Model checking a logic over systems with regular sets of processes

Anantha Padmanabha MS
Institute of Mathematical Sciences
Chennai, India 600113
ananthap@imsc.res.in

Ramanujam R
Institute of Mathematical Sciences
Chennai, India 600113
jam@imsc.res.in

ABSTRACT

Though systems with process creation give rise to unboundedly many processes, their names are systematically generated and typically form a regular set. When we consider modal logics to specify properties of such systems, it is natural to consider quantification over such regular sets. These are in the realm of *term modal logics*, which are usually undecidable. We consider the *monodic* variant, in which there is only one free variable in the scope of any modality, and present a model checking algorithm for this logic.

KEYWORDS

Term modal logic, model checking, regular transition systems

1 INTRODUCTION

Modal logic, and its variants including temporal logics play an important role in system verification[1, 3, 8]. van Benthem et al. [12] show the close correspondence between processes, transition systems, verification and modal logic. The modalities are generally indexed by agents in specifying many of these systems. In most of the logics, the number of agents is assumed to be fixed and thus the number of modalities is constant. But we encounter many systems where the number of agents keeps changing dynamically. For instance, if the agents are considered as clients requesting a service, then at any point of time the number of agents is unbounded even though it is finite and more importantly it keeps changing dynamically. Another example would be an operating system running many concurrent threads. The logic studied in this paper is set in this context.

We illustrate this using an example.

Example 1.1. Consider an operating system which can execute many processes at a time. A configuration of the system is given by the states of its active processes. Any active process can change the system state by making a move. Let us assume that a move by a process can create one or more new processes (threads), thus making the active set dynamic and potentially unbounded. In this setting, we can consider assertions such as: *there is at least one process active in the present configuration* or *for all possible next configurations, the processes that are currently active will remain active in the next* . . . These cannot be expressed in modal logics with fixedly many modalities.

One solution is to consider a modal logic with infinitely many modalities. For instance, $\Diamond_n \alpha$ may be read as "process n makes a transition to a state satisfying α", where n is any natural number. However, a simple exercise shows that a formula in the logic can force only boundedly many agents and hence models are systems with fixedly many agents. Also, it is not possible to state properties like "*there exists a process such that* $\cdots$" since it needs an infinite disjunction. So we need another logical device to force unboundedly many agents.

A natural such logical mechanism is quantification. It is possible to express all the properties of the kind mentioned above using first order modal logic[4]. But what we want are not arbitrary predicates and quantification over domain elements but quantification over agent names: that is, over modality indices. In general we want a term structure over modality indices. Logics with modalities indexed by terms were studied by Grove and Halpern [7], Passy and Tinchev [11], Gargov and Goranko [6], Blackburn [2], and by some others.

Perhaps what fits us best are the $term - modal\ logics(TML)$ introduced by Fitting, Thalmann and Voronkov [5] in which modalities are indexed by terms, built on a generic first order logic. In term−modal logic, we have quantification over agents, and can identify agents based on properties. For instance, we can assert: "all agents who are waiting are also sending messages" as:
$\forall x\ \Box_x(waiting) \Rightarrow \Box_x(sent\_message)$.

Note that TML is different from standard first order modal logic, since in TML quantified variables are allowed as indices for modalities which is not the case in first order modal logics. Term− modal logic offers an intuitive way to talk about unboundedly or countably many agents and gives us a device to quantify over them. Fitting $et.al$ in [5] discuss the importance of this logic and offer a complete proof theory for reasoning in the logic. However, with the full power of first order logic inside, this logic is undecidable. In fact, TML is undecidable even if we restrict ourselves to have only propositions at the atomic level ([10]). In [9], Barteld Kooi studies the expressivity of dynamic term modal logic in the epistemic setting.

Since the logic is undecidable, we focus on the *monodic* fragment: formulas in which only one free variable occurs in the scope of any modality. In [10], we have shown that this fragment is decidable; here we study the model checking problem for the logic. Note that for model checking we need a finite specification of systems with potentially infinitely many processes, and we do this by considering systems with regular sets of active processes.

2 SYNTAX AND SEMANTICS

We now define $m - PTML$, the monodic *propositional* fragment of TML.

Definition 2.1. Given a countable set of propositions P, and a countable set of variables Var, the syntax is defined as follows:

$$\phi := p \in P \mid \neg\phi \mid \phi \wedge \phi \mid \Diamond_x \alpha \mid \exists x \alpha, x \in Var$$

where $FV(\alpha) \subseteq \{x\}$.

The notion of free variable of a formula (referred to in the syntax) is standard. $FV(p) = \emptyset$, for $p \in P$. $FV(\neg\phi) = FV(\phi)$. $FV(\phi \wedge \psi) = FV(\phi) \cup FV(\psi)$. $FV(\Diamond_x \phi) = FV(\phi) \cup \{x\}$. $FV(\exists_x \phi) = FV(\phi) \setminus \{x\}$. We call a formula ϕ a *sentence* if $FV(\phi) = \emptyset$.

Similarly the modal depth of ϕ, denoted $md(\phi)$ is defined inductively: $md(p) = 0$, for $p \in P$. $md(\neg\phi) = md(\phi)$. $md(\phi \wedge \psi) = max\{md(\phi), md(\psi)\}$. $md(\Diamond_x \phi) = md(\phi) + 1$. $md(\exists_x \phi) = md(\phi)$.

As one would expect, Kripke models now need to be enriched with interpretations for variables over agent names. In addition, agent dynamics is captured by a function (γ below) that specifies, at any world w, the set of agents *alive* (or meaningful) at w. Then coherence demands that whenever $(u, d, v) \in R$, we have that $d \in \gamma(u)$.

Definition 2.2. A model is a tuple $M = (W, D, R, \gamma, I, V)$ where W is a non-empty set of worlds, D is a non-empty set of names, $R \subseteq W \times D \times W$, $\gamma : W \to 2^D$, $I : Var \to D$, $V : W \to 2^P$. Moreover, for all $u \in W$ and $d \in D$, if there exists $v \in W$ such that $(u, d, v) \in R$, then $d \in \gamma(u)$.

In [5], a *monotonicity* condition is imposed on the accessibility relation: whenever $(u, d, v) \in R$, we also have that $\gamma(u) \subseteq \gamma(v)$. This is a reasonable assumption, but since we do not need it for our technical development, we do not assume monotonicity.

We will often display the model as $M = (F, I)$ where $F = (W, D, R, \gamma, V)$ is referred to as a *frame*. Sometimes we denote (u, d, v) by $u \xrightarrow{d} v$.

Given a formula ϕ and a model $M = (W, D, R, \gamma, I, V)$, for any $w \in W$ we define $M, w \models \phi$ (ϕ is true at w) inductively as follows:

- $M, w \models p$ if $p \in V(w)$.
- $M, w \models \neg\phi$ if $M, w \not\models \phi$.
- $M, w \models \phi_1 \wedge \phi_2$ if $M, w \models \phi_1$ and $M, w \models \phi_2$.
- $M, w \models \Diamond_x \phi$ if there is some $u \in W$ such that $(w, I(x), u) \in R$ and $M, u \models \phi$.
- $M, w \models \exists x\ \phi$ if there is some $d \in \gamma(w)$ such that $M_{x \to d}, w \models \phi$ where $M_{x \to d}$ is defined by (W, D, R, γ, I', V) such that for any $y \neq x$, $I'(y) = I(y)$ and $I'(x) = d$.

The boolean connectives, $\Box_x \phi$ and $\forall x \phi$ are defined in the standard way. A formula ϕ is satisfiable if there is some model and some world w such that ϕ is true at w.

Example revisited. We show some properties in the example system 1.1 discussed earlier, using formulas of the proposed logic.

- There is at least one process active which can potentially change the system state:
  $\exists x\ \Diamond_x \top$.
- For all possible next configurations, property p holds:
  $\forall x (\Box_x p)$.
- There are at least two processes active:
  $\exists x \Box_x p \wedge \exists y\ \Diamond_y \neg p$ [1].
- There is a process such that it can change to a configuration in which none of the processes can make a move (system halts):
  $\exists x \Diamond_x \forall y \Box_y \bot$.

Note that all the properties discussed could be expressed in the monodic fragment of the logic.

3 TRANSITION SYSTEM MODELS

Systems with unboundedly many processes to be model checked are presented as transition systems with finitely many states and edges between the states labelled by regular expressions. The idea is that there is a finite alphabet set and every process name is some string over this alphabet. If a string s is in the language of some regular expression r and there is an r edge from q to q' then it means that process s can change the system state from q to q'.

Definition 3.1. Let Σ be a finite alphabet. Let $Reg(\Sigma)$ be the set of all regular expressions over Σ. For all $r \in Reg(\Sigma)$ let L_r denote the regular language generated by the expression r. If $s, t \in \Sigma^*$ then $s \cdot t$ denotes the concatenation of strings s and t, often written st.

A *regular agent transition system* is given by $T = (Q, S, \delta, q_0)$ where

- Q is a finite set of states,
- $S \subseteq_{fin} Reg(\Sigma)$ is a finite set of regular expressions,
- $\delta \subseteq (Q \times S \times Q)$ and
- $q_0 \in Q$.

Note that we could equivalently define $\delta \subseteq_{fin} (Q \times Reg(\Sigma) \times Q)$ and extracted S from δ. Note that when we have $u_1 \xrightarrow{r_1} u_2 \xrightarrow{r_2} u_3 \in R_1$ with $a \in L_{r_1}$ and $ab \in L_{r_2}$ we could think of b as a child process of parent process a. The language of regular expressions is rich enough to consider tree structures of concurrent and sequential threads with forking, as well as process threads created within loops, perhaps while waiting for an external event to occur.

Definition 3.2. Given a finite set of regular expressions S over Σ, for any two strings $s, t \in \Sigma^*$, define $s \equiv t$ if for all $r \in S$, $s \in L_r$ iff $t \in L_r$.

Note that $\equiv$ induces a partition of Σ^*. For all strings $s \in \Sigma^*$, define $[\![s]\!] = \{t \mid s \equiv t\}$. We denote this by $\equiv_T$.

3.1 Input specification and Model

The input for model checking is specified by the tuple (T, ϕ, I, U, s_0) where T is a regular agent transition system as defined above, ϕ is a monodic formula $I : FV(\phi) \to \Sigma^*$, $U : Q \to 2^{P_\phi}$ and $s_0 \in \Sigma^*$. This induces the following model on which the model checking is done.

Definition 3.3. Given an input (T, ϕ, I, V, s_0), define the induced frame $F_T = (W, D, R, \gamma, V')$ as follows:

- $D = \Sigma^*$
- $W \subseteq (Q \times \Sigma^*)$ and $R \subseteq (W \times D \times W)$ are the least sets such that the following conditions hold:
  - $(q_0, s_0) \in W$.
  - if $(q, s) \in W$ and $(q, r, q') \in \delta$ then $(q', st) \in Q$ and $\Big((q, s),\ t,\ (q', st)\Big) \in R$ where $st \in L_r$.
- for all $(q, s) \in W$, $\gamma((q, s)) = \{s\} \cup \{t \in \Sigma^* \mid$ for some q' we have $\Big((q, s), t, (q', st)\Big) \in R\}$.

[1] x and y cannot have same witness hence at least two processes are required.

- for all $(q,s) \in W$, $V((q,s) = U(q)$.

The clause defining γ needs some explanation. It specifies the set of active agents at a node to be *exactly* those that have some transitions; these agents surely need to be active but do we not need more ? For instance, what about inheriting already active agents at parent nodes ? The answer is that for *monodic* formulas, there is only one variable free at any node and hence the children "created" are all forced only by bound variables, and once we have substituted for the free variables at the root, we do not need any further inheritance. Thus the definition here suffices.

4 MODEL CHECKING PROBLEM

Given an input instance (T,ϕ,I,V,s_0), the model checking problem is to decide whether $(F_T,I),(q_0,s_0) \models \phi$.

Note that it is enough to consider a tree frame which is the unravelling of F_T upto depth $md(\phi)$ with (q_0,s_0) as the root. Hence the model checking problem is on a rooted tree with bounded depth but can have potentially infinite branching. Let the *induced tree model* be $M_{(T,\phi,I,V,s_0)} = (W,D,R,\gamma,I,V)$, the corresponding rooted tree, with W_i being the set of all nodes at level i, where $W_0 = (q_0,s_0)$.

4.1 Example revisited

Consider the example 1.1. We now illustrate the model checking problem for a system of this kind. Let the system processes be named by strings over $\Sigma = \{a,b\}$. Let the regular agent transition system T be as given in the figure 1.

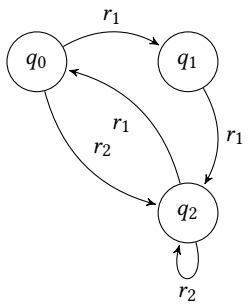

Figure 1: A regular transition system.

Suppose $r_1 = \Sigma^* a$ and $r_2 = (ab)^*$, then the induced model is given in figure 2. The set of processes active at any state is given by the union of the labels of outgoing edges from the corresponding state. For instance, $\gamma\Big((q_0,\epsilon)\Big) = \{\epsilon, a, ab\cdots\}$.

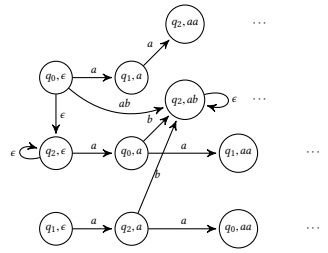

Figure 2: Induced model of Fig 1 where $r_1 = \Sigma^* a$ and $r_2 = (ab)^*$.

Now, if the model checking is to be done at the node (q_2,ϵ), then the corresponding unravelled tree is given in fig 3.

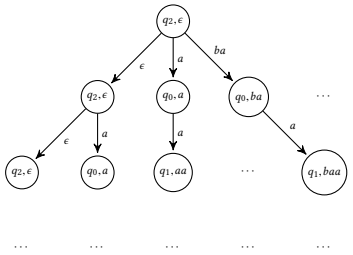

Figure 3: Model unravelled into a tree with (q_2,ϵ) as the root.

Now suppose the valuation functions are given by $V(q_0) = \{p\}$, $V(q_1) = \{p,r\}$ and $V(q_2) = \{r\}$ then $\forall x \Box_x (p \wedge r)$ is false at (q_2,ϵ) however, $\exists x \Diamond_x \Diamond_x (p \wedge r)$ is true at (q_2,ϵ).

4.2 Bisimulation

Before addressing the model checking problem, we define the notion of bisimulation.

Definition 4.1. Given two frames $F_1 = (W_1,D_1,R_1,\gamma_1,V_1)$ and $F_2 = (W_2,D_2,R_2,\gamma_2,V_2)$, we define a tuple (G,H) to be a bisimulation where G is called *world bisimulation* with $G \subseteq W_1 \times W_2$ and $H \subseteq D_1 \times D_2$ is called *name bisimulation* such that for any $(u_1,u_2) \in G$ the following conditions hold:

- $V_1(u_1) = V_2(u_2)$.
- for any $d_1 \in \gamma_1(u_1)$ there is some $d_2 \in \gamma_2(u_2)$ such that $(d_1,d_2) \in H$.
- for any $d_2 \in \gamma_2(u_2)$ there is some $d_1 \in \gamma_1(u_1)$ such that $(d_1,d_2) \in H$.
- for every $d_1 \in D_1$ and $v_1 \in W_1$ if $u_1 \xrightarrow{d_1} v_1 \in R_1$ then for every $d_2 \in D_2$ if $(d_1,d_2) \in H$ then there exists $v_2 \in W_2$ such that $(v_1,v_2) \in G$ and $u_2 \xrightarrow{d_2} v_2$.
- for every $d_2 \in D_2$ and $v_2 \in W_2$ if $u_2 \xrightarrow{d_2} v_2 \in R_2$ then for every $d_1 \in W_1$ if $(d_1,d_2) \in H$ then there exists $v_1 \in W_1$ such that $(v_1,v_2) \in G$ and $u_1 \xrightarrow{d_1} v_1$.

It is an easy exercise to show that formulas of our logic preserve bisimulation ([10]).

Definition 4.2. For any given input instance (T,ϕ,I,V,s_0), consider the induced tree model $M_{(T,\phi,I,V,s_0)} = (W,D,R,\gamma,I,V)$. For every height $i \leq md(\phi)$ and for all (q,s), $(q',s') \in W_i$ define $(q,s) \cong_i (q's')$ if $q = q'$, and $s \equiv_T s'$.

$\cong$ induces a partition of W. Define $[(q,s)] = \{(q',s') \mid (q,s) \cong (q',s)\}$ and $W' = \{[(q,s)] \mid (q,s) \in W\}$.

Define $D' = \{[s] \mid s \in \Sigma^*\}$.

For that $\cong_i$ and $\equiv_T$ induce a *bounded* partition and the size of $|D'| \leq 2^{|S|}$ and $|W'| \leq l^2 \cdot 2^{|S|} \cdot |Q|$ where Q and S are the finite set of states and finite set of regular expressions specified in T respectively and l is the length of the formula ϕ.

Definition 4.3. Given an input instance (T,ϕ,I,V,s_0), consider the induced tree model $M_{(T,\phi,I,V,s_0)} = (W,D,R,\gamma,I,V)$. Define the filtered model $M'_{(T,\phi,I,V,s_0)} = (W',D',R',\gamma',I',V')$ as follows:

- W' and D' are as defined above.
- $R' = \{\Big([(q,s)],[t],[(q',st)]\Big) \mid \Big((q,s),t,(q',st)\Big) \in R\}$.
- $\gamma'([(q,s)]) = \{[t] \mid t \in \gamma(q,s)\}$.
- For any variable x, if $I(x) = t$ then $I'(x) = [t]$.
- $V'([(q,s)] = V((q,s))$.

4.3 Main result

First we observe that given a finite model M, there are only finitely many agents used as names in the model and hence the standard labelling algorithm for model checking modal logics can be used [3]. Hence over finite models, model checking is in P for the logic. The difficulty appears for us because models are implicitly infinite, with unbounded branching (though of bounded depth). The theorem below assures us that we can reduce the problem to the standard one.

Theorem 4.4. *Given an input instance (T,ϕ,I,V,s_0), consider the induced model $M = (W,D,R,\gamma,I,V)$ and the filtered model $M' = (W',D',R',\gamma',I',V')$. Then, for all $(q,s) \in W$ we have $(q,s) \in W_i$ is bisimilar to $[(q,s)] \in W'_i$.*

Proof. Define $G = \{\Big((q,s),[(q,s)]\Big) \mid (q,s) \in W$ where (q,s) and $[(q,s)]$ are at same level $\}$ and $H = \{(t,[\![t]\!]) \mid t \in \Sigma^*\}$. We show that (G,H) is a bisimulation by verifying all the required properties.

- The first property is satisfied since by definition $V'([(q,s]) = V(q,s)$.
- Suppose $t \in \gamma(q,s)$ then by definition $[\![t]\!] \in \gamma'([q,s])$.
- For the third property, first we observe the following: suppose $t \in \gamma((q,s))$; then there is some r such that $t \in L_r$ and $(q,r,q') \in \delta$ for some q'. Hence for any $(q_1,s_1) \cong (q,s)$ we have $t \in \gamma((q_1,s_1)$ because by definition $q_1 = q$ and $s \equiv_T s_1$.

  Now suppose that $[\![t]\!] \in \gamma'([(q,s)])$; we need to prove that $t_1 \in \gamma(q,s)$ for some $t' \equiv_T t$. Since $[\![t]\!] \in \gamma'([(q,s)])$ there is some $t_2 \in [\![t]\!]$ and some $(q',s') \in [(q,s)]$ such that $t_2 \in \gamma(q',s')$ then by the observation above, for all $(q_1,s_1) \in [(q,s)]$ we have $t_2 \in \gamma((q_1,s_1)$.
- Suppose $\Big((q,s),t,(q',st)\Big) \in R$ then by definition $\Big([(q,s)],[t],[(q',st)]\Big) \in R'$.
- For the last condition, suppose $\Big([(q,s)],[\![t]\!],[(q',st)]\Big) \in R'$; let $(q,s_1) \in [(q,s)]$ we need to prove that there is some $t_1 \in [\![t]\!]$ such that $(q',s_1t_1) \cong (q,st)$.

  By definition there is some $t_2 \in [\![t]\!]$, $(q,s_2) \in [(q,s)]$ and $(q',s'_2t_2) \in [(q',s_2t_2)]$ such that $\Big((q,s_2),t_2,(q',s_2t_2)\Big)$ $\in R$. Since $s_1 \equiv s_2$ and $t_1 \equiv t$ we have $s_1t \equiv s_2t$ and thus $(q',s_1t) \cong (q',st)$. Now, by definition $\Big((q,s_1),t,(q',s_1t_1)\Big) \in R$ and hence we are done.

Hence (q,s) in M and $[(q,s)]$ in M' at the same level satisfy the same formulas, and we are done.

□

Theorem 4.5. *Given an input instance (T,ϕ,I,V,s_0), the corresponding model checking problem is in $O(l^2 \cdot 2^{|S|} \cdot |Q|)$ non – deterministic time where l is the length of ϕ and Q and S are the states and regular expressions mentioned in T respectively.*

Proof. Construct M' which is $O(l \cdot l \cdot 2^{|S|} \cdot |Q|)$ size. This is because the two l factors corresponding to height of the tree and length of the formula and the $|S| \cdot |Q|$ factor is due to the number of equivalence paritions. Now, since M' is finite, we can check if $M',[(q_0,s_0)] \models \phi$ in polynomial time (in the size of M') and since (q_0,s_0) in M is bisimilar to $[(q_0,s_0)] \in M'$, the truth of the formula is preserved. □

5 CONCLUSION

In this paper, we show that model checking for monodic *PTML* over transition systems with regular sets of agents is in *non – deterministic EXP*-time. We do not have a matching lower bound, but with the power of quantification, an exponential cost is rather to be expected. However, it remains to explore the model checking of full *PTML*.

Another interesting direction is to use dynamic logic (for regular sets of names) over modalities rather than quantification.

On the systems side, the most important line to pursue is to have a processes specified by a term algebra with process creation, and consider model checking of such processes directly. One would then naturally be led to process termination as well. However, it remains to be seen how far monodic specifications will be useful for verifying properties of such systems. Perhaps more interesting would be constraints on such systems as well as structure on formulas that yield better algorithmic properties for model checking.

6 ACKNOWLEDGEMENT

We thank the reviewers for many insightful comments which have given us interesting directions for further development of this work.‘

REFERENCES

[1] Roberta Ballarin. 2014. Modern Origins of Modal Logic. In *The Stanford Encyclopedia of Philosophy* (winter 2014 ed.), Edward N. Zalta (Ed.).
[2] Patrick Blackburn. 1993. Nominal Tense Logic. *Notre Dame Journal of Formal Logic* 34, 1 (1993), 56–83. DOI: http://dx.doi.org/10.1305/ndjfl/1093634564
[3] Patrick Blackburn, Maarten de Rijke, and Yde Venema. 2001. *Modal Logic (Cambridge Tracts in Theoretical Computer Science).* Cambridge University Press.
[4] M. Fitting and Richard L. Mendelsohn. 1999. *First-Order Modal Logic (Synthese Library).* Springer.
[5] Melvin Fitting, Lars Thalmann, and Andrei Voronkov. 2001. Term-Modal Logics. *Studia Logica* 69, 1 (2001), 133–169. DOI: http://dx.doi.org/10.1023/A:1013842612702

[6] George Gargov and Valentin Goranko. 1993. Modal logic with names. *J. Philosophical Logic* 22, 6 (1993), 607–636. DOI: http://dx.doi.org/10.1007/BF01054038

[7] Adam J. Grove and Joseph Y. Halpern. 1991. Naming and Identity in a Multi-Agent Epistemic Logic. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991.* 301–312.

[8] MJ Hughes and GE Cresswell. 1996. *A New Introduction to Modal Logic. Routledge. 1996.* Routledge.

[9] Barteld Kooi. 2007. Dynamic term-modal logic. In *A Meeting of the Minds.* 173–186.

[10] Anantha Padmanabha MS and R Ramanujam. 2016. Propositional term–modal logic: Decidability. *Logic and the Foundations of Game and Decision Theory(LOFT)* (2016). http://loft.epicenter.name/wp-content/uploads/2016/07/loft12_prog.html Technical report: http://www.imsc.res.in/~ananthap/ptml.pdf.

[11] Solomon Passy and Tinko Tinchev. 1985. Quantifiers in combinatory PDL: completeness, definability, incompleteness. In *Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985.* 512–519. DOI: http://dx.doi.org/10.1007/BFb0028835

[12] Johan van Benthem, Jan van Eijck, and Vera Stebletsova. 1994. Modal Logic, Transition Systems and Processes. *J. Log. Comput.* 4, 5 (1994), 811–855. DOI: http://dx.doi.org/10.1093/logcom/4.5.811