On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic

David Janin
LaBRI[1]
Université de Bordeaux I
351, Cours de la Libération
F-33 405 Talence cedex, France
e-mail: janin@labri.u-bordeaux.fr

Igor Walukiewicz
BRICS[2,3]
Department of Computer Science
University of Aarhus
Ny Munkegade
DK-8000 Aarhus C, Denmark
e-mail: igw@mimuw.edu.pl

Abstract. Monadic second order logic (MSOL) over transition systems is considered. It is shown that every formula of MSOL which does not distinguish between bisimilar models is equivalent to a formula of the propositional μ-calculus. This expressive completeness result implies that every logic over transition systems invariant under bisimulation and translatable into MSOL can be also translated into the μ-calculus. This gives a precise meaning to the statement that most propositional logics of programs can be translated into the μ-calculus.

1 Introduction

Transition systems are structures consisting of a nonempty set of *states*, a set of unary relations describing properties of states and a set of binary relations describing *transitions* between states. It was advocated by many authors [26, 3] that this kind of structures provide a good framework for describing behaviour of programs (or program schemes), or even more generally, engineering systems, provided their evolution in time is discrete.

Take as an example an operational semantics of a (scheme of a) programming language, say CCS. It is given in two steps. First one associates with every program a transition system describing all possible executions of the program. This can be done using SOS rules [25] or similar formalism. Next one defines an equivalence relation between transition systems which depends on the intended notion of observable. The meaning of the program is an equivalence class of the associated transition system. Bisimulation relation is often considered to be the finest relation which is interesting in this context [22, 30].

[1] **La**boratoire **B**ordelais de **R**echerche en **I**nformatique (CNRS URA 1304)
[2] **B**asic **R**esearch **i**n **C**omputer **S**cience, Centre of the Danish National Research Foundation.
[3] On leave from: Institute of Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, POLAND

In the setting described above, checking that a modelled system (say a program) has some property amounts to checking that the corresponding transition system has the property. The fact that the meaning of the program is really an equivalence relation and not a transition system itself is reflected in the fact that the properties we are interested in do not distinguish between equivalent systems. This motivates our claim that logics suitable for verification should not distinguish between bisimilar systems. Indeed most of the program logics proposed in the literature have this property.

Since this approach to verification has been suggested [26] a big variety of logics over transition systems has been proposed. New logics were introduced because they were more manageable, more expressive, or represented a better balance between these two kinds of properties. Manageability is concerned with axiomatisations, the complexity of the validity problem, and the complexity of the model checking problem (i.e. a problem of verifying whether a given formula is satisfied in a given transition system). These complexity issues are important especially for computer aided verification. Of course there is no point in considering even very manageable logic if it is not capable of expressing the properties we are interested in.

The question arises: how one does decide which properties are interesting. Of course it is important to list first some example properties which one would like to express (see [13, 10] for such lists), but how can one be sure that we have listed all the properties of potential interest? The solution is to find a "yardstick" which is usually some well established logic. If we can express all the properties from our list in the "yardstick" logic then we know that the set of properties expressible in the logic is complete, in a sense that it is closed under logical operations and contains our interesting properties. This approach was initiated by Kamp [20] who investigated expressive power of propositional temporal logic (PTL) with respect to expressive power of the first order logic over $\langle \omega, \leq \rangle$. This lead him to the discovery of the *until* operator and the proof that PTL with the *until* operator is *expressively complete* with respect to first order logic, i.e.: a class of models is definable by a PTL formula with *until* iff it is definable in the first order logic. PTL is still widely used but it turned out that there are interesting properties which are not expressible in the first order logic and MSOL over $\langle \omega, \leq \rangle$ was proposed as a new yardstick. This choice was particularly useful as it brought new insights and a wealth of automata theoretic methods to the field. The logic expressively complete with respect to MSOL over $\langle \omega, \leq \rangle$ is the μ-calculus of linear time [4].

As noted by Emerson ([10] p. 1026) the situation for branching time logics is not so well understood. Known results are limited to transition systems which are binary trees. For this restricted class of models the yardstick is MSOL theory of the binary tree (*S2S*). It is known that the binary μ-calculus is expressively complete with respect to full *S2S* [24, 11] and CTL* is expressively complete with respect to the fragment of *S2S* where only quantification over paths is allowed [16].

As far as we are aware, the only expressive completeness results dealing

with the general case of logics over all transition systems were given by van Benthem [5] and van Benthem and Bergstra [6]. They show that a bisimulation closed class of transition systems is definable in first order logic (resp. in infinitary first order logic) iff it is definable in (system K) modal logic (resp. infinitary modal logic). For these results to hold it is essential that one admits disconnected transition systems. This makes properties like "there is a transition from every node" not closed under bisimulation. These kind of properties also show that these results are not true when restricted only to connected transition systems. Expressive completeness of temporal logic with respect to first order logic over various kinds of orders was investigated among others in [14, 2].

From Gaifman's characterisation of expressive power of first order logic over transition systems [15] it follows that first order logic is not a very interesting logic from a verification point of view. In our opinion the proper yardstick for logics over transition systems should be MSOL, or rather, bisimulation invariant properties expressible in MSOL. This choice is motivated by the fact that it is a very expressive logic, capable of expressing most of the properties considered in the literature. Moreover the set of properties of MSOL is closed under quantification over sets which makes it possible to express for example: path quantification, reachability, least and greatest fixpoints of the properties.

Let us briefly comment why *S2S* is not a good candidate for defining an expressibility standard over transition systems of arbitrary degree. It is of course possible to code every countable transition system into a binary tree but this comes with a price. Any such coding introduces an ordering between siblings which is not available in the original structure. This order allows even a very weak logics over binary trees to express properties of codings not expressible in MSOL over transition systems (see [28] p. 540 for an example).

Finally observe that MSOL over trees of arbitrary degree is very different from monadic second order theory of ω-successors ($S\omega S$). In the later theory even the relation "x is a son of y" is not definable (one would need an infinite formula to do this).

1.1 Synopsis

Our main result is that, every bisimulation closed MSOL definable property of transition systems is definable in the propositional μ-calculus. This shows that among all possible behavioural specification languages whose semantics is expressible in MSOL over transition systems, the μ-calculus is the most expressive one. In particular, this immediately shows that CTL^* and $ECTL^*$ are translatable into the μ-calculus [9] since these logics are easily translatable into MSOL over unwindings of transition systems and formulas resulting from the translation are bisimulation closed, hence invariant under unwinding operation.

Maybe an interesting aspect of this result is that the set of MSOL formulas closed under bisimulation is not recursive (it is even not arithmetical). On the other hand it turns out to be decidable whether a MSOL formula defines a bisimulation closed set of trees. Let us also remark that unlike van Benthem and

Bergstra's results mentioned above our expressibility result also holds when we restrict to connected transition systems or even finite branching trees.

The main tools we will be using are recently developed automata characterisations of the μ-calculus [19] and MSOL [31] over trees. It turns out that there is a more general notion of automata of which both characterisations are special cases. This gives us a common ground to compare the two logics.

The paper is organised as follows. We start with the section introducing transition systems and the bisimulation relation. We also introduce there a notion of ω-expansion which we will need in the main proof. Next we give definitions of MSOL and the μ-calculus. In Section 4 we define a general notion of automaton and give characterisations of MSOL and the μ-calculus in terms of these automata. These characterisations are used in the following section where we prove our main result.

2 Transition systems and bisimulation

Let $Pred = \{p, p', \ldots\}$ be a set of unary predicate symbols and let $Rel = \{r, r', \ldots\}$ be a set of binary predicate symbols. A *transition system with a source*, simply called *transition system* in the sequel, is a tuple:

$$M = \langle S^M, sr^M, \{r^M\}_{r \in Rel}, \{p^M\}_{p \in Prop} \rangle$$

where: S_M is a nonempty set of *states*; $sr_M \in S$ is a *source*; each r^M is a binary relation on S_M and each p^M is a subset of S^M.

For every $r \in Rel$, let:

$$succ_r^M(s) = \{s' \in S^M \mid (s, s') \in r^M\}$$

Transition system M is called a *transition tree* (or simply a tree) if for every state $s \in M$ there exists a unique path to the root, or more formally there exists a unique sequence $s_0, \ldots, s_n$ such that $s_0 = sr^M$, $s_n = s$ and for every $i = 1, \ldots, n$ we have $(s_i, s_{i+1}) \in r_i^M$ for some $r_i \in Rel$.

Transition systems M and N are called *bisimilar* when there exists a relation $R \subseteq S^M \times S^N$, called a *bisimulation relation*, such that $(sr^M, sr^N) \in R$ and for every $(s, t) \in R$, $p \in Prop$ and $r \in Rel$:

- $s \in p^M$ iff $t \in p^M$,
- whenever $(s, s') \in r^M$ for some s', then there exists t' such that $(t, t') \in r^N$ and $(s', t') \in R$,
- whenever $(t, t') \in r^N$ for some t', then there exists s' such that $(s, s') \in r^M$ and $(s', t') \in R$.

Definition 1 ω-expansion. Given a transition system M, an *ω-indexed path* of M is a sequence u of the form:

$$u = s_0(a_1, r_1, s_1)(a_2, r_2, s_2) \cdots (a_n, r_n, s_n)$$

where $s_0 = sr^M$, $a_i \in I\!N$ and $(s_{i-1}, s_i) \in r_i^M$ for $i = 1, \ldots, n$.

The *ω-expansion* $\widehat{M}$ of the system M is defined by :

1. $S^{\widehat{M}}$ is the set of ω-indexed paths of M,
2. $sr^{\widehat{M}} = sr^M$,
3. for every $r \in Rel$, every u and $v \in S^{\widehat{M}}$: $(u,v) \in r^{\widehat{M}}$ iff v is an ω-indexed path of the form $u(a,r,s)$, for some a and s,
4. for every $p \in Prop$:

$$p^{\widehat{M}} = \{u(a,r,s) : s \in p^M, \quad u,a,r \text{ arbitrary}\} \cup \{sr^{\widehat{M}} : sr^M \in p^M\}$$

In the rest of this section let us briefly point out how the concept of ω-expansion arises from a general consideration about bisimulation relation.

Definition 2. Given two transition systems M and N, we say that M is an *expansion* of N, denoted $M \succeq N$, when there exists a partial function $h : S_M \to S_N$ such that :

1. $h(sr^M) = sr^N$,
2. for every $s \in S^M$, $p \in Prop$ and $r \in Rel$:

$$s \in p^M \iff h(s) \in p^N \text{ and } h(succ_r^M(s)) = succ_r^N(h(s))$$

Remark. In [8], with distinct notations and names, Castellani shows that M_1 and M_2 are bisimilar iff there exists N such that $N \preceq M_1$ and $N \preceq M_2$. Intuitively N is a quotient of M_1 and M_2 under bisimulation relation, henceforth a minimal representative. Next fact states that ω-expansions are, in the countable case, maximal representatives of behaviours.

Fact 3. *Considering only transition systems with at most countably many states: for every transition system M we have $M \preceq \widehat{M}$, and, for every transition system N, if M and N are bisimilar then $\widehat{M}$ and $\widehat{N}$ are isomorphic.*

3 Monadic second order logic and the propositional μ-calculus

In this section we will define monadic second order logic (MSOL) and the propositional μ-calculus [21]. Both logics will be interpreted over transition systems of the signature containing only unary symbols from $Prop$ and binary symbols from Rel. These sets were fixed at the beginning of the previous section. Let $Var = \{X, Y, \ldots\}$ be a countable set of (second order) *variables*.

3.1 MSOL

Monadic second order logic over the signature $\{Rel, Prop\}$ and constant sr can be defined as follows. The set of MSOL formulas is the smallest set containing formulas:

$$p(X),\ r(X,Y),\ X \subseteq Y,\ sr(X)$$

for $p \in Prop$, $r \in Rel$, $X, Y \in Var$; and closed under negation, disjunction and existential quantification. A *sentence* is a formula without free variables.

The definition of the truth of a formula in a given transition system M and a valuation $V : Var \to \mathcal{P}(S^M)$ is defined by induction on the length of the formula:

$$\begin{array}{ll}
M, V \models p(X) & \text{iff } V(X) \subseteq p^M \\
M, V \models r(X, Y) & \text{iff } V(X) = \{s\}, V(Y) = \{t\} \text{ and } (s, t) \in r^M \\
M, V \models sr(X) & \text{iff } V(X) = \{sr^M\} \\
M, V \models X \subseteq Y & \text{iff } V(X) \subseteq V(Y) \\
M, V \models \alpha \vee \beta & \text{iff } M, V \models \alpha \text{ or } M, V \models \beta \\
M, V \models \neg\alpha & \text{iff not } M, V \models \alpha \\
M, V \models \exists X.\ \alpha(X) & \text{iff there is } T \subseteq S^M \text{ s.t. } M, V[T/X] \models \alpha(X)
\end{array}$$

We will concentrate here on definability by sentences. Of course it makes no difference for MSOL because the quantification is available, but it will make the difference in the case of the μ-calculus. We write $M \models \varphi$ to mean that the sentence φ is true in M. A sentence φ of MSOL defines a class of transition systems: $\{M : M \models \varphi\}$. A class of transition systems is *MSOL definable* if there exists an MSOL sentence defining this class. A class $\mathcal{C}$ of transition systems is *bisimulation closed* if whenever $M \in \mathcal{C}$ and M' is bisimilar to M then $M' \in \mathcal{C}$. A sentence is *bisimulation invariant* if the class of transition systems it defines is bisimulation closed.

Remark. There exist formulas of MSOL which are not bisimulation invariant. Take for example a formula stating that there is exactly one r–transition from the source. Observe that the problem of checking whether an MSOL formula is bisimulation invariant is not arithmetical because the validity problem is not arithmetical.

3.2 Propositional μ-calculus

The set of the μ-calculus formulas is the smallest set containing $Prop \cup Var$ which is closed under negation, disjunction and the following two formation rules:

- if α is a formula and $r \in Rel$ then $\langle r \rangle \alpha$ is a formula,
- if $\alpha(X)$ is a formula and X occurs only positively (i.e. under even number of negations) in $\alpha(X)$ then $\mu X.\alpha(X)$ is a formula.

Observe that we use relation names in the modalities.

The meaning of a formula α in a transition system M and a valuation $V : Var \to \mathcal{P}(S^M)$ is a set of states, $\|\alpha\|_V^M$, where it is true. It is defined by induction on the length of the formula:

$$\begin{array}{ll}
\|p\|_V^M & = p^M \\
\|\neg\alpha\|_V^M & = S^M - \|\alpha\|_V^M \\
\|\alpha \vee \beta\|_V^M & = \|\alpha\|_V^M \cup \|\beta\|_V^M \\
\|\langle r \rangle \alpha\|_V^M & = \{s : \exists t.(s, t) \in r^M \wedge t \in \|\alpha\|_V^M\} \\
\|\mu X.\alpha(X)\|_V^M & = \bigcap \{T \subseteq S^M : \|\alpha(X)\|_{V[T/X]} \subseteq T\}
\end{array}$$

For a sentence φ we write $M, s \models \varphi$ when $s \in \|\varphi\|_V^M$ (the choice of a valuation V is irrelevant as φ is a sentence). A sentence φ of the μ-calculus defines a class of transition systems $\{M : M, sr^M \models \varphi\}$. The class of transition systems is *μ-definable* if there exists a μ-calculus sentence defining this class. It is well known that:

Fact 4. *Every μ-definable class is bisimulation closed.*

Remark. Let us comment on the fact that we consider only definability by sentences. Call a class $\mathcal{C}$, μ-f-definable (μ-formula-definable) if there is a formula φ of the μ-calculus such that:

$$\mathcal{C} = \{M : sr^M \in \|\varphi\|_V^M \text{ for arbitrary } V : Var \to S^M\}$$

There are μ-f-definable classes which are not closed under bisimulation. Consider for example the class defined by the formula $\neg(\langle r\rangle X \wedge \langle r\rangle \neg X)$. This formula defines a class of structures M where there is at most one $s \in S^M$ such that $(sr^M, s) \in r^M$. This class is clearly not bisimulation closed. The μ-f-definability corresponds to definability of frames in modal logic. It is easy to see that the notion of μ-f-definability is not closed under complement. Hence this notion of definability is not interesting from expressive completeness point of view.

4 Automata characterisations

Here we will define automata running on transition systems. Then we will give characterisations of the expressive power of MSOL and the μ-calculus in terms of these automata.

First problem we have to deal with is the description of a transition function. In case of words, a transition function of an automaton with alphabet Σ and states Q is an element of $Q \times \Sigma \to \mathcal{P}(Q)$. In case of binary trees, it is an element from $Q \times \Sigma \to \mathcal{P}(Q \times Q)$. This suggest that for trees of degree less than or equal to κ, a transition function should be an element of $Q \times \Sigma \to \mathcal{P}(Q^\kappa)$. But surely it cannot be an arbitrary such function because MSOL has limited expressive power. The idea is to shift the attention a little. Let us consider the set S of sons of a node. An assignment of states to the elements of S can be seen as a function $m : Q \to \mathcal{P}(S)$, which for each state $q \in Q$ gives a set of elements to which q is assigned. We call such a function a *marking*. The set of markings can be described by a formula with free second order variables $\{Z_q\}_{q \in Q}$ representing the sets of elements assigned q. For example in case of binary trees, S will be always a two element set $\{l, r\}$ and a transition, say, $\delta(q, a) = \{(q_1, q_2), (q_3, q_4)\}$ will be translated into the formula: $(Z_{q_1}(l) \wedge Z_{q_2}(r)) \vee (Z_{q_3}(l) \wedge Z_{q_4}(r))$. This approach extends easily to alternating automata on binary trees [23] but this time formulas obtained in the translation will be arbitrary positive boolean combinations of atomic formulas of the from $Z_{q_i}(l)$ or $Z_{q_i}(r)$. In the case of trees of arbitrary degree the use of formulas to describe markings allows us to abstract from the cardinality of S. By restricting to specific classes of formulas we can control the

expressive power of the obtained automata. It turns out that this gives us enough control to characterise the μ-calculus or MSOL over trees. Hence we obtain a common ground to compare the two logics.

Let us proceed with the formal definition of these automata.

Definition5 Basic formulas. For every finite set $\mathcal{U}$, let $\mathrm{BF}(\mathcal{U})$ be some set of sentences of the first order logic, possibly with the equality predicates, over the signature consisting of unary predicates $\{p\}_{p\in\mathcal{U}}$. A marking of a given set S is a function $m : \mathcal{U} \to \mathcal{P}(S)$. We say that m satisfies a sentence $\varphi \in \mathrm{BF}(\mathcal{U})$ iff φ is satisfied in the structure $\langle S, \{m(p)\}_{p\in\mathcal{U}}\rangle$, i.e., the structure with the carrier S and each predicate $p \in \mathcal{U}$ interpreted as $m(p)$.

An automaton is a tuple:

$$\mathcal{A} = \langle Q, \Sigma_p \subseteq Prop, \Sigma_r \subseteq Rel, q_0 \in Q, \\ \delta : Q \times \mathcal{P}(\Sigma_p) \to \mathrm{BF}(\Sigma_r \times Q), \Omega : Q \to I\!N\rangle \tag{1}$$

where Q is a finite set of states, Σ_p is a *finite* subset of $Prop$ and Σ_r is a *finite* subset of Rel. Observe that the automaton has two alphabets. One is for examining properties of states and the other is for checking labels of taken transitions.

We find it convenient to give the definition of acceptance in terms of games.

Definition6 Acceptance. Let M be a transition system and let $\mathcal{A}$ be an automaton as above. We define a game $G(M, \mathcal{A})$ as follows:

- The initial position is a pair (sr^M, q_0).
- If the current position is a pair (s, q) then player I is to move. Let

  $$L(s) = \{p \in Prop : s \in p^M\} \cap \Sigma_p$$

  be a set of relevant propositions holding in s. Player I chooses a marking $m : \Sigma_r \times Q \to \mathcal{P}(\bigcup_{r\in\Sigma_r} succ_r(s))$ such that for every r, q we have $m(r, q) \subseteq succ_r(s)$ and

  $$\langle \bigcup_{r\in\Sigma_r} succ_r(s), \{m(r, q)\}_{(r,q)\in\Sigma_r\times Q}\rangle \models \delta(q_i, L(s))$$

  The marking m becomes the current position.
- If the current position is a marking m then player II chooses some $r \in \Sigma_r$, some automaton state $q \in Q$ and some state $s \in m(r, q)$. The pair (s, q) becomes the current position.

If one of the players cannot make a move then the other player wins. If the play is infinite then as the result we obtain an infinite sequence:

$$(s_0, q_0), m_1, (s_1, q_1), m_2, \ldots$$

Let j be the smallest number appearing infinitely often in the sequence:

$$\Omega(q_0), \Omega(q_1), \ldots$$

Player I wins if j is even, otherwise player II is the winner.

We say that M is *accepted* by $\mathcal{A}$ iff there is a winning strategy for player I in the game $G(M, \mathcal{A})$. A *language* recognised by $\mathcal{A}$ is the class of transition systems accepted by $\mathcal{A}$.

From now on, let $\mathcal{U} = \Sigma_r \times Q$. The following is a reformulation of a result from [19].

Theorem 7. *A class of transition systems is definable by a μ-calculus sentence iff it is a language recognised by an automaton as in (1) with $BF(\mathcal{U})$ containing only disjunctions of sentences of the form:*

$$\exists x_1, \ldots, x_k.\ (p_1(x_1) \wedge \ldots \wedge p_k(x_k) \wedge \forall z. p_1(z) \vee \ldots \vee p_k(z)) \tag{2}$$

where $p_i \in \mathcal{U}$ for $i = 1, \ldots, k$.

The goal in [19] was to find the simplest possible form of automaton. Here we will be content with more liberal formalisation. The proof of the fact below can be found in [18], it also follows from [31].

Fact 8. *A class of transition systems is definable by some μ-calculus formula iff it is a language recognised by an automaton as in (1) with $BF(\mathcal{U})$ containing only disjunctions of formulas of the form:*

$$\exists x_1, \ldots, x_k.\ (p_1(x_1) \wedge \ldots \wedge p_k(x_k) \wedge \forall z. \beta(z))$$

where $\beta(z)$ is a disjunction of conjunctions of formulas of the form $p(z)$ for $p \in \mathcal{U}$.

Example 1. As an example we construct an automaton equivalent to the μ-calculus formula $\mu X. p \vee \langle r \rangle X$. This automaton is:

$$\langle \{q\}, \{p\}, \{r\}, q, \delta, \Omega \rangle$$

where $\Omega(q) = 1$ and δ is defined by:

$$\begin{array}{ll} \delta(q, \emptyset) = & \exists x.\ (r, q)(x) \wedge \forall z. true \\ \delta(q, \{p\}) = & \forall z.\ true \end{array}$$

The following was shown in [31]:

Theorem 9. *A class of trees is definable by a MSOL sentence iff it is a language recognised by an automaton as in (1) with $BF(\mathcal{U})$ containing only disjunctions of formulas of the form:*

$$\exists x_1, \ldots, x_k.\ diff(x_1, \ldots, x_k) \wedge p_{i_1}(x_1) \wedge \cdots \wedge p_{i_k}(x_k) \wedge \forall z.\ diff(z, x_1, \ldots, x_k) \Rightarrow \beta(z)$$

where $\beta(z)$ is a disjunction of conjunctions of formulas of the form $p(z)$, for $p \in \mathcal{U}$, and $diff(x_1, \ldots, x_k)$ is a formula stating that the meanings of all the variables are different.

Remark. In the above theorem we can allow arbitrary first or even monadic second order formulas as basic formulas. The set of basic formulas specified above is the smallest set which was shown to be sufficient in [31]. Of course, the simpler the set of basic formulas, the easier would be our task of translating MSOL formulas into μ-calculus formulas.

Because the construction of an automaton equivalent to a given formula is effective and because the emptiness problem for these automata can be shown to be decidable we obtain:

Corollary 10. *MSOL theory of trees is decidable.*

For countably branching trees this corollary is a consequence of Rabin's theorem about decidability of *S2S* [27].

5 Expressive completeness

Theorem 9 together with Fact 8 suggest that there is a very strong connection between the two logics. Basic formulas in case of MSOL automata are more expressive because, for example, they can compare the number of sons with a constant (by the use of existential quantification together with $\mathit{diff}(\vec{x})$ formula). Intuitively if an MSOL formula is bisimulation closed, an equivalent automaton should not use $\mathit{diff}(\vec{x})$ formulas, hence it should be equivalent to a μ-calculus formula. A precise argument confirming this intuition must take into account the fact that automata are nondeterministic which means that the automaton may have only runs which use $\mathit{diff}(\vec{x})$ formulas but nevertheless accept a bisimulation closed set.

Theorem 11 Expressive completeness. *A bisimulation closed class of transition systems is MSOL definable iff it is μ-definable.*

Proof. It is easy to see that every μ-definable class is also MSOL definable. For converse we use the following lemma:

Lemma 12. *For every MSOL sentence φ one can build a μ-calculus sentence $\varphi^\vee$ such that for every transition system M:*

$$M \models \varphi^\vee \quad \textit{iff} \quad \widehat{M} \models \varphi$$

Assume that the lemma was proved. Let φ be a MSOL sentence defining a bisimulation closed class of transition systems. This in particular means that for every transition system M we have: $M \models \varphi$ if and only if $\widehat{M} \models \varphi$. Let $\varphi^\vee$ be the formula given by the lemma above. We have:

$$M \models \varphi \quad \text{iff} \quad \widehat{M} \models \varphi \quad \text{iff} \quad M \models \varphi^\vee$$

□

Proof of Lemma 12. For every formula ψ of the form:

$$\exists x_1, \ldots, x_l.\ \mathit{diff}(x_1, \ldots, x_l) \wedge p_1(x_1) \wedge \cdots \wedge p_l(x_l) \wedge \forall z.\mathit{diff}(z, x_1, \ldots, x_l) \Rightarrow \beta(z) \tag{3}$$

we let $\psi^\vee$ to be a formula obtained by substituting *true* for *diff* in the above:

$$\exists x_1, \ldots, x_l.\ p_1(x_1) \wedge \ldots \wedge p_l(x_l) \wedge \forall z.\beta(z) \tag{4}$$

For a disjunction $\theta = \psi_1 \vee \ldots \vee \psi_j$ of formulas as in 3 we define: $\theta^\vee = \psi_1^\vee \vee \ldots \vee \psi_j^\vee$.

By Theorem 9 there is an automaton $\mathcal{A} = \langle Q, \Sigma, q, \delta, \Omega \rangle$ accepting the class of tree models of φ. We know that for every $q \in Q$ and $a \in \mathcal{P}(\Sigma_p)$, formula $\delta(q, a)$ is a disjunction of formulas of the form (3). We define the automaton $\mathcal{A}^\vee$ which has all the same components as $\mathcal{A}$ but the transition function $\delta^\vee$. For every $q \in Q$ and $P \in \Sigma_p$ we let $\delta^\vee(q, P) = (\delta(q, P))^\vee$.

Observation 1. *The automaton $\mathcal{A}^\vee$ accepts M iff $\mathcal{A}$ accepts $\widehat{M}$.*

It is quite easy to see that if M is accepted by $\mathcal{A}^\vee$ then $\widehat{M}$ is accepted by $\mathcal{A}$. Conversely, suppose $\widehat{M}$ is accepted by $\mathcal{A}$. We will show that M is accepted by $\mathcal{A}^\vee$.

By the definition, $\mathcal{A}$ accepts $\widehat{M}$ iff player I has a winning strategy $\widehat{\sigma}$ in the game $\widehat{G} = G(\widehat{M}, \mathcal{A})$. We define a winning strategy $\sigma^\vee$ for player I in the game $G^\vee = G(M, \mathcal{A}^\vee)$. The idea of the strategy is to play simultaneously the games $G^\vee$ and $\widehat{G}$ and transfer each move of player II from $G^\vee$ to $\widehat{G}$. Then one can consult the strategy $\widehat{\sigma}$ for $\widehat{G}$ and transfer the suggested move of player I back to $G^\vee$.

The initial position of $G^\vee$ is (sr^M, q_0) and it is also the initial position of $\widehat{G}$.

Assume that each of the players has made k moves. Assume also that the histories of the two plays are respectively:

$$(sr^M, q_0), m^1, (s_1, q_1), \ldots, m^k, (s_k, q_k)$$

for $G^\vee$ and

$$(sr^M, q_0), \widehat{m}^1, (u_1, q_1), \ldots, \widehat{m}^k, (u_k, q_k)$$

for $\widehat{G}$, where for every $i = 1 \ldots, k$ we have $u_i = u_i'(a_i, r_i, s_i)$ for some ω-indexed path u_i', $a_i \in I\!N$ and $r_i \in \mathit{Rel}$.

In this position player I is to move. Let $\widehat{m}^{k+1} = \widehat{\sigma}(u_k, q_k)$ be a marking suggested by the strategy $\widehat{\sigma}$. Let us introduce a notation for the two structures

$$\mathcal{M}_{u_k} = \langle \bigcup_{r \in \Sigma_r} succ_r^{\widehat{M}}(u_k), \{\widehat{m}^{k+1}(r, q)\}_{(r,q) \in \Sigma_r \times Q} \rangle$$
$$\mathcal{M}_{s_k} = \langle \bigcup_{r \in \Sigma_r} succ_r^{M}(s_k), \{m^{k+1}(r, q)\}_{(r,q) \in \Sigma_r \times Q} \rangle$$

By definition the marking $\widehat{m}^{k+1} : \Sigma_r \times Q \to \mathcal{P}(succ^{\widehat{M}}(u_k))$ satisfies:

$$\mathcal{M}_{u_k} \models \delta(q_k, L(u_k)) \tag{5}$$

We define $m^{k+1} : \Sigma_r \times Q \to \mathcal{P}(succ^M(s_k))$ by letting:

$$m^{k+1}(r,q) = \{s : u_k(a,r,s) \in \widehat{m}^{k+1}(r,q) \text{ for some } a \in I\!N\} \tag{6}$$

Let us check that:

$$\mathcal{M}_{s_k} \models \delta^\vee(q_k, L(s_k))$$

We know that $\delta(q_k, L(u_k))$ is a disjunction of formulas of the form (3). By definition of $\widehat{M}$ and the fact that $u_k = u'_k(a_k, r_k, s_k)$ we have: $L(u_k) = L(s_k)$. Hence $\delta^\vee(q_k, L(s_k)) = (\delta(q_k, L(u_k)))^\vee$ by the definition of $\mathcal{A}^\vee$.

Assume that $\mathcal{M}_{u_k} \models \psi$ for some disjunct ψ of $\delta(q_k, L(u_k))$. We will show that $\mathcal{M}_{s_k} \models \psi^\vee$.

We know that $\psi^\vee$ is of the form (4) where each p_j is of the form (r_j, q_j). Let us first check that for every $j = 1, \ldots, l$ there is $s \in m^{k+1}(r_j, q_j)$. For this it is enough to take $u_k(a, r_j, s) \in \widehat{m}^{k+1}(r_j, q_j)$ known to exist by property (5). To see that $\mathcal{M}_{s_k} \models \forall z.\ \beta(z)$ observe that for every $s \in succ_r^M(s_k)$ there is $a \in I\!N$ such that $\mathcal{M}_{u_k} \models \beta((a, r, s))$. By the fact that β is monotone in predicates $\{p\}_{p \in \Sigma_r \times Q}$ and the definition of m, we obtain: $\mathcal{M}_{s_k} \models \beta(s)$.

Hence taking m is a legal move of player I in the game $G^\vee$. After this move we obtain the position:

$$(sr^M, q_0), m^1, (s_1, q_1), \ldots, m^k, (s_k, q_k), m^{k+1}$$

and at the same time in $\widehat{G}$ we obtain the position:

$$(sr^M, q_0), \widehat{m}^1, (u_1, q_1), \ldots, \widehat{m}^k, (u_k, q_k), \widehat{m}^{k+1}$$

with m^{k+1} defined from $\widehat{m}^{k+1}$ by (6). From this position player II chooses some $r_{k+1} \in \Sigma_r$, $q_{k+1} \in Q$ and a state $s_{k+1} \in m^{k+1}(r_{k+1}, q_{k+1})$. The history of the play G becomes:

$$(sr^M, q_0), m^1, (s_1, q_1), \ldots m^k, (s_k, q_k), m^{k+1}, (s_{k+1}, q_{k+1})$$

We make player II in $\widehat{G}$ to choose q_{k+1} and $u_k(a_{k+1}, r_{k+1}, s_{k+1}) \in \widehat{m}^{k+1}(r_{k+1}, q_{k+1})$ which exists by (6). We arrive at the position satisfying our initial assumptions so we can repeat the whole argument.

By definition of the strategy player I can always make a move hence he cannot lose in a finite number of steps. If the play is infinite then the result of the play is an infinite sequence:

$$(sr^M, q_0), m_1, (s_1, q_1), \ldots, m_k, (s_k, q_k) \ldots$$

At the same time we know that the corresponding play in the game $\widehat{G}$ has been infinite and its result is:

$$(sr^M, q_0), \widehat{m}_1, (u_1, q_1), \ldots, \widehat{m}_k, (u_k, q_k) \ldots$$

Because in the game $\widehat{G}$ player I used the winning strategy $\widehat{\sigma}$ we know that the smallest integer appearing infinitely often in the sequence $\Omega(q_1), \Omega(q_2), \ldots$ is

even. But this implies that player I won in the game $G^\vee$. Hence the strategy we have defined is winning and $\mathcal{A}^\vee$ accepts M.

Function $\delta^\vee$ was defined in such a way that the automaton $\mathcal{A}^\vee$ is of the form required in Fact 8. Hence there is a μ-calculus sentence $\varphi^\vee$ equivalent to $\mathcal{A}^\vee$. □

From Corollary 10 and the fact that the sentence $\varphi^\vee$ from Lemma 12 can be constructed effectively it follows:

Corollary 13. *It is decidable whether a MSOL sentence defines a bisimulation closed set of trees.*

Remark. Analysing the proof of Theorem 11 one can observe that the theorem remains true also when we restrict to finite branching transition systems.

Remark. One may ask what is the meaning of $\widehat{\varphi}$ given in Lemma 12 if φ is not bisimulation invariant. Unfortunately the class defined by $\widehat{\varphi}$ is not so easy to describe and it does not seem to be very interesting. On the other hand we have the following fact.

Fact 14. *Bisimulation closure of a MSOL sentence is not always MSOL definable.*

Let us give an example of such a sentence. Let φ be a sentence saying that every node has exactly one successor and that on the unique path from the source there is exactly one state where a predicate p holds. The bisimulation closure of φ contains all the trees with the property that on every two paths p holds at exactly the same distance from the root. If all such trees are models of some MSOL formula then from the automata characterisation it follows that some tree which does not have this property is also a model of this formula. But this last tree is not bisimilar to a model of φ.

6 Concluding remarks

We have investigated the expressive completeness problem for branching time logics. For this we have introduced a new kind of automata capable of recognising classes of transition systems. The definition of automata has been parametrised by the set of basic formulas. This has given us a common ground to compare expressive power of MSOL and the μ-calculus. The fact that the proof of Theorem 11 is relatively easy suggest that this notion of automata may be an interesting one. .

Of course not all properties of potential interest can be expressed in MSOL. Some logics capable of expressing nonregular properties were proposed in the literature (see for example [17, 7]). We think that in this case it is also important to look for some new standards to compare expressive power with.

There is one new area of verification were the need for "yardsticks" seems to be particularly pressing. We have in mind verification with respect to so called non-interleaving semantics [32, 1, 29]. There are good reasons for considering these

semantics, as for example, some problems undecidable for transition systems semantics become decidable in this setting [12]. Nevertheless if we consider the number of different non-interleaving semantics and multiply it by the number of logics proposed to date for transition systems semantics we can see that there is a possibility of "problem explosion". That is at some point we may have a big number of incomparable approaches. We think that it would be very useful to find some expressibility standards in this area and we hope that a generalisation of automata presented here may be also a small step in this direction.

References

1. Rajeev Alur, Doron Peled, and Wojciech Penczek. Model-checking of causality properties. In *LICS '95*, pages 90–100, 1995.
2. A. Amir. Separation in nonlinear time models. *Information and Computation*, 66:177–203, 1985.
3. Andre Arnold. *Finite Transition Systems*. Masson, Prentice-Hall, 1994.
4. B. Banieqbal and H. Barringer. Temporal logic with fixed points. volume 398 of *LNCS*, pages 62–74. Springer-Verlag, 1989.
5. J. van Benthem. *Languages in Action: Categories, Lambdas and Dynamic Logic*, volume 130 of *Studies in Logic*. North-Holland, 1991.
6. J. van Benthem and J.A. Bergstra. Logic of transition systems. Report P9308, Programming Research Group, University of Amsterdam, 1993.
7. Ahmed Bouajjani, Rachid Echahed, and Peter Habermehl. On verification problem of nonregular properties of nonregular processes. In *LICS '95*, pages 123–133, 1995.
8. I. Castellani. Bisimulations and abstraction homomorphisms. *Journal of Computer and System Sciences*, 34:210–235, 1987.
9. Mads Dam. CTL* and ECTL* as a fragments of the modal μ-calculus. In *CAAP'92*, volume 581 of *LNCS*, pages 145–165, 1992.
10. E. Allen Emerson. Temporal and modal logic. In J.van Leeuven, editor, *Handbook of Theoretical Computer Science Vol.B*, pages 995–1072. Elsevier, 1990.
11. E. Allen Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. FOCS 91*, 1991.
12. Javier Esparza and Astrid Kiehn. On the model checking problem for branching time logics and basic parallel processes. In *CAV '95*, volume 939 of *LNCS*, pages 353–366, 1995.
13. A. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *7th Ann. ACM Symp. on Principles of Programming Languages*, pages 163–173, 1980.
14. D. Gabbay. Expressive functional completeness in tense logic. In *Aspects of Philosophical Logic*, pages 91–117. Reidel, 1981.
15. Haim Gaifman. On local and non-local properties. In *Herbrand Symposium, Logic Colloquium'81*, pages 105–135. North-Holland, 1982.
16. T. Hafer and W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *14th Internat. Coll. on Automata, Languages and Programming*, volume 267 of *LNCS*, pages 269–279, 1987.
17. D. Harel and D. Raz. Deciding properties of nonregular programs. *SIAM J. Comput*, 22:857–874, 1993.

18. David Janin. *Propriérés logiques du non-déterminisme et μ-calcul modal*. PhD thesis, LaBRI – Université de Bordeaux I, 1996. Available from `http://www.labri.u-bordeaux.fr/~janin`.
19. David Janin and Igor Walukiewicz. Automata for the μ-calculus and related results. In *MFCS '95*, volume 969 of *LNCS*, pages 552–562, 1995.
20. H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, 1968.
21. Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
22. Robin Milner. *Communication and Concurrency*. Prentice-Hall, Englewood Clifs, 1989.
23. D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.
24. Damian Niwiński. Fixed points vs. infinite generation. In *Proc. 3rd. IEEE LICS*, pages 402–409, 1988.
25. Gordon Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN–19, Aarhus University, 1981.
26. Amir Pnueli. The temporal logic of programs. In *18th Symposium on Foundations of Computer Science*, pages 46–57, 1977.
27. M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.
28. Colin S. Stirling. Modal and temporal logics. In S.Abramsky, D.Gabbay, and T.Maibaum, editors, *Handbook of Logic in Comuter Science*, pages 477–563. Oxford University Press, 1991.
29. P.S. Thiagarajan. A trace based extension of linear time temporal logic. In *LICS*, pages 438–447, 1994.
30. R. van Glabbeek. The linear time – branching time spectrum. In *CONCUR'90*, volume 458 of *LNCS*, pages 278–297, 1990.
31. Igor Walukiewicz. Monadic second order logic on tree-like structures. In *STACS '96*, LNCS, pages 401–414, 1996.
32. Glynn Winskel and Mogens Nielsen. *Handbook of Logic in Computer Science*, volume 4, chapter Models for Concurrency, pages 1–148. Clarendon Press – Oxford, 1995.

This article was processed using the LaTeX macro package with LLNCS style