The parity condition Ω is irreverently used here to count down towards the point at which the modal formula the automaton simulates becomes blind. At this stage a true state is reached: the automaton will accept anything. An alternative definition would not involve the parity function at all (equivalently: always set it to 0) but impose the bounded vision of modal automata externally. But the parity function is there, so why not (ab)use it?

The following theorem is just a result of the design of modal automata. A proof may be found in [9].

Theorem 4.2 (D'Agostino and Hollenberg) *Every modal formula corresponds to a modal automaton and vice versa.*

Theorem 4.3 *If a modal formula $\phi(p)$ distributes over finite disjunctions then it is equivalent to a formula $\langle\pi\rangle p$, where π is a p-free modal program.*

Proof.
The proof is analogous to the proof of theorem 3.4: we will focus on the differences but use some of the same terminology.

Let $\mathcal{A}$ be a modal automaton corresponding to $\phi(p)$. The notion of a path-type is the same as before. If $\tau = (\mathsf{type}_0, a_1, d_1, \mathsf{type}_1, \ldots, a_n, d_n, \mathsf{type}_n)$ is such a type then define π_τ to be the following modal program:

$$\mathsf{type}_0?; a_1; \mathsf{type}_1?; \ldots; a_n; \mathsf{type}_n?$$

Note that as $\mathcal{A}$ is a modal automaton, every 2-type calculated with respect to it is in fact a modal formula. As $\phi(p)$ is modal there is an $n \in \mathbb{N}$ such that only points up to depth $\leq n$ are of relevance to the truth of $\phi(p)$. We will prove that:

$$\phi(p) \equiv \bigvee \{\langle\pi_\tau\rangle p \mid \tau \text{ is a path-type of length at most } n \text{ and } \langle\pi_\tau\rangle p \models \phi(p)\ \}$$

Suppose $(\mathcal{M}, P), s \Vdash \phi(p)$, where $(\mathcal{M}, P)$ is 2-unraveled and s its root. If we restrict P to $P' := \{x \in P \mid x \text{ lies at a distance } \leq n \text{ from } s\}$ then $\phi(p)$ remains true: $(\mathcal{M}, P'), s \Vdash \phi(p)$. Now partition P' into sets P_τ as before, but only for path-types τ of length at most n. There are only finitely many types of at most this length, so we have a finite partition of P'. By the assumption that $\phi(p)$ distributes over finite unions, there must be a type τ of at most length n such that $(\mathcal{M}, P_\tau), s \Vdash \phi(p)$.

Trivially $(\mathcal{M}, P_\tau), s \Vdash \langle\pi_\tau\rangle p$. What we still need to prove is that $\langle\pi_\tau\rangle p$ implies $\phi(p)$. But the proof of this is exactly the same as that of Claim 3 of theorem 3.4. □

So modal formulas that distribute over finite unions in fact distribute over arbitrary unions: they are completely additive. This means that checking whether a formula $\phi(p)$ is completely additive can be done entirely within the modal calculus: $\phi(p)$ is completely additive iff the following two are theorems of the minimal modal logic K:

$$\begin{array}{rcl} \phi(p \vee q) & \equiv & \phi(p) \vee \phi(q) \\ \phi(\bot) & \equiv & \bot \end{array}$$

Corollary 4.4 *A FOL formula $\phi(x, y)$ is safe for bisimulation iff it is equivalent to a modal program.*

Corollary 4.5 *The same is true if we restrict to the class of finite models.*

Proof.
Suppose $\phi(x, y)$ is safe for bisimulation in FIN, the class of finite models. Then $\exists y.(\phi(x, y) \wedge p(y))$

(where p is fresh) is invariant for bisimulation in FIN. By a result of Rosen ([30]) the latter formula must be equivalent (in FIN) to a modal formula $\psi(p)$. $\psi(p)$ must be completely additive in p (again, when restricting to finite models). So the following are valid equivalences in FIN: $\psi(p \vee q) \equiv \psi(p) \vee \psi(q)$ and $\psi(\bot) \equiv \bot$. Due to the finite model property of modal logic (proved for instance in [17]), equivalence in FIN and in the class of all models coincides. So by theorem 4.3, $\psi(p)$ must be equivalent to a formula $\langle\pi\rangle p$, where π is a p-free modal program. This π must then be FIN-equivalent to $\phi(x, y)$. $\square$

5 Further research

The question of characterizing fragments of logics that are safe for bisimulation was originally motivated by PDL (cf. [4]). The original question was: can we semantically characterize the class of programs of PDL? This is still an open problem: the safe fragment of MSO is strictly larger than the class of programs of PDL and the safe fragment of FOL is strictly smaller.

Future research could concentrate on finding an interesting language (i.e. not PDL itself) such that its safe fragment *does* correspond to the class of PDL-programs. Gerard Renardel has suggested to consider a fragment of $\mathcal{L}_{\omega_1\omega}$ that only has conjunctions and disjunctions over certain effective sets. A candidate for such a fragment is *weak* MSO, where second-order quantification is restricted to finite sets.

Another question suggested by the previous section and Rosen's work ([30]) is whether the invariant fragment of MSO also coincides with the μ-calculus in the class of finite models. Even if this is so, the proof of safety would not automatically transfer, because complete additivity of μ-formulas does not coincide with distributivity over finite unions.

Finally, all of these questions could be posed for other notions of process equivalence than bisimulation (see [1, 14, 15]), such as branching bisimulation ([16]). The task is easier in these cases, it seems, because bisimulation subsumes a whole range of process-equivalences. Thus if a certain MSO formula is invariant for branching bisimulation, say, then it must also be invariant for bisimulation, hence it must be equivalent to a μ-calculus formula. So the question reduces to: which μ-sentences are invariant for branching bisimulation?

Notions of safety would have to be modified to fit the appropriate notion of process-equivalence. Branching bisimulation has different clauses for different kinds of binary relations: one for *silent* actions and one for normal actions. This automatically gives us *two* notions of safety for branching bisimulation.

Acknowledgement: The author wishes to thank Igor Walukiewicz for an enlightening discussion on this subject, and Johan van Benthem and Yde Venema for carefully reading and commenting on earlier versions of this paper.

References

[1] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990, 1994.

[2] J.F.A.K. van Benthem. *Modal Correspondence Theory*. PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[3] J.F.A.K. van Benthem. *Language in Action. Categories, Lambdas and Dynamic Logic*, volume 130 of *Studies in Logic*. North-Holland, Amsterdam, 1991.

[4] J.F.A.K. van Benthem. Programming operations that are safe for bisimulation. CSLI Report 93-179, Center for the Study of Language and Information, Stanford University, 1993. To appear in *Studia Logica*.

[5] J.F.A.K. van Benthem. Safety for bisimulation in infinitary logic. Unpublished manuscript, 1996.

[6] J.F.A.K. van Benthem and J.A. Bergstra. Logic of transition systems. Report P9308, Programming Research Group, University of Amsterdam, 1993.

[7] J. Bradfield. The modal μ-calculus alternation hierarchy is strict. To appear in the Proceedings of CONCUR'96, 1996.
`http://www.dcs.ed.ac.uk/~jcb/Research/althi.ps.gz`.

[8] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems Using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[9] G. D'Agostino and M.J. Hollenberg. Uniform interpolation, automata and the modal μ-calculus. Logic Group Preprint Series 165, Dept. of Philosophy, Utrecht University, December 1996.
`ftp://ftp.phil.ruu.nl/pub/logic/PREPRINTS/preprint165.ps.Z`.

[10] E.A. Emerson and J.Y. Halpern. Sometimes and not never revisited: on branching versus linear time. *Journal of the ACM*, 33(1):151–178, 1986.

[11] E.A. Emerson and C.S. Jutla. Tree automata, μ-calculus and determinacy. In *Proceedings FOCS'91*, 1991.

[12] E.A. Emerson and Chin-Laung Lei. Modalities for model checking: branching time logic strikes back. *Science of Computer Programming*, 8:275–306, 1987.

[13] M.J. Fisher and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[14] R.J. van Glabbeek. The linear time - branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR '90*, volume 458 of *LNCS*, pages 278–297. Springer Verlag, 1990.

[15] R.J. van Glabbeek. The linear time - branching time spectrum II: the semantics of sequential processes with silent moves. In E. Best, editor, *Proceedings CONCUR'93*, volume 715 of *LNCS*, pages 66–81. Springer Verlag, 1993.
`http://theory.stanford.edu/~rvg/abstracts.html#26`.

[16] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89: Proceedings of the IFIP 11th World Computer Congress, San Fransisco, 1989*, pages 613–618. Elsevier Science Publishers B.V., North Holland, 1989.

[17] R. Goldblatt. *Logics of Time and Computation. Second Edition*, volume 7 of *CSLI Lecture Notes*. CSLI Publications, 1992.

[18] R. Goré. Tableau methods for modal and temporal logics. To appear in M. D'Agostino, D. Gabbay, R. Hähnle and J. Posegga, editors, *Handbook of Tableau Methods*, Kluwer, Dordrecht, due in 1996.
`http://arp.anu.edu.au/pub/techreports/1995/TR-ARP-15-95.ps.gz`.

[19] D. Harel. Dynamic logic. In D.M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic, Vol. II*, pages 497–604. Reidel, Dordrecht, 1984.

[20] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. Assoc. Comput. Mach.*, 32:137–161, 1985.

[21] D. Janin. *Propriétés Logiques du NonDéterminisme et μ-Calcul Modal.* PhD thesis, LaBRI – Université de Bordeaux I, 1996.
`http://www.labri.u-bordeaux.fr/~janin/these/these.ps`.

[22] D. Janin and I. Walukiewicz. Automata for the modal μ-Calculus and related results. In *Proceedings MFCS'95*, pages 552–562. Springer, 1995.
`http://www.daimi.aau.dk/~igw/mfcs.ps`.

[23] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus w.r.t. monadic second-order logic. To appear in the Proceedings of CONCUR'96, 1996.
`http://www.labri.u-bordeaux.fr/~janin/Rech/concur_96.html`.

[24] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:234–241, 1983.

[25] G. Lenzi. A hierarchy theorem for the μ-calculus. In J. van Leeuwen G. Goos, J. Hartmanis, editor, *Proceedings ICALP'96*, volume 1099 of *LNCS*, 1996.

[26] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In A. Skowron, editor, *5th Symposium in Computation Theory*, volume 208 of *LNCS*, pages 157–168, 1984.

[27] D. Niwiński and I. Walukiewicz. Games for the μ-calculus. *Theoretical Computer Science*, 163(1-2):99–116, 1996.

[28] D. Park. Concurrency and automata on infinite sequences. In *Proceedings 5th GI Conference*, pages 167–183. Springer, 1981.

[29] V. Pratt. Semantical considerations on floyd-hoare logic. In *Proceedings 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[30] E. Rosen. Modal logic over finite structures. Report ML-95-08, Institute for Logic, Language and Computation, University of Amsterdam, 1995.
`ftp://ftp.cis.upenn.edu/pub/ircs/tr/95-27.ps.Z`.

[31] C. Stirling. Modal and temporal logics. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic for Computer Science, Vol. 2*, pages 477–563. Reidel, Dordrecht, 1992.