Math. Struct. in Comp. Science (2010), *vol.* 20, *pp.* 75–103. © Cambridge University Press 2010
doi:10.1017/S0960129509990260

Modal and mixed specifications: key decision problems and their complexities

ADAM ANTONIK[†], MICHAEL HUTH[‡§], KIM G. LARSEN[¶‖], ULRIK NYMAN[¶] and ANDRZEJ WĄSOWSKI[◇‖]

[†]*CNRS, Ecole Normale Supérieure de Cachan, France*
Email: antonik@lsv.ens-cachan.fr
[‡]*Department of Computing, Imperial College London, United Kingdom*
Email: m.huth@imperial.ac.uk
[¶]*Department of Computer Science, Aalborg University, Denmark*
Email: {kgl;ulrik}@cs.aau.dk
[◇]*IT University of Copenhagen, Denmark*
Email: wasowski@itu.dk

Received 16 October 2009

Modal and mixed transition systems are specification formalisms that allow the mixing of over- and under-approximation. We discuss three fundamental decision problems for such specifications:

— whether a set of specifications has a common implementation;
— whether an individual specification has an implementation; and
— whether all implementations of an individual specification are implementations of another one.

For each of these decision problems we investigate the worst-case computational complexity for the modal and mixed cases. We show that the first decision problem is EXPTIME-complete for both modal and mixed specifications. We prove that the second decision problem is EXPTIME-complete for mixed specifications (it is known to be trivial for modal ones). The third decision problem is also shown to be EXPTIME-complete for mixed specifications.

1. Introduction

Labelled transition systems are often used to define the semantics of modelling languages, and then to reason about models in these languages. However, it is frequently the case that a single transition system is incapable of serving multiple purposes. For example, an over-approximating transition system can be used to establish safety properties soundly, but not liveness properties. Similarly, an under-approximating transition system can be used to prove liveness properties, but not safety properties. A simple remedy for this

§ This research was partially supported by the UK EPSRC projects *Efficient Specification Pattern Library for Model Validation EP/D50595X/1* and *Complete and Efficient Checks for Branching-Time Abstractions EP/E028985/1.*
‖ The work of these authors was supported by MT-LAB, VKR Centre of Excellence.

problem is to use two transition systems in a verification process that requires us to capture both viewpoints: one describing an over-approximation, the other describing an under-approximation of the same behaviour.

However, this solution introduces a lack of precision, which is caused by decoupling the states of one abstraction from those of the other, and this means we cannot verify nested properties, which are typical for recursive logics. For example, one cannot prove that a state in which a certain liveness property holds is unreachable. To deal with this problem, model checkers such as Yasm (Gurfinkel *et al.* 2006) handle over- and under-approximation in a single structure based on a single transition system.

This idea can be traced back to the late eighties, when Larsen and Thomsen proposed modal transition systems (Larsen and Thomsen 1988), which are also known as modal specifications (Larsen 1989) and mixed specifications (Dams 1996). Modal specifications combine over- and under-approximation in a single transition system using two transition relations but a single set of states. However, inconsistencies may arise in such specifications if some behaviour is both required and disallowed. We chose to call the general, and possibly inconsistent, form of specifications *mixed specifications*, reserving the term *modal specifications* for the subset that syntactically enforces consistency (in modal specifications, the required transition relation is included in the allowed transition relation). For clarity, we use this naming convention in this paper, but note that it has not been universally adopted in the existing literature.

Mixed specifications have since been applied as suitable abstractions in, amongst other areas, program analysis (Huth *et al.* 2001; Schmidt 2001), model checking (Godefroid *et al.* 2001; Børjesson *et al.* 1993; Gurfinkel *et al.* 2006), verification (Larsen *et al.* 1995; Bruns 1997), solving process algebraic equation systems (Larsen and Xinxin 1990), compositional reasoning with interface theories (Larsen *et al.* 2007a), modelling of variability in software product lines (Larsen *et al.* 2007a; Fischbein *et al.* 2006) and other model management areas such as model merging (Uchitel and Chechik 2004; Brunet *et al.* 2006).

As an example, we will briefly consider a model originating in interface theories, which can be used to explain the motivation of our work. Figure 1 shows an interface of a communication component. This interface models communication components that retry transmission at least once after a failure, and that optionally can check the link status upon a failure (so that it can react appropriately). Specifically, the interface specifies five output actions (ok, fail, trnsmt, linkStatus, log) and five input actions (send, ack, nack, up, down), all enumerated on the rectangular frame in the figure. The interior of the frame contains an automaton specifying the desired and allowed behaviours. Transitions labelled by $\Box$ are *required* by the interface, transitions labelled by $\diamond$ are *allowed* by the interface. Thus, assuming that the state labelled 14 is the initial state, the component must first await a send request, and when it receives one, it is obliged to transmit a message, and then wait for an acknowledgement. If the acknowledgement arrives (state 19), the component successfully closes the communication and sends the requester an ok! notification, if an error message arrives (state 17), the component needs to retransmit, or, alternatively, it *may* check the status of the underlying link. After the second attempt to transmit (state 18), the component either disallows failure, or *may* retry again (the nack transition).

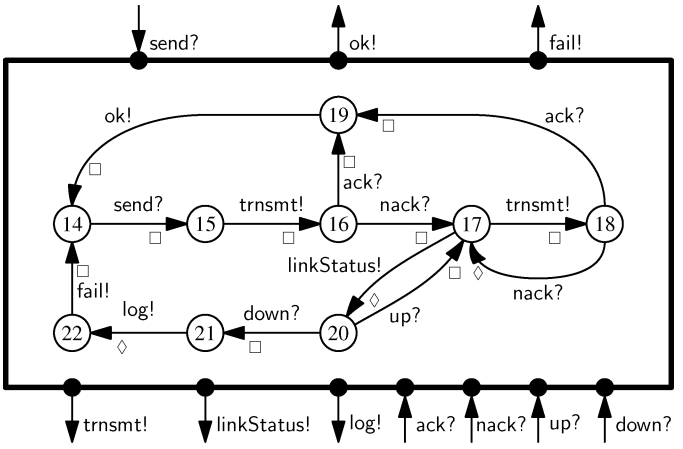


Fig. 1. An interface of a simple communication module (originally presented in Larsen *et al.* (2007a))

Larsen *et al.* (2007a) and Raclet (2008) have described interface models like this. Here we will just observe that the underlying semantic model is that of modal specifications. Thus, decision procedures for interfaces often relate to decision procedures for modal specifications. For example, if a component needs to implement several interfaces, the question arises as to whether the interfaces are consistent. A similar question is whether a certain interface is a proper generalisation of another one, that is, does every component implementing the former also implement the latter. In the present paper we discuss the computational complexity of these questions, formulating them for both 'mixed' and 'modal' specifications implicitly:

C Is an individual specification consistent, that is, can it be implemented?
CI Is a collection of specifications consistent, that is, does there exist a common implementation for them?
TR Does one specification thoroughly refine another, that is, is every implementation of the former an implementation of the latter?

Our results are obtained as follows. First we argue that all three decision problems are in EXPTIME for both modal and mixed specifications. Then we prove three reductions, which give us lower bounds:

1 We show that the EXPTIME-complete problem of acceptance of an input in a linearly bounded alternating Turing machine reduces to CI for *modal* specifications. From this we learn that CI is EXPTIME-hard for modal specifications, and thus also for mixed specifications.
2 We show that CI for modal specifications reduces to C for mixed specifications, and thus that the EXPTIME-hardness of CI give us the EXPTIME-hardness of C for mixed specifications.
3 Finally, we show that C for mixed specifications reduces to TR for mixed specifications, and thus we get the EXPTIME-hardness of TR for mixed specifications from the EXPTIME-hardness of C for mixed specifications.

This reduction chain begins with modal specifications, but has to resort to mixed, non-modal specifications for C. Therefore, we are only able to infer that CI is EXPTIME-complete for modal specifications, and are unable to determine any new lower bounds for TR for modal specifications.

Structure of the paper

In Section 2, we give the background required to appreciate the technical development of the paper. We discuss some related work in Section 3. In Sections 4, 5 and 6, we describe the three reductions that give us the EXPTIME-completeness of CI (for modal and mixed specifications), C (for mixed specifications) and TR (for mixed specifications). We put these results into context in Section 7 and present conclusions in Section 8.

2. Background

We will begin by giving formal definitions for the basic models of interest in our study (Larsen 1989; Dams 1996; Clarke *et al.* 1994).

Definition 1. Let Σ be a finite alphabet of actions.

1 A *mixed specification* M is a triple $(S, R^{\Box}, R^{\Diamond})$, where S is a finite set of states and $R^{\Box}, R^{\Diamond} \subseteq S \times \Sigma \times S$ are the must- and may-transition relations (respectively).
2 A *modal* specification is a mixed specification satisfying $R^{\Box} \subseteq R^{\Diamond}$; all of its must-transitions are also may-transitions.
3 A *pointed* mixed specification (M, s) is a mixed specification M with a designated initial state $s \in S$.
4 The size $|M|$ of a mixed specification M is defined as $|S| + |R^{\Box} \cup R^{\Diamond}|$.

Remark 1. Throughout this paper, unless stated otherwise, references to 'mixed' specifications also apply to 'modal' ones, as in the last two items of Definition 1.

Refinement (Larsen 1989; Dams 1996; Clarke *et al.* 1994), called 'modal refinement' in Larsen *et al.* (2007b), is a co-inductive relationship between two mixed specifications that verifies that one such specification is more abstract than the other. This generalises the co-inductive notion of bisimulation (Park 1981) to mixed specifications.

Definition 2. A pointed, mixed specification $(N, t_0) = ((S_N, R_N^{\Box}, R_N^{\Diamond}), t_0)$ *refines* another pointed, mixed specification $(M, s_0) = ((S_M, R_M^{\Box}, R_M^{\Diamond}), s_0)$ over the same alphabet Σ, written $(M, s_0) \prec (N, t_0)$, if and only if there is a relation $Q \subseteq S_M \times S_N$ containing (s_0, t_0) such that whenever $(s, t) \in Q$:

1 for all $(s, a, s') \in R_M^{\Box}$, there exists some $(t, a, t') \in R_N^{\Box}$ with $(s', t') \in Q$;
2 for all $(t, a, t') \in R_N^{\Diamond}$, there exists some $(s, a, s') \in R_M^{\Diamond}$ with $(s', t') \in Q$.

Deciding whether an individual finite-state, pointed, mixed specification refines another is in PTIME, and can be implemented by a standard fixpoint algorithm like those used for checking simulation or bisimilarity.

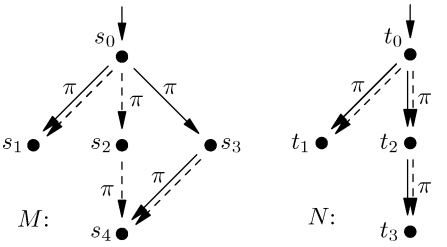


Fig. 2. Pointed, mixed $((M, s_0))$ and pointed, modal $((N, t_0))$ specifications over alphabet $\Sigma = \{\pi\}$ with $I(M, s_0) = I(N, t_0)$ but not $(N, t_0) \prec (M, s_0)$. Throughout this and later figures showing specifications, solid arrows denote must-transitions, whereas dashed arrows depict may-transitions.

Example 1. The pointed, mixed specification (M, s_0) and pointed, modal specification (N, t_0) in Figure 2 have the same set of implementations $I(M, s_0) = I(N, t_0)$ (to be defined shortly) with $(M, s_0) \prec (N, t_0)$ given by

$$Q = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_2), (s_4, t_3)\}.$$

But we do not have $(N, t_0) \prec (M, s_0)$. To see this, assume that there is a relation Q with $(t_0, s_0) \in Q$ satisfying the properties in Definition 2. Then, from $(s_0, \pi, s_2) \in R_M^{\Diamond}$, we can infer that there must be some x with $(t_0, \pi, x) \in R_N^{\Diamond}$ and $(x, s_2) \in Q$. In particular, x can only be t_1 or t_2. If x is t_1, then since $(s_2, \pi, s_4) \in R_M^{\Diamond}$ and $(t_1, s_2) \in Q$, there has to be some $R_N^{\Diamond}$ transition out of t_1, which is not the case. If x is t_2, then $(t_2, \pi, t_3) \in R_N^{\Box}$ and $(t_2, s_2) \in Q$ imply that there is some $R_M^{\Box}$ transition out of s_2, which is not the case. In conclusion, there cannot be such a Q, so $(N, t_0) \not\prec (M, s_0)$.

Labelled transition systems over an alphabet Σ are pairs (S, R) where S is a non-empty set of states and $R \subseteq S \times \Sigma \times S$ is a transition relation. We identify labelled transition systems (S, R) with modal specifications (S, R, R). The set of implementations $I(M, s)$ of a pointed, mixed specification (M, s) are all pointed labelled transition systems (T, t) refining (M, s). Note that $I(M, s)$ may be empty in general, but is guaranteed to be non-empty if M is a modal specification.

Definition 3. Let (N, t) and (M, s) be pointed, mixed specifications. As in Larsen *et al.* (2007b), we define *thorough refinement* $(M, s) \prec_{th} (N, t)$ to be the predicate $I(N, t) \subseteq I(M, s)$.

Refinement approximates this notion: $(M, s) \prec (N, t)$ implies $(M, s) \prec_{th} (N, t)$ since refinement is transitive. The converse is known to be false (Hüttel 1988; Xinxin 1992; Schmidt and Fecher 2007), contrary to the claim in Huth (2005b), with Figure 2 providing a counterexample.

We shall now formally define the decision problems informally stated above. Each decision problem has two instances: one for modal and the other for mixed specifications:

— *Common implementation* (CI): Given $k > 1$ specifications (M_i, s_i), is the intersection $\bigcap_{i=1}^{k} I(M_i, s_i)$ non-empty?
— *Consistency* (C): Is $I(M, s)$ non-empty for a specification (M, s)?

— *Thorough refinement* (TR): Does a specification (N, t) thoroughly refine a specification (M, s), that is, do we have $I(N, t) \subseteq I(M, s)$?

As far as these decision problems are concerned, the restriction to finite implementations, which follows from restricting our definitions to finite specifications, does not cause any loss of generality, as explained in Antonik *et al.* (2008b): a mixed specification (M, s) is consistent in the infinite sense if and only if its characteristic modal μ-calculus formula $\Psi_{(M,s)}$ (Huth 2005a) is satisfiable. In general, a transition system satisfying a modal μ-calculus formula may be infinite. The small model theorem for μ-calculus (Kozen 1988) tells us that $\Psi_{(M,s)}$ is satisfiable if and only if it is satisfiable over finite-state implementations. Hence, reasoning about consistency does not require reasoning about infinite structures. We can reason in a similar manner about common implementation and thorough refinement, which justifies the restriction to finite-state implementations. The restriction to finite-state specifications is needed in order to do complexity analysis.

Now we establish an EXPTIME upper bound for our key decision problems for modal and mixed specifications.

Lemma 4. The decision problems CI, C and TR for both modal and mixed specifications are in EXPTIME in the sum of their sizes.

Sketch of proof. Mixed and modal specifications (M, s) have characteristic formulae $\Psi_{(M,s)}$ (Huth 2005a) in the modal μ-calculus such that pointed labelled transition systems (L, l) are implementations of (M, s) if and only if (L, l) satisfies $\Psi_{(M,s)}$. The common implementation and consistency problems, CI and C, reduce to satisfiability checks of $\bigwedge_i \Psi_{(M_i,s_i)}$ and $\Psi_{(M,s)}$, respectively. The thorough refinement problem of whether $(M, s) \prec_{th} (N, t)$ reduces to a validity check of $\neg\Psi_{(N,t)} \vee \Psi_{(M,s)}$.

Validity checking of such vectorised modal μ-calculus formulae is in EXPTIME. One way to see this is by translating the problem into alternating tree automata. It is well known that a formula $\Psi_{(M,s)}$ can be efficiently translated (Wilke 2001) into an alternating tree automaton $\mathsf{A}_{(M,s)}$ (with the parity acceptance condition) that accepts exactly those pointed labelled transition systems that satisfy $\Psi_{(M,s)}$. Since non-emptiness, intersection and complementation of languages is in EXPTIME for alternating tree automata, we get our EXPTIME upper bounds if these automata have size polynomial in $|M|$.

Since the size of $\Psi_{(M,s)}$ may be exponential in $|M|$, we require a direct translation from (M, s) into a version of $\mathsf{A}_{(M,s)}$. The formulae $\Psi_{(M,s)}$ can be written as a system of recursive equations (Larsen 1989) $X_s = body_s$ for each state s of M. We can therefore construct all $\mathsf{A}_{(M,s)}$ in a compositional manner: whenever X_s refers in its $body_s$ to some X_t, we ensure $\mathsf{A}_{(M,s)}$ has a transition to the initial state of $\mathsf{A}_{(M,t)}$ at that point. This $\mathsf{A}_{(M,s)}$ generates the same language as the one constructed from $\Psi_{(M,s)}$, by appeal to the existence of memoryless winning strategies in parity games (Zielonka 1998). The system of equations is polynomial in $|M|$, so the compositional version of $\mathsf{A}_{(M,s)}$ is polynomial in the size of that system of equations. □

For full details, see Wilke (2001) and Larsen (1989).

Remark 2. Throughout this paper we work with Karp reductions, that is, many–one reductions computable by deterministic Turing machines in polynomial time. This choice is justified since we reduce problems that are EXPTIME-complete or PSPACE-hard.

3. Related work

In this section we briefly discuss some research directly relevant to this paper.

The workshop paper Antonik *et al.* (2008c) contains a sketch of the reduction of $\mathrm{ATM}_{\mathrm{LB}}$, that is, the acceptance of input for a linearly bounded alternating Turing machine, to CI for modal specifications. This reduction was discovered, independently, by Antonik and Nyman in their Ph.D. work (Antonik 2008; Nyman 2008). This reduction constitutes an improvement over the reduction to CI for modal specifications from the PSPACE-complete problem of Generalised Geography, which appeared in Antonik *et al.* (2008b).

The conference paper Antonik *et al.* (2008b) also contains the reductions of C for mixed to CI for modal specifications, and of TR for mixed to C for mixed specifications – but the stronger reduction to alternating Turing machines makes these reductions stronger by transitivity.

Antonik *et al.* (2008b) also shows that TR for modal specifications is PSPACE-hard. This result is completely orthogonal to the techniques and results reported in the current paper.

We refer the interested reader to the invited concurrency column Antonik *et al.* (2008a), which provides more motivation and potential applications of the decision problems studied in the current paper.

The prime numbers construction in the example of Section 4 was originally proposed by Antonik, and published in Antonik (2008). Only after the fact did we learn that the same technique had also been used by Berwanger and colleagues in two other papers that were published around the same time (Berwanger *et al.* 2008; Berwanger and Doyen 2008). In these papers, the technique of multiplication of small prime numbers was used to:

(i) show that imperfect information games require exponential strategies; and
(ii) reduce imperfect information parity games to imperfect information safety games.

4. Common implementation

We begin by developing an intuition as to why the CI problem is hard before going on to give a formal proof. We will do this by constructing a set of specifications whose size is exponentially smaller than its smallest common implementation. The succinctness of specifications as a representation does not in itself prove the hardness of the problem, but, we think, it does make it quite evident that the problem is hard.

Example 2. The construction used below originated in Antonik (2008). Let I be a finite set of natural indices and, for $i \in I$, let M_i be modal specifications consisting of:

— states s_i^j, $j = 1 \ldots i$, such that $(s_i^j, \pi, s_i^{j+1}) \in R^{\Box}, R^{\Diamond}$ for $1 \leqslant j \leqslant i-1$ and $(s_i^i, \pi, s_i^1) \in R^{\Diamond}$;
— an extra deadlock state d such that $(s_i^j, a_l, d) \in R^{\Diamond}$ if $l \in I - \{i\}$, while $(s_i^i, a_i, d) \in R^{\Diamond}, R^{\Box}$.

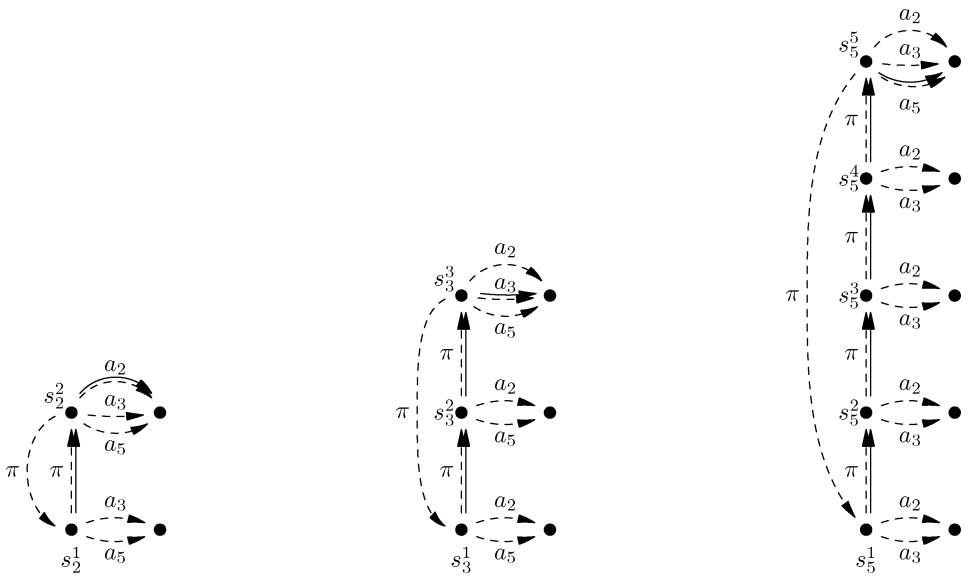


Fig. 3. Pointed specifications (M_2, s_2^1), (M_3, s_3^1) and (M_5, s_5^1) whose common implementation has at least $2 \cdot 3 \cdot 5 = 30$ states.

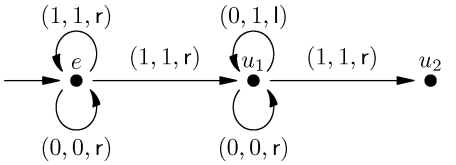
Figure 3 shows an example of the specifications M_2, M_3 and M_5 for $I = \{2, 3, 5\}$. Observe that each M_i is a counter that counts $i-1$ transitions labelled by π, allowing the implementation to stop after $i-1$ π-steps (or any multiple thereof). In any state, M_i is allowed to make an a_j transition to a deadlocking state, but only in its topmost state (see Figure 3) is it both allowed and required to be able to make an a_i transition to this state.

It is not hard to see that if we take a collection of M_i models for $i = p_1, \ldots, p_n$, ranging over the first n primes, then any implementation has at least $\prod_{i=1}^{n} p_i > \prod_{i=1}^{n} 2 = 2^n$ states. Thus the size of any common implementation of the family of models for the first n primes is exponential in n. However, we still need to show that the total size of the specifications themselves remains polynomial in n.

By a theorem of Chebyshev (Chebyshev, 1852), there exists a constant $\theta > 0$ such that the number of primes less than a given k is at least $\theta k / \log k$. Since for sufficiently large k we have $\log k < k^{1/2}$, the number of primes is greater than $\theta k^{1/2}$. In order to ensure at least n primes in the range $[0, x]$, it suffices to take x larger than $(\frac{n}{\theta})^2$. The total size of M_i specifications corresponding to these numbers is $O(n(\frac{n}{\theta})^2) = O(n^3)$. Thus the set of specifications has size polynomial in n, while its common implementations are at least exponential in n. Note that it is easy to adapt this construction so that it only uses a binary alphabet.

In the remaining part of this section, we present a formal reduction demonstrating the EXPTIME-hardness of CI. We begin with a definition of the decision problem used in the lower bound proof for common implementation.

An *Alternating Turing Machine* (Chandra *et al.* 1981), or an ATM, is a tuple $T = (Q, \Gamma, \delta, q_0, \mathsf{mode})$, where Q is a non-empty finite set of control states, Γ is an alphabet of



$$\begin{aligned}
\delta(e,\ 0) &= \{(e,\ 0, \mathsf{r})\}\\
\delta(e,\ 1) &= \{(e,\ 1, \mathsf{r}), (u_1, 1, \mathsf{r})\}\\
\delta(u_1, 0) &= \{(u_1, 1, \mathsf{l}), (u_1, 0, \mathsf{r})\}\\
\delta(u_1, 1) &= \{(u_2, 1, \mathsf{r})\}\\
\delta(u_2, 0) &= \delta(u_2, 1) = \{\}
\end{aligned}$$

Fig. 4. The transition relation of an ATM as a labelled graph and as a function.

tape symbols, $\mathsf{null} \notin \Gamma$ is a special symbol denoting empty cell contents,

$$\delta : Q \times (\Gamma \cup \{\mathsf{null}\}) \to \mathscr{P}(Q \times \Gamma \times \{\mathsf{l}, \mathsf{r}\})$$

is a transition relation, $q_0 \in Q$ is the initial control state and $\mathsf{mode} : Q \to \{\mathsf{Univ}, \mathsf{Exst}\}$ is a labelling of control states as universal or existential, respectively. Universal and existential states with no successors are called accepting and rejecting states (respectively). Each ATM T has an infinite tape of cells with a leftmost cell. Each cell can store one symbol from Γ. A head points to one cell at a time, which can then be read or written to. The head can then move to the left or right: $(q', a', \mathsf{r}) \in \delta(q, a)$, for example, says 'if the head cell (say c) reads a at control state q, then a successor state can be q', in which case cell c now contains a' and the head is moved to the cell on the right of c'. The state of the tape is an infinite word over $\Gamma \cup \{\mathsf{null}\}$.

We will now introduce a simple example, which we will use for illustrative purposes as a running example throughout this paper.

Example 3. Figure 4 presents an example of an ATM T over a binary alphabet $\Gamma = \{0, 1\}$ where arrows $q \xrightarrow{(a,a',d)} q'$ denote $(q', a', d) \in \delta(q, a)$. The initial control state e is an existential one, and both of the u_i control states are universal.

Definition 5.

1 Configurations of an ATM T are triples $\langle q, i, \tau \rangle$ where $q \in Q$ is the current control state, the head is on the ith cell from the left and $\tau \in (\Gamma \cup \mathsf{null})^\omega$ is the current tape state.
2 For input $w \in \Gamma^*$, the initial configuration is $\langle q_0, 1, w\mathsf{null}^\omega \rangle$.
3 The recursive and parallel execution of all applicable[†] transitions δ from initial configuration $\langle q_0, 1, w\mathsf{null}^\omega \rangle$ yields a computation tree $\mathsf{T}_{\langle T, w \rangle}$. We say that ATM T accepts input w if and only if the tree $\mathsf{T}_{\langle T, w \rangle}$ accepts, where the latter is defined recursively:
   — Subtree $\mathsf{T}_{\langle T, w \rangle}$ with root $\langle q, i, \tau \rangle$ and $\mathsf{mode}(q) = \mathsf{Exst}$ accepts if and only if there is a successor $\langle q', i', \tau' \rangle$ of $\langle q, i, \tau \rangle$ in $\mathsf{T}_{\langle T, w \rangle}$ such that the sub-tree with root $\langle q', i', \tau' \rangle$ accepts.

[†] Transitions $(\_, \_, \_, \mathsf{l})$ are not applicable in configurations $\langle \_, 1, \_ \rangle$ as the head cannot move over the left boundary of the tape, where we use $\_$ as a wildcard.

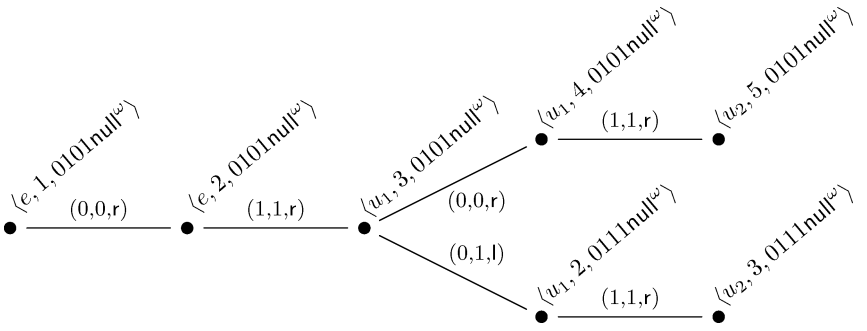


Fig. 5. An accepting computation tree $\mathsf{T}_{\langle T,0101\mathsf{null}^\omega\rangle}$ for the ATM T of Example 4.

— Subtree $\mathsf{T}_{\langle T,w\rangle}$ with root $\langle q,i,\tau\rangle$ and $\mathsf{mode}(q) = \mathsf{Univ}$ accepts if and only if for all successors $\langle q',i',\tau'\rangle$ of $\langle q,i,\tau\rangle$ in $\mathsf{T}_{\langle T,w\rangle}$, the sub-tree with root $\langle q',i',\tau'\rangle$ accepts (in particular, this is the case if there are no such successors).

Example 4. The ATM of Figure 4 accepts the regular language $(0+1)^*10^*1(0+1)^*$. Observe that u_2 is the only accepting state. Intuitively, the part of T rooted in e accepts the prefix $(0+1)^*1$: the semantics of existential states is locally that of states in non-deterministic Turing machines. The part of T rooted in u_1 consumes a series of 0 symbols until 1 is reached, which leads to acceptance. The suffix of the input word after the final 1 is ignored. Note that the computation forks in u_1 whenever a 0 is encountered. However, the top branch would reach the earlier 1 eventually and accept. Figure 5 shows one possible accepting tree for this ATM and the word $0101\mathsf{null}^\omega$.

An ATM T is *linearly bounded* if and only if for all words $w \in \Gamma^*$ accepted by T, the accepting part of the computation tree $\mathsf{T}_{\langle T,w\rangle}$ only contains configurations $\langle q,i,v\mathsf{null}^\omega\rangle$, where the length of $v \in \Gamma^*$ is no greater than the length of w. That is to say, by choosing exactly one accepting successor for each existential configuration in $\mathsf{T}_{\langle T,w\rangle}$, and removing all the remaining successors and configurations unreachable from the root, one can create a smaller tree that only contains configurations with $\langle q,i,v\mathsf{null}^\omega\rangle$ where $|v| \leqslant |w|$. We refer to such pruned computation trees simply as 'computations'.

Our notion of 'linear boundedness' follows Landweber (1963) and Laroussinie and Sproston (2007) in limiting the tape size to the size of the input. This limitation does not change the hardness of the acceptance problem (see below). In addition, we assume that linearly bounded ATMs have no infinite computations since any linearly bounded ATM can be transformed into another linearly bounded ATM, which accepts the same language, but also counts the number of computation steps used, rejecting any computation whose number of steps exceeds the number of possible configurations. This is possible because ASPACE = EXPTIME (Sipser 1996, Theorem 10.18).

Fact 1. Consider the formal language

$$\mathrm{ATM}_{\mathrm{LB}} = \{\langle T,w\rangle \mid w \in \Gamma^* \text{ is accepted by linearly bounded ATM } T\}.$$

The problem of deciding whether for an arbitrary linearly bounded ATM T and an input w, the pair $\langle T, w\rangle$ is in $\mathrm{ATM}_{\mathrm{LB}}$ is EXPTIME-complete (Chandra *et al.* 1981).

We are now in a position to prove our first EXPTIME-hardness result, which is for the decision problem of common implementations of *modal* specifications.

Theorem 6. Let $\{(M_l, s_l)\}_{l\in\{1\ldots k\}}$ be a finite family of modal specifications over the same action alphabet Σ. Deciding whether there exists an implementation (I, i) such that $(M_l, s_l) \prec (I, i)$ for all $l = 1 \ldots k$ is EXPTIME-hard.

We prove Theorem 6 by demonstrating a PTIME reduction from $\mathrm{ATM}_{\mathrm{LB}}$. Given an ATM T and an input word w of length n, we synthesise a collection of (pointed) modal specifications

$$\mathscr{M}_w^T = \{M_i \mid 1 \leqslant i \leqslant n\} \cup \{M_{\mathrm{head}}, M_{\mathrm{ctrl}}, M_{\mathrm{exist}}\}, \tag{1}$$

whose sum of sizes is polynomial in n and in the size of T, such that T accepts w if and only if there exists a (pointed) implementation I refining all members of $\mathscr{M}_w^T$.

The specifications M_i, M_{head}, M_{ctrl} and M_{exist} model the tape cell i, the current head position, the finite control of T and acceptance, respectively. Common implementations of these specifications model action synchronisation to agree on the symbol being read from the tape, the head position, the symbol written to the tape, the direction the head moves in, the transitions taken by the finite control and whether a computation is accepting. The result is that any common implementations of these specifications correspond to an accepting computation of T on input w. More precisely, any common implementation will correspond to different unfoldings of the structure of the finite control into a computation tree based on the content of the tape cells and the tape head position.

We now describe the specifications in $\mathscr{M}_w^T$ both formally and through our running example in Figure 4. All specifications in $\mathscr{M}_w^T$ have the same alphabet. Actions are of the form (a_1, i, a_2, d) and denote the fact that the machine's head is over the ith cell of the tape, which contains the a_1 symbol, and that it shall be moved one cell in the direction d after writing a_2 in the current cell. In addition, two special actions, $\exists$ and π, are used to encode logical constraints like disjunction and conjunction. The alphabet for our running example is

$$\{\pi, \exists\} \cup (\{0, 1\} \times \{1..n\} \times \{0, 1\} \times \{\mathsf{l}, \mathsf{r}\})$$

Note that a stricter and more complex reduction to CI of modal specifications over a *binary* alphabet is possible by encoding actions in binary form.

Encoding tape cells. For each tape cell i, the specification M_i represents the possible contents of cell i. It has $|\Gamma|$ states $\{p_{\langle i,a\rangle}\}_{a\in\Gamma}$ and initial state $p_{\langle i,w_i\rangle}$, representing the initial contents of the ith cell. There are no must-transitions:

$$R^{\Box} = \varnothing.$$

The may-transition relation connects any two states:

$$\text{for all symbols } a_1, a_2 \text{ in } \Gamma \text{ we have } (p_{\langle i,a_1\rangle}, (a_1, i, a_2, \_), p_{\langle i,a_2\rangle}) \in R^{\Diamond}.$$

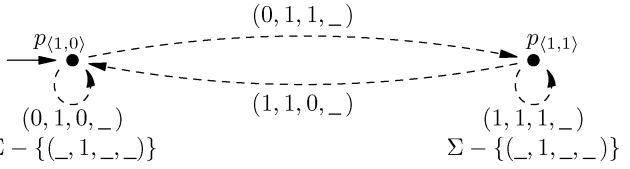


Fig. 6. The specification M_1 of the first tape cell in our running example, assuming $w_1 = 0$. In this and later figures we represent multiple transitions having the same source and target as single arrows labelled with sets of actions. Several labels placed alongside the same arrow denote a union of sets. Wildcards (the '_' symbol) are used to generate sets of actions that match the pattern in the usual sense.

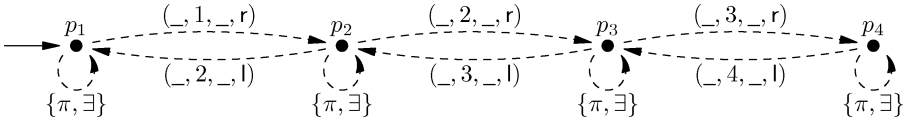


Fig. 7. Example of the head specification M_{head} assuming $|w| = 4$.

Changes in cells other than i are also consistent with M_i:

$$\text{for all } a \in \Gamma \text{ if } i \neq j \text{ with } 1 \leqslant j \leqslant n, \text{ then } (p_{\langle i,a\rangle}, (\_, j, \_, \_), p_{\langle i,a\rangle}) \in R^{\Diamond}.$$

Finally, the π and $\exists$ actions may be used freely as they do not affect the contents of the cell:

$$(p_{\langle i,a\rangle}, \pi, p_{\langle i,a\rangle}) \in R^{\Diamond} \text{ and } (p_{\langle i,a\rangle}, \exists, p_{\langle i,a\rangle}) \in R^{\Diamond} \text{ for any } a \in \Gamma.$$

There are no other may-transitions in M_i.

Figure 6 presents a specification M_1 for the leftmost cell of an ATM over a binary alphabet.

Encoding the head. The specification M_{head}, which tracks the current head position, has n states labelled p_1 to p_n, one for each possible position. Initially, the head occupies the leftmost cell, so p_1 is the initial state of M_{head}. There are no must-transitions:

$$R^{\Box} = \varnothing.$$

The may-transitions are consistent with any position changes based on the direction encoded in observed actions. More precisely,

$$\text{for every position } 1 \leqslant i < n \text{ we have } (p_i, (\_, i, \_, \mathrm{r}), p_{i+1}) \in R^{\Diamond}$$
$$\text{for every position } 1 < i \leqslant n \text{ we have } (p_i, (\_, i, \_, \mathrm{l}), p_{i-1}) \in R^{\Diamond}.$$

The π and $\exists$ transitions may again be taken freely, but in this case without moving the machine's head:

$$(p_i, \pi, p_i) \in R^{\Diamond} \text{ and } (p_i, \exists, p_i) \in R^{\Diamond} \text{ for each position } 1 \leqslant i \leqslant n.$$

There are no other may-transitions in M_{head}. Note that the head of T is only allowed to move between the first and nth cell in any computation. Figure 7 shows the specification M_{head} for our running example.

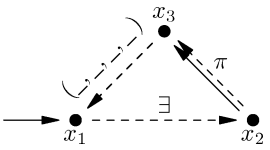


Fig. 8. The specification M_{exist}, which enforces a π-transition after each $\exists$-transition.

Encoding the finite control. The specifications M_{ctrl} and M_{exist} model the finite control of the ATM T. The specification M_{exist} is independent of the ATM T, and is defined in Figure 8. It ensures that a π-transition is taken after every $\exists$-transition. The specification M_{ctrl} mimics the finite control of T almost directly. Each control state $q_s \in Q$ is identified with a state in M_{ctrl} of the same name. Additional internal states of M_{ctrl} encode existential and universal branching:

for each q_s a state $q_{s\exists}$ with two $\exists$-transitions $(q_s, \exists, q_{s\exists}) \in R^{\Diamond} \cap R^{\Box}$ is added.

Depending on $\mathsf{mode}(q_s)$, additional states and transitions are created:

— If $\mathsf{mode}(q_s) = \mathsf{Exst}$, then for each $1 \leqslant i \leqslant n$, $a_{\text{old}} \in \Gamma$, and for each transition $(q_t, a_{\text{new}}, d) \in \delta(q_s, a_{\text{old}})$, we add a may π-transition from $q_{s\exists}$ to a new intermediate state uniquely named $\langle q_s a_{\text{old}} i a_{\text{new}} d q_t \rangle$. We then add a must-transition labelled $(a_{\text{old}}, i, a_{\text{new}}, d)$ from that intermediate state to q_t. Formally,

$$(q_{s\exists}, \pi, \langle q_s a_{\text{old}} i a_{\text{new}} d q_t \rangle) \in R^{\Diamond}$$
$$(\langle q_s a_{\text{old}} i a_{\text{new}} d q_t \rangle, (a_{\text{old}}, i, a_{\text{new}}, d), q_t) \in R^{\Diamond} \cap R^{\Box}.$$

Figure 9 shows this encoding for the state e of our running example.

— If $\mathsf{mode}(q_s) = \mathsf{Univ}$, then for each $1 \leqslant i \leqslant n$, $a_{\text{old}} \in \Gamma$, and for each transition $(q_t, a_{\text{new}}, d) \in \delta(q_s, a_{\text{old}})$, we add a may π-transition from $q_{s\exists}$ to an intermediate state named $\langle q_s a_{\text{old}} i \rangle$. We then add a must-transition labelled $(a_{\text{old}}, i, a_{\text{new}}, d)$ from the intermediate state $\langle q_s a_{\text{old}} i \rangle$ to q_t. Formally,

$$(q_{s\exists}, \pi, \langle q_s a_{\text{old}} i \rangle) \in R^{\Diamond}$$
$$(\langle q_s a_{\text{old}} i \rangle, (a_{\text{old}}, i, a_{\text{new}}, d), q_t) \in R^{\Diamond} \cap R^{\Box}.$$

The initial state of M_{ctrl} is its state named q_0, where q_0 is the initial state of T. Figure 10 demonstrates the encoding of the state u_1 of the ATM in Figure 4. The complete M_{ctrl} specification for our running example is shown in Figure 11.

Notice how the two specifications M_{ctrl} and M_{exist} cooperate to enforce the nature of alternation. For example, for an existential state, M_{ctrl} forces every implementation to have an $\exists$-transition, which may be followed by a π-transition. Simultaneously, M_{exist} allows an $\exists$-transition but subsequently requires a π-transition. Effectively, at least one of the π branches from M_{ctrl} must be implemented (which is an encoding of a disjunction).

This concludes the description of all specifications from set $\mathscr{M}_w^T$ in (1). All these specifications are modal by construction. Since the sum of their sizes is bounded by a polynomial in n and in the size of T, the remainder of the proof for Theorem 6 follows from the following lemma.

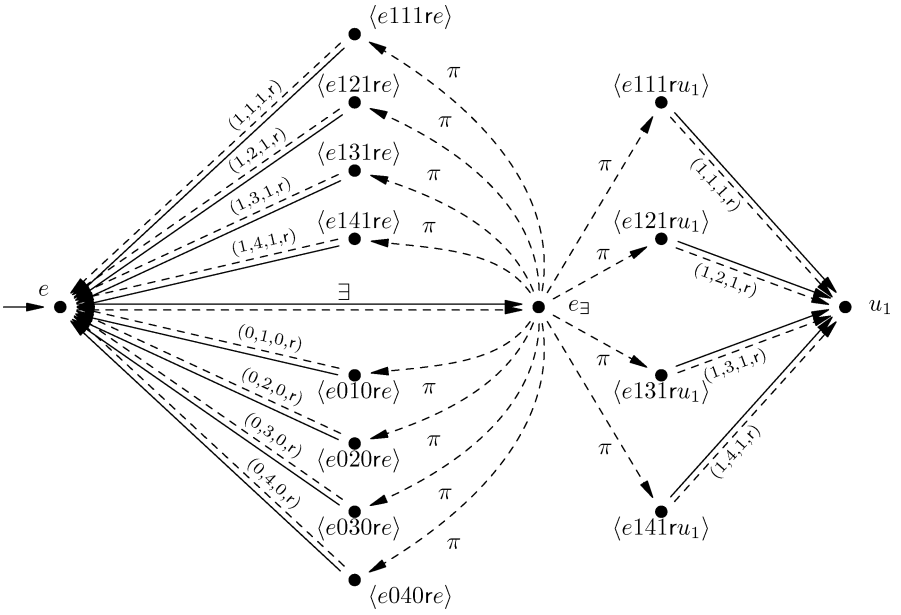


Fig. 9. Encoding for the existential state of the running example, assuming $|w| = 4$.

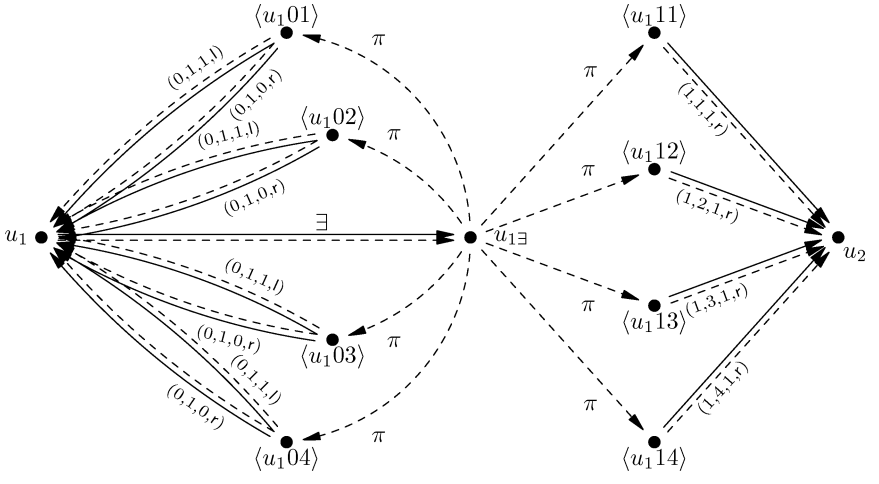


Fig. 10. Encoding for the universal state u_1 of the running example, assuming $|w| = 4$.

Lemma 7. For each linearly bounded ATM T and an input w, T accepts w if and only if the set of modal specifications $\mathscr{M}_w^T$ has a common implementation.

The proof of this lemma can be found in Appendix A. We will just mention here some points of interest. From an accepting computation tree $\mathsf{T}_{\langle T,w\rangle}$, one can construct a specification N by structural induction on $\mathsf{T}_{\langle T,w\rangle}$. This N effectively adds to $\mathsf{T}_{\langle T,w\rangle}$ some new states and labelled transitions so that the computation encoded in $\mathsf{T}_{\langle T,w\rangle}$ then interlocks with the action synchronisation of specifications in $\mathscr{M}_w^T$. Since N is of the form

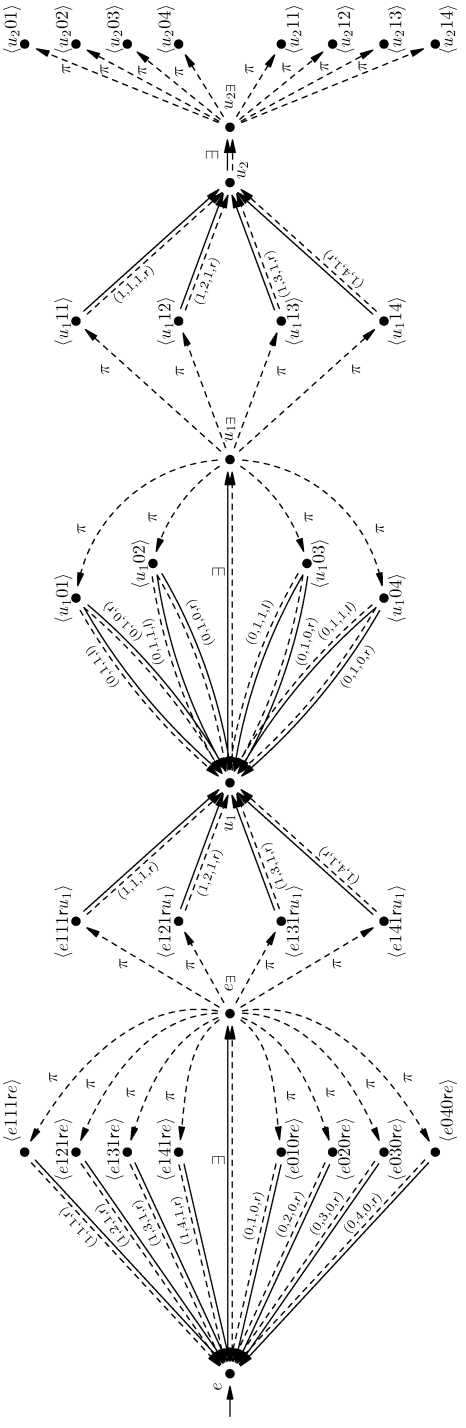


Fig. 11. The entire M_{ctrl} specification for the example of Figure 4, assuming $|w| = 4$.

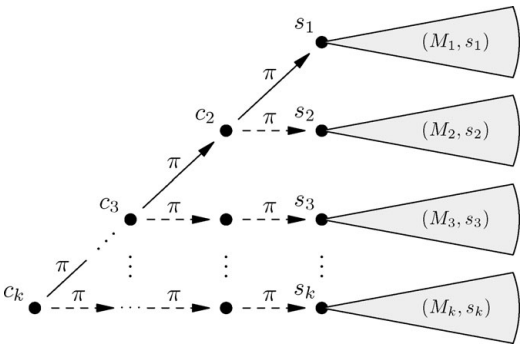


Fig. 12. Conjunction of k mixed specifications into one mixed specification

(S, R, R), it suffices to show that N is a common refinement of all members in $\mathscr{M}_w^T$. This is a lengthy but routine argument.

For the converse, a common implementation of $\mathscr{M}_w^T$ is cycle-free by our assumption that T never repeats a configuration. So the pointed common implementation is a DAG and we can use structural induction on that DAG to synthesise an accepting computation tree of T for input w. This makes use of the fact that the head of T never reaches a cell that was not initialised by input w.

We can now deduce EXPTIME-completeness for the decision problem CI for both modal and mixed specifications.

Corollary 8. The decision problem CI is EXPTIME-complete in the sum of their sizes for both modal and mixed specifications.

Proof. Theorem 6 states EXPTIME-hardness of CI for *modal* specifications. Since modal specifications are also mixed specifications, this also gives the EXPTIME-hardness of CI for mixed specifications. From Lemma 4, we know that both instances of CI are in EXPTIME. □

5. Consistency for mixed specifications

The decision problem C is of course trivial for modal specifications since all such specifications have implementations by construction. Given a pointed, modal specification $((S, R^{\Box}, R^{\Diamond}), s_0)$, one such implementation is $(S, R^{\Box}, R^{\Box}, s_0)$. In contrast, we will now show that deciding the consistency of a single mixed specification is EXPTIME-hard in its size. We achieve this using Theorem 6 and by reducing CI for several modal specifications to the decision problem C for a single mixed specification.

Theorem 9. Consistency of a mixed specification is EXPTIME-hard in its size.

Proof. By Theorem 6, it suffices to show how $k > 1$ mixed specifications (M_i, s_i) can be conjoined into one mixed specification (M, c_k) with $|M|$ being polynomial in $\sum_i |M_i|$ such that (M, c_k) has an implementation if and only if all (M_i, s_i) have a common implementation.

Figure 12 illustrates the construction, which originated in Larsen *et al.* (2007b), by showing a conjunction of states s_1, s_2, s_3 up to s_k. In order to conjoin two states s_1 and

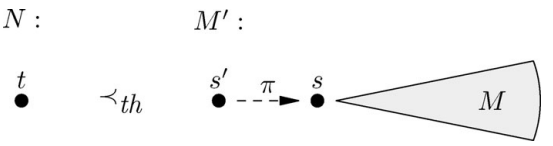


Fig. 13. Reduction of C for mixed specification (M,s) to TR for mixed specifications (N,t) and (M',s'): mixed specification (M,s) is consistent if and only if not $(N,t)\prec_{th}(M',s')$.

s_2, two new π-transitions are added from a fresh state c_2 to each of s_1 and s_2. One of the π-transitions is an $R^{\Diamond}\setminus R^{\Box}$ π-transition and the other is an $R^{\Box}$ π-transition. Only two states can be conjoined directly in this way, but the process can be iterated as many times as needed, as shown in Figure 12, by adding a corresponding number of π-transitions to the newly conjoined systems. Observe that the resulting specification is properly mixed (not modal) since it contains π-transitions that are in $R^{\Diamond}\setminus R^{\Box}$. Its size is linear in $\sum_i |M_i|$ and quadratic in k, which itself is $O(\sum_i |M_i|)$.

If the specifications that are being conjoined have a common implementation, the new specification will also have an implementation, which is the same implementation prefixed with a sequence of $k-1$ π-transitions. Conversely, if the new mixed specification has an implementation, this implementation will contain at least a sequence of $k-1$ π-transitions, followed by an implementation that must individually satisfy all the systems that have been conjoined. □

6. Thorough refinement for mixed specifications

We show EXPTIME-hardness of the decision problem TR for mixed specifications using Theorem 9 and a reduction of consistency checks to thorough refinement checks.

Theorem 10. Thorough refinement of mixed specifications is EXPTIME-hard in the size of these specifications.

Proof. By Theorem 9, deciding C for a mixed specification is EXPTIME-hard. Therefore it suffices to reduce C for mixed specifications to TR for mixed specifications. Let (M,s) be a pointed, mixed specification over Σ. Consider a pointed, modal specification (N,t) over $\Sigma\cup\{\pi\}$ with $N=(\{t\},\{\},\{\})$, which has only one state and no transitions. From (M,s), we construct the mixed specification (M',s') over $\Sigma\cup\{\pi\}$ by prefixing s with a new state s' and a single transition $(s',\pi,s)\in R^{\Diamond}_{M'}\setminus R^{\Box}_{M'}$. This construction is shown in Figure 13.

We show that (M,s) is consistent if and only if not $(N,t)\prec_{th}(M',s')$. (It is easy to see, but irrelevant to this proof, that the converse $(M',s')\prec_{th}(N,t)$ always holds.)

1 If (M,s) is consistent, it has an implementation (L,l), from which we get an implementation (L',l') of (M',s') by creating a new state l' with a transition (l',π,l). But (M',s') then has an implementation that is not allowed by (N,t), so $I(M',s')\not\subseteq I(N,t)$.
2 Conversely, if $I(M',s')\not\subseteq I(N,t)$ there exists an implementation (L,l') of (M',s'), which is not an implementation of (N,t), so (L,l') has a transition (l',π,l). Moreover, (L,l) refines (M,s) since (L,l') refines (M',s') and s is the unique successor of s' in M'. Thus (M,s) is consistent. □

Table 1. *Tabular summary of the results provided in this paper.*

	Modal specifications	Mixed specifications
Common impl.	**EXPTIME-complete**	**EXPTIME-complete**
Consistency	**trivial**	**EXPTIME-complete**
Thorough ref.	**EXPTIME**	**EXPTIME-complete**

Remark 3. Observe that part 1 of this proof works for refinement as well as thorough refinement. However, we would not be able to get the second implication for refinement in part 2 of the proof since thorough refinement does not generally imply refinement.

Also note that not only have we just shown EXPTIME-completeness for deciding whether a *mixed* specification thoroughly refines another *mixed* specification, but also for deciding whether a *mixed* specification thoroughly refines a *modal* specification.

7. Discussion

We begin by summarising the complexity results obtained in this paper:

Corollary 11. The worst-case computational complexities shown in Table 1 are correct.

There is one complexity gap in Table 1, that for TR for *modal* specifications. We have studied this fairly extensively without being able to settle the exact complexity of this decision problem. However, we learned recently that this problem has been determined to be EXPTIME-complete also (Beneš *et al.* 2009). It would be interesting to see whether the proof of this result can shed any light on the complexity of the validity problem for formulae given in the vectorised form of Larsen (1989), since the latter is one way in which one can re-express TR for both modal and mixed specifications.

Interestingly, we can reduce thorough refinement to a universal version of generalised model checking (Bruns and Godefroid 2000). In their paper, Bruns and Godefroid consider judgments $\mathsf{GMC}(M, s, \varphi)$ that are true if and only if there exists an implementation of (M, s) satisfying φ. They observe that this generalises both model checking (when (M, s) is an implementation) and satisfiability checking (when (M, s) is such that all labelled transition systems refine it). This existential judgment has a universal dual (see, for example, Antonik and Huth (2009)), $\mathsf{VAL}(M, s, \varphi)$, which is true if and only if all implementations of (M, s) satisfy φ, thus generalising both model checking and validity checking. The former judgment is useful for finding counter-examples; the latter for verification. For example, both of these uses can be seen in the CEGAR technique for program verification of Godefroid and Huth (2005). Since $(M, s) \prec_{th} (N, t)$ reduces directly to $\mathsf{VAL}(N, t, \Psi_{(M,s)})$, it would be interesting to understand the exact complexity of $\mathsf{VAL}(N, t, \varphi)$ for modal specifications (N, t) when φ ranges over characteristic formulae $\Psi_{(M,s)}$ in vectorised form.

8. Conclusion

In this paper we have revisited modal and mixed specifications. Such specifications consist of state spaces with two transition relations that can serve, respectively, as over- and under-approximations of transition relations in labelled transition systems. We then discussed three fundamental decision problems for modal and mixed specifications:

— Common implementation: do finitely many specifications have a common implementation?
— Consistency: does a specification have an implementation?
— Thorough refinement: are all implementations of one specification also implementations of another specification?

We investigated the worst-case computational complexity for these three decision problems for both modal and mixed specifications. In the case of mixed specifications, we showed that all three decision problems are EXPTIME-complete in the sizes of these systems. In the case of modal specifications, we proved that the decision problem of common implementation is also EXPTIME-complete in the size of these systems. (The decision problem of consistency for modal specifications is known to be trivial.) However, for the decision problem of modal specifications for thorough refinement, we could not give any new results as our reductions for TR only work for mixed specifications.

In securing these results, our use of a new reduction of input acceptance for linearly bounded alternating Turing machines to the existence of a common implementation for modal specifications was crucial.

Appendix A. Proof of Lemma 7

We need to show that if the linearly bounded ATM T has an accepting computation on input w, then the set $\mathscr{M}_w^T$ of constructed modal specifications will have a common implementation; and, conversely, that if this set $\mathscr{M}_w^T$ of modal specifications has a common implementation, this common implementation witnesses an accepting computation for the linearly bounded ATM T on input w. We will prove the two directions separately.

A.1. *Acceptance implies existence of common implementation*

Let the ATM T accept input w. We will show that $\mathscr{M}_w^T$ has a common implementation. Since we have assumed that T does not repeat configurations on any computation path, we know that there exists a computation tree $\mathsf{T}_{\langle T,w\rangle}$ demonstrating that T accepts w in an exponentially bounded number of steps.

We will use $\mathsf{T}_{\langle T,w\rangle}$ to construct a modal specification

$$N = (N_{\text{states}}, R_N, R_N)$$

over Σ, where N_{states} is a set of states, R_N is a transition relation and Σ is the alphabet of specifications in $\mathscr{M}_w^T$. The proof that N is indeed an implementation of all specifications in $\mathscr{M}_w^T$ will follow shortly after the construction.

Since N has identical must- and may-transition relations, we will just refer to transitions for N without mentioning their type. States of N are labelled by configurations of the computation tree $\mathsf{T}_{\langle T,w\rangle}$. More precisely, we distinguish three kinds of states:

— Type 1 states, indexed by a configuration of $\mathsf{T}_{\langle T,w\rangle}$ only, for example state $n_{\langle q_0,1,w\rangle}$.
— Type 2 states, indexed by a configuration and an extra subscript $\exists$, as in $n_{\langle q,i,\tau\rangle\exists}$.
— Type 3 states, indexed by a configuration and an extra subscript π, as in $n_{\langle q,i,\tau\rangle\pi}$.

We construct N recursively, starting from the root of the accepting computation tree. We start by creating the initial state of N labelled $n_{\langle q_0,1,w\rangle}$, where $\langle q_0,1,w\rangle$ is the configuration of the root node in $\mathsf{T}_{\langle T,w\rangle}$. We shall be adding new successor states and transitions in a top-down fashion as we progress. Our recursive procedure accepts two parameters $(\langle q,i,\tau\rangle, n_{\langle q,i,\tau\rangle})$: a node from $\mathsf{T}_{\langle T,w\rangle}$ and a state from N_{states}. For any pair of parameters $(\langle q,i,\tau\rangle, n_{\langle q,i,\tau\rangle})$ proceed as follows:

— If $\mathsf{mode}(q) = \mathsf{Univ}$, create two new states $n_{\langle q,i,\tau\rangle\exists}$ and $n_{\langle q,i,\tau\rangle\pi}$ and an $\exists$-transition from $n_{\langle q,i,\tau\rangle}$ to $n_{\langle q,i,\tau\rangle\exists}$, and a π-transition from $n_{\langle q,i,\tau\rangle\exists}$ to $n_{\langle q,i,\tau\rangle\pi}$. Then, for each of the successors $\langle q',i',\tau'\rangle$ of $\langle q,i,\tau\rangle$, create a new state $n_{\langle q',i',\tau'\rangle}$ and a transition from $n_{\langle q,i,\tau\rangle\pi}$ to $n_{\langle q',i',\tau'\rangle}$ labelled by (τ_i, i, τ'_i, d) where $d = \mathsf{r}$ if $i' = i+1$ and $d = \mathsf{l}$ otherwise[†]. Then continue recursively for every successor $\langle q',i',\tau'\rangle$ of $\langle q,i,\tau\rangle$, and its corresponding state $n_{\langle q',i',\tau'\rangle}$. See Figure 14(a).
— If $\mathsf{mode}(q) = \mathsf{Exst}$, create two new states $n_{\langle q,i,\tau\rangle\exists}$ and $n_{\langle q,i,\tau\rangle\pi}$ and an $\exists$-transition from $n_{\langle q,i,\tau\rangle}$ to $n_{\langle q,i,\tau\rangle\exists}$ and a π-transition from $n_{\langle q,i,\tau\rangle\exists}$ to $n_{\langle q,i,\tau\rangle\pi}$. Then, because $\mathsf{T}_{\langle T,w\rangle}$ is accepting, we know that there exists at least one successor configuration $\langle q',i',\tau'\rangle$ that is accepted by the subtree with this configuration as root. Select this configuration and create a new state $n_{\langle q',i',\tau'\rangle}$ and a transition from $n_{\langle q,i,\tau\rangle\pi}$ to $n_{\langle q',i',\tau'\rangle}$ labelled by (τ_i, i, τ'_i, d) where $d = \mathsf{r}$ if $i' = i+1$ and $d = \mathsf{l}$ otherwise. Then continue recursively with $\langle q',i',\tau'\rangle$ and $n_{\langle q',i',\tau'\rangle}$. See Figure 14(b).

Observe that the above recursive computation terminates in universal states with no successors due to an iteration over an empty set. This is because $\mathsf{T}_{\langle T,w\rangle}$ is an accepting computation tree, so we are guaranteed that the existential branch can always continue, and, because T only allows execution of a bounded number of steps, every branch of the above recursive procedure will eventually terminate.

We shall now show that specification $(N, n_{\langle q_0,1,w\rangle})$ refines each of the modal specifications in $\mathscr{M}^T_w$.

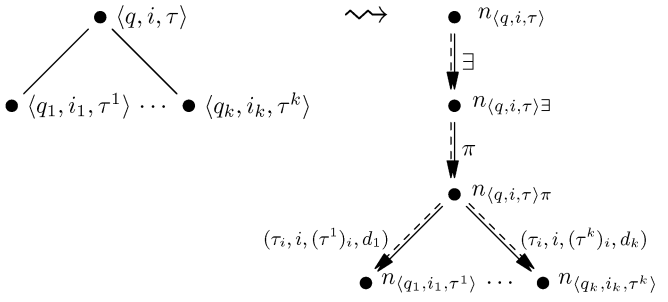
1. $(M_{\text{exist}}, x_1) \prec (N, n_{\langle q_0,1,w\rangle})$:

   Recall that the specification M_{exist} has exactly three states named x_1, x_2 and x_3 (see Figure 8). Consider the following binary relation on states of M_{exist} and states of N:

$$\begin{aligned} Q_1 = \{(x_1, n_{\langle q_s,i,\tau\rangle}) \mid n_{\langle q_s,i,\tau\rangle} \in N_{\text{states}}\} \;\cup \\ \{(x_2, n_{\langle q_s,i,\tau\rangle\exists}) \mid n_{\langle q_s,i,\tau\rangle\exists} \in N_{\text{states}}\} \;\cup \\ \{(x_3, n_{\langle q_s,i,\tau\rangle\pi})) \mid n_{\langle q_s,i,\tau\rangle\pi} \in N_{\text{states}}\}. \end{aligned}$$

[†] We write τ_i to mean the ith symbol of the tape state τ.

(a) Construction for a universal state q. For $l = 1..k$ we define $d_l = \mathsf{r}$ if $i_l = i + 1$ and $d_l = \mathsf{l}$ otherwise.



(b) Construction for an existential state q. Here a selected single $l \in 1..k$ is the index of the accepting successor and $d_l = \mathsf{r}$ if $i_l = i + 1$ and $d_l = \mathsf{l}$ otherwise.

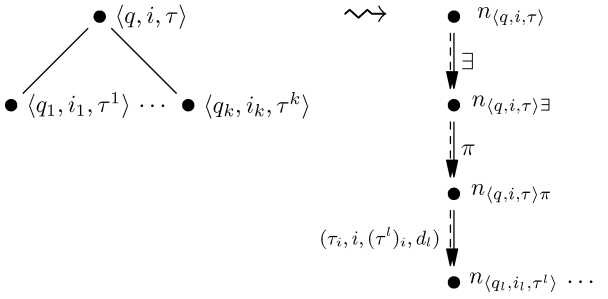


Fig. 14. Construction of a common implementation N from fragments of the accepting computation tree $\mathsf{T}_{\langle T,w \rangle}$.

We will show that Q_1 witnesses a refinement of (M_{exist}, x_1) by $(N, n_{\langle q_0,1,w \rangle})$. First, observe that the pair of initial states $(x_1, n_{\langle q_0,1,w \rangle})$ of M_{exist} and N are related in Q_1. Then we check that Q_1 fulfils the conditions of Definition 2:

(1) We need to show for all pairs $(x, n) \in Q_1$ that for all states x' of M_{exist}, if $(x, a, x') \in R^{\Box}_{M_{\text{exist}}}$, there exists a state $n' \in N_{\text{states}}$ with $(n, a, n') \in R^{\Box}_N$ and $(x', n') \in Q_1$. A must-transition occurs in $R^{\Box}_{M_{\text{exist}}}$ only if $x = x_2$. In this case there is exactly one must π-transition going to x_3. We see from Q_1 that x_2 is paired only with states of form $n = n_{\langle q_s,i,\tau \rangle \exists}$. By construction of N, the latter state always has a must π-transition to some state $n' = n_{\langle q_s,i,\tau \rangle \pi}$ which gives us that $(x', n') \in Q_1$ by the construction of Q_1.

(2) We need to show for all pairs $(x, n) \in Q_1$ that for all states $n' \in N_{\text{states}}$, if $(n, a, n') \in R^{\Diamond}_N$, there exists a state x' of M_{exist} such that $(x, a, x') \in R^{\Diamond}_{M_{\text{exist}}}$ with $(x', n') \in Q_1$. We consider three sub-cases:

— n is of type 1, so $n = n_{\langle q_s,i,\tau \rangle}$:

By Q_1's construction, we have $x = x_1$. By the construction of N, any may-transition leaving n will be labelled by $\exists$ and target a type 2 state $n' = n_{\langle q_s,i,\tau \rangle \exists}$.

This can be matched by $(x_1, \exists, x_2) \in R^{\Diamond}_{M_{\text{exist}}}$ and, for $x' = x_2$, we get $(x', n') \in Q_1$ by construction of Q_1.

— n is of type 2, so $n = n_{\langle q_s,i,\tau\rangle\exists}$:

By Q_1's construction, we have $x = x_2$. By the construction of N, there is exactly one may π-transition leaving n. It targets a state n' of type 3, so $n' = n_{\langle q_s,i,\tau\rangle\pi}$. This can be matched by $(x_2, \pi, x_3) \in R^{\Diamond}_{M_{\text{exist}}}$, so taking $x' = x_3$, we get $(x', n') \in Q_1$ by the construction of Q_1.

— n is of type 3, so $n = n_{\langle q_s,i,\tau\rangle\pi}$:

By Q_1's construction, we have $x = x_3$. By the construction of N, all possible may-transitions leaving n target type 1 states of the form $n' = n_{\langle q_s,i,\tau\rangle}$. All these transitions have labels in $(\_,\_,\_,\_)$. These can all be matched by (M_{exist}, x_3), as that specification contains all transitions of type $(\_,\_,\_,\_)$ going from x_3 to x_1. Since x_1 is paired with all states of type 1 in Q_1 this again gives us that $(x', n') \in Q_1$ for $x' = x_1$.

2. $(M_i, p_{\langle i,w_i\rangle}) \prec (N, n_{\langle q_0,1,w\rangle})$ for each tape cell $1 \leqslant i \leqslant n$:

For any selection of i above, consider the following relation Q^i_2 over the states of M_i and the states of N:

$$Q^i_2 = \{(p_{\langle i,\tau_i\rangle}, n) \mid n = n_{\langle q_s,j,\tau\rangle} \text{ or } n = n_{\langle q_s,j,\tau\rangle\pi} \text{ or } n = n_{\langle q_s,j,\tau\rangle\exists}, \text{ for } 1 \leqslant j \leqslant n\}$$

First note that the initial states of the two specifications are related in Q^i_2. This is clearly the case since the initial state of each M_i is $p_{\langle i,w_i\rangle}$, so by the definition of Q^i_2 it is related to $n_{\langle q_0,1,w\rangle}$. We still need to show, given $(p, n) \in Q^i_2$, that the refinement conditions are preserved:

(1) This condition is vacuously true since M_i's have no must transitions.

(2) We need to show for all pairs $(p, n) \in Q^i_2$ that for all states $n' \in N_{\text{states}}$ if $(n, a, n') \in R^{\Diamond}_N$, there exists a state p' of M_i such that $(p, a, p') \in R^{\Diamond}_{M_i}$ with $(p', n') \in Q^i_2$. With only one exception, whenever N takes a may-transition, M_i will be able to match it. The exception is if the label contains as its old tape symbol a symbol different from the one that M_i has in its current state and where i is the current position of the head in n, so $i = j$. Since the transitions of N are created from a legal computation tree for the ATM T, we can conclude that N will never change the content of the tape without writing to it, so N will never try to read something from a tape cell that is not in that given tape cell. It will also always update the new content of the tape cell correctly, so we are assured that $(p', n') \in Q^i_2$.

3. $(M_{\text{head}}, p_1) \prec (N, n_{\langle q_0,1,w\rangle})$:

The relation Q_3 witnessing this refinement is defined as follows:

$$Q_3 = \{\, (p_i, n) \mid n = n_{\langle q_s,i,\tau\rangle} \text{ or } n = n_{\langle q_s,i,\tau\rangle\pi} \text{ or } n = n_{\langle q_s,i,\tau\rangle\exists} \,\}.$$

We first have to ensure that the initial states of the two specifications are in Q_3. This is the case since the initial state of N has $i = 1$, which is Q_3-related to p_1, the initial state of M_{head}. We now need to show that for any given $(p, n) \in Q_3$, the two refinement conditions of Definition 2 are preserved:

(1) This condition is vacuously satisfied M_{head} has no must-transitions.

(2) We need to show that whenever $(n, a, n') \in R_N^{\Diamond}$, there exists p', a state of M_{head}, such that $(p, a, p') \in R_{M_{\text{head}}}^{\Diamond}$ with $(p', n') \in Q_3$. We will just discuss the case when n is of type 3 here, so $n = n_{\langle q_s, i, \tau\rangle\pi}$, since for the other two types the transitions leaving n do not move the head and the preservation of refinement can be concluded directly.

By construction of N, whenever $n_{\langle q_s, i, \tau\rangle\pi}$ takes a may-transition, this transition is labelled $(\_, i, \_, d)$ targeting a type 1 state $n_{\langle q', i', \tau'\rangle}$, where $i' = i + 1$ if $d = \mathsf{r}$ and $i' = i - 1$ otherwise. Now, by the construction of M_{head}, the state p_i can match such a transition, moving to $p_{i'}$ accordingly. The only case where M_{head} would not be able to match is if N tried to move the head off either end of the tape, but this will never happen since N is constructed from a legal accepting computation tree. Thus we conclude that the refinement condition is preserved.

4. $(M_{\text{ctrl}}, q_0) \prec (N, n_{\langle q_0, 1, w\rangle})$:

Consider the following binary relation Q_4 on states of M_{ctrl} and N:

$$\begin{aligned} Q_4 = \{(q_s, n) \mid n = n_{\langle q_s, i, \tau\rangle}\} \cup \\ \{(q_{s\exists}, n) \mid n = n_{\langle q_s, i, \tau\rangle\exists}\} \cup \\ \{(\langle q_s \tau_i i\rangle, n_{\langle q_s, i, \tau\rangle\pi}) \mid \mathsf{mode}(q_s) = \mathsf{Univ}\} \cup \\ \{(\langle q_s \tau_i i a_2 d q_t\rangle, n_{\langle q_s, i, \tau\rangle\pi}) \mid \mathsf{mode}(q_s) = \mathsf{Exst} \text{ and} \\ (n_{\langle q_s, i, \tau\rangle\pi}, (\tau_i, i, a_2, d), n_{\langle q_t, i', \tau'\rangle}) \in R_N^{\Diamond}\} \ . \end{aligned}$$

First observe that the initial states of the two specifications are in Q_4 since q_0 is the initial state of M_{ctrl} and $n_{\langle q_0, 1, w\rangle}$ is the initial state of N (see the first summand in the definition of Q_4). Now we need to show that, given a pair $(q, n) \in Q_4$, the two refinement conditions of Definition 2 are preserved:

(1) We need to show that whenever $(q, a, q') \in R_{M_{\text{ctrl}}}^{\Box}$, there exists a state $n' \in N_{\text{states}}$ such that $(n, a, n') \in R_N^{\Box}$ with $(q', n') \in Q_4$.

We need to consider four cases:

— $q = q_s$ for some $q_s \in Q$ (a state of the ATM T):

There is exactly one must $\exists$-transition leaving it, which targets $q_{s\exists}$. This transition can be matched by an $\exists$-transition leaving $n_{\langle q_s, i, \tau\rangle}$ and targeting $n_{\langle q_s, i, \tau\rangle\exists}$. These new target states remain in relation Q_4, as in the above definition.

— $q = q_{s\exists}$ for some $q_s \in Q$ (a state of the ATM T):

The condition is satisfied vacuously simply because there is no must-transition leaving q.

— q has the form $\langle q_s \tau_i i\rangle$, where q_s is a universal state of the ATM T:

n has the form $n_{\langle q_s, i, \tau\rangle\pi}$, but since $n_{\langle q_s, i, \tau\rangle\pi}$ was constructed by our recursive procedure from a universal configuration of an accepting computation tree, we know that for all must-transitions leaving $\langle q_s \tau_i i\rangle$ to some state q_t, there will be a matching must-transition in N leaving $n_{\langle q_s, i, \tau\rangle\pi}$ and targeting $n_{\langle q_t, i', \tau'\rangle}$, which is in relation with q_t as in the first summand in the definition of Q_4.

— q has the form $\langle q_s \tau_i i a_2 d q_t \rangle$, where q_s is an existential state of the ATM T:
n has the form $n_{\langle q_s,i,\tau\rangle\pi}$ and the state $\langle q_s \tau_i i a_2 d q_t \rangle$ has exactly one must-transition labelled (τ_i, i, a_2, d) and targeting state q_t. Since q_s is an existential state, we know that $n_{\langle q_s,i,\tau\rangle\pi}$ was constructed from an existential configuration and, consequently, there is a single must-transition leaving it. This transition is labelled (τ_i, i, a_2, d) as in the construction of the Q_4 relation (see the last summand). Finally, this transition targets $n' = n_{\langle q_t,i',\tau'\rangle}$, so we again have $(q', n') \in Q_4$.

(2) We need to show that if $(n, a, n') \in R_N^{\Diamond}$, there exists a state q' of M_{ctrl} such that $(q, a, q') \in R_{M_{\text{ctrl}}}^{\Diamond}$ with $(q', n') \in Q_4$.
We consider three cases according to the type of state n:

— n is of type 1, so $n = n_{\langle q_s,i,\tau\rangle}$:
By the construction of N, there is a may $\exists$-transition leaving n targeting $n_{\langle q_s,i,\tau\rangle\exists}$. This is followed by $(q_s, \exists, q_{s\exists}) \in R_{M_{\text{ctrl}}}^{\Diamond}$ and again gives us that $(q', n') \in Q_4$.

— n is of type 2, so $n = n_{\langle q_s,i,\tau\rangle\exists}$:
By the construction of Q_4 (see the second summand), q is of the form $q_{s\exists}$. By the construction procedure for N, there is a single may π-transition leaving $n_{\langle q_s,i,\tau\rangle\exists}$ and targeting $n' = n_{\langle q_s,i,\tau\rangle\pi}$.
– If $\mathsf{mode}(q_s) = \mathsf{Univ}$, there is exactly one transition $(q_{s\exists}, \pi, \langle q_s \tau_i i\rangle) \in R_{M_{\text{ctrl}}}^{\Diamond}$, and its target state is related to $n_{\langle q_s,i,\tau\rangle\pi}$ in Q_4.
– If $\mathsf{mode}(q_s) = \mathsf{Exst}$, there can be many may π-transitions leaving $q_{s\exists}$. We will choose which one to match with, based on the label of the single transition leaving $n_{\langle q_s,i,\tau\rangle\pi}$. We are, so to speak, looking one step ahead. Since $n_{\langle q_s,i,\tau\rangle\pi}$ says that the head is in position i over a tape containing τ, we choose to match our transition with the transition of M_{ctrl} targeting the state whose name matches the prefix '$\langle q_s \tau_i i$'. Such a state always exists by construction of M_{ctrl}, and it is exactly the state that is related to $n_{\langle q_s,i,\tau\rangle\pi}$ in Q_4 (see the last summand).

— n is of type 3, so $n = n_{\langle q_s,i,\tau\rangle\pi}$:
We consider two cases according to the mode of q_s in the ATM T:
– $\mathsf{mode}(q_s) = \mathsf{Univ}$:
There may be several may-transitions leaving $n_{\langle q_s,i,\tau\rangle\pi}$. Since N has been created from a legal computation tree, we know that any may-transition leaving $n_{\langle q_s,i,\tau\rangle\pi}$ and targeting $n' = n_{\langle q_t,i',\tau'\rangle}$ follows the transition relation δ of T. Moreover, by the construction of M_{ctrl}, its state $\langle q_s \tau_i i\rangle$ will consequently be able to match this transition arriving in the state q_t related to n' in Q_4.
– $\mathsf{mode}(q_s) = \mathsf{Exst}$:
There is exactly one may-transition leaving $n_{\langle q_s,i,\tau\rangle\pi}$ and exactly one may-transition leaving $\langle q_s \tau_i i a_2 d q_t \rangle$. These transitions have the same label and have target states $n_{\langle q_t,i',\tau'\rangle}$ and q_t, respectively, which are related in Q_4.

This concludes the argument that each specification in $\mathscr{M}_w^T$ is refined by N.

A.2. *The existence of a common implementation implies acceptance*

Let $\mathscr{M}_w^T$ have a common implementation. We need to show that the ATM T accepts input w. Given a modal specification

$$U_{\text{new}} = (U_{\text{states}}, R_U, R_U)$$

that is a common implementation of $\mathscr{M}_w^T$, we will construct a computation tree $\mathsf{T}_{\langle M,w\rangle}$ demonstrating that T accepts w.

Since U_{new} is a common implementation of $\mathscr{M}_w^T$, we have $3+n$ refinement relations,

$$Q_{\text{ctrl}},\ Q_{\text{head}},\ Q_{\text{exist}},\ Q_1,\ \ldots,\ Q_n,$$

each demonstrating for one of the corresponding specifications $S \in \mathscr{M}_w^T$ that $S \prec U_{\text{new}}$.

The construction of $\mathsf{T}_{\langle M,w\rangle}$ is inductive. Along with the construction, we argue that the nodes of the tree preserve the following property (**IH**):

(1) For every configuration $\langle q, i, \tau\rangle$ of $\mathsf{T}_{\langle M,w\rangle}$ there exists a state $u_x \in U_{\text{states}}$ such that:

(IH1) $(u_x, x_1) \in Q_{\text{exist}}$;
(IH2) $(u_x, q) \in Q_{\text{ctrl}}$;
(IH3) $(u_x, p_i) \in Q_{\text{head}}$;
(IH4) $(u_x, p_{\langle k,\tau_k\rangle}) \in Q_k$ for each $k = 1..n$.

(We follow the conventions of Section 4 here. So q is a name of T's state, which also uniquely identifies a state of M_{ctrl}. Specifically, we mean that q represents a label without any special suffixes. Label p_i refers to a particular state of M_{head}, the one representing position i. Similarly, $p_{\langle k,\tau_k\rangle}$ denotes the state of M_k that represents the fact that the kth symbol of τ is stored in the kth cell of the tape.)

(2) Moreover:

(IH5) if a configuration $\langle q', i', \tau'\rangle$ is a successor of $\langle q, i, \tau\rangle$ in $\mathsf{T}_{\langle M,w\rangle}$, then it is also a successor of $\langle q, i, \tau\rangle$ in the ATM T;

and, conversely:

(IH6) the tree $\mathsf{T}_{\langle M,w\rangle}$ has all the successors of $\langle q, i, \tau\rangle$ that T has for universal states, and at least one of them for all existential states.

We will address the problem of whether $\mathsf{T}_{\langle M,w\rangle}$ actually is an accepting computation tree of T, witnessing acceptance of w, after discussing the construction of $\mathsf{T}_{\langle M,w\rangle}$, and after arguing that it satisfies the above inductive property.

Root (base case):
The root of $\mathsf{T}_{\langle M,w\rangle}$ is selected to be the configuration $\langle q_0, 1, w\rangle$, where q_0 is the initial control state of T. We need to show that $\langle q_0, 1, w\rangle$ exhibits property IH. Observe that U_{new} has a distinct initial state u_0. Take u_x to be this u_0.

(IH2) Since $M_{\text{ctrl}} \prec U_{\text{new}}$, there is a pair $(u_0, q_0) \in Q_{\text{ctrl}}$.
(IH3) Since $M_{\text{head}} \prec U_{\text{new}}$ and p_1 is the initial state of M_{head}, we know that $(u_0, p_1) \in Q_{\text{head}}$.
(IH4) Since w is the initial content of the tape, and thus $p_{\langle k,w_k\rangle}$ is an initial state of M_k, the refinement $M_k \prec U_{\text{new}}$ gives us that $(u_0, p_{\langle k,w_k\rangle}) \in Q_k$, so IH4 holds for $\langle q_0, 1, w\rangle$.

(IH1) Since $M_{\text{exist}} \prec_{U_{\text{new}}}$, we get that $(u_0, x_1) \in Q_{\text{exist}}$.

We shall argue that IH5 and IH6 hold for the root node when we discuss adding successors below, so this concludes the base case.

Non-root nodes (inductive step):
Given a configuration $\langle q, i, \tau \rangle$ for which properties IH1–IH4 hold, we will now construct the next level of $\mathsf{T}_{\langle M,w \rangle}$ in such a way that IH5–IH6 hold for $\langle q, i, \tau \rangle$ and IH1–IH4 hold for all its successors.

Before we consider the two cases based on the modes of the states separately, we shall describe the part of the proof common to both of them. The induction hypothesis allows us to assume existence of a specific state u_x of U_{states} and the respective refinement relations. Since the state u_x is related to a state without a π or $\exists$ subscript in M_{ctrl}, that u_x must implement an $\exists$ transition to a new state, which we will call $u_{x\exists}$. Because $(u_x, x_1) \in Q_{\text{exist}}$, we know that $(u_{x\exists}, x_2) \in Q_{\text{exist}}$, so $u_{x\exists}$ must implement a π transition to a new state, say $u_{x\pi}$. Since all π and $\exists$ transitions in M_{head} and M_1 up to M_n are loops, we know that $u_{x\pi}$ is related to the same states as u_x in these specifications.

The remainder of the proof, consists of a case analysis on the mode of q:

— $\mathsf{mode}(q) = \mathsf{Exst}$:
We know that (M_{ctrl}, q) has to implement an $\exists$-transition followed by at least one π-transition reaching a state of the form $\langle q\tau_i i a' d q' \rangle$. Also, because $u_{x\exists}$ is related to $q_\exists$, it must be possible to choose $u_{x\pi}$ above such that $(u_{x\pi}, \langle q\tau_i i a' d q' \rangle) \in Q_{\text{ctrl}}$, but then we know that $u_{x\pi}$ can take a transition labelled (τ_i, i, a', d) to some state u'_x related to q' in Q_{ctrl}.
So, if we extend $\mathsf{T}_{\langle M,w \rangle}$ at $\langle q, i, \tau \rangle$ with a new child $\langle q', i', \tau[\tau_i \mapsto a'] \rangle$, the new execution step will follow the semantics of the ATM T satisfying conditions IH1–IH6, provided $i' = i + 1$ if $d = \mathsf{r}$, and $i' = i - 1$ otherwise.
The argument that IH5–IH6 hold is direct since we have added a successor as required out of all those available in the semantics of T.
The arguments showing that IH1–IH4 hold are more involved, but standard – for each of them a unique successor in M_{exist}, M_{ctrl}, M_{head} and M_k's can be identified by following the transition labelled (τ_i, i, a', d), and then shown to witness fullfilment of the condition for u'_x by the induction hypothesis (from refinement of u_x).

— $\mathsf{mode}(q) = \mathsf{Univ}$:
Since (U_{new}, u_x) is a refinement of (M_{ctrl}, q) and (M_{exist}, x_1), we get that it is possible to choose $u_{x\pi}$ above so that it refines a state of M_{ctrl} that has a label of the form $(M_{\text{ctrl}}, \langle q\tau_i i \rangle)$.
The refinement relation with M_{head} and M_i ensures that this state is the only successor of q in M_{ctrl} that can be implemented, implying that $u_{x\pi}$ must implement all the transitions corresponding to the transition relation δ of T.
So we can extend $\mathsf{T}_{\langle M,w \rangle}$ with new children $\langle q', i', \tau'[\tau_i \mapsto a'] \rangle$ for all (q', i', τ') such that (M_{ctrl}, q') can be reached from $(M_{\text{ctrl}}, \langle q\tau_i i \rangle)$ in one step with a transition labelled (τ', i, a', d). Also $i' = i + 1$ if $d = \mathsf{r}$, and $i' = i - 1$ otherwise.

Again, it is not hard to see that all the newly added successors maintain the induction hypothesis.

We now have to prove that the induction hypothesis holds for all of these target states. As they are all reached by a transition in M_{ctrl}, we know that there exists a state $u_y \in U_{\text{states}}$ such that $(u_y, q') \in Q_{\text{ctrl}}$. Because of the label on the transition, we also know that $(u_y, p_l) \in Q_{\text{head}}$ for $l = i+1$ if $d = \mathsf{r}$, and $l = i-1$ if $d = \mathsf{l}$. This is also ensured to be done in such a way that the tape cell specifications M_1 to M_n again match the content of the tape. We also know, because of all the transitions of type $(\_, \_, \_, \_)$ going from x_3 to x_1 in M_{exist}, that $(u_y, x_1) \in Q_{\text{exist}}$. This completes the proof of the inductive step.

In this way we can construct a pruned computation tree $\mathsf{T}_{\langle M,w\rangle}$ recursively. The constructed tree is finite because we have argued that it follows the semantics of the ATM T, and T repeats no configuration along a single computation path. Moreover, $\mathsf{T}_{\langle M,w\rangle}$ is accepting as it is never stuck in a rejecting (existential) state.

Acknowledgments

We thank Nir Piterman for having pointed out to us that the constructions for prime numbers in Berwanger *et al.* (2008) and Berwanger and Doyen (2008) are similar to the constructions for prime numbers given in Example 2. We also thank Jiří Srba and Jan Křetínský for sharing with us their recent discovery that TR for modal specifications is EXPTIME-complete.

References

Antonik, A. (2008) *Decision problems for partial specifications: empirical and worst-case complexity*, Ph.D. thesis, Imperial College, London.

Antonik, A. and Huth, M. (2009) On the complexity of semantic self-minimization. In: Proc. AVOCS 2007. *Electronic Notes in Theoretical Computer Science* **250** 3–19.

Antonik, A., Huth, M., Larsen, K. G., Nyman, U. and Wąsowski, A. (2008a) 20 years of modal and mixed specifications. *Bulletin of EATCS* **95**. (Available at `http://processalgebra.blogspot.com/2008/05/concurrency-column-for-beatcs-june-2008.html`.)

Antonik, A., Huth, M., Larsen, K. G., Nyman, U. and Wąsowski, A. (2008b) Complexity of decision problems for mixed and modal specifications. In: FoSSaCS'08. *Springer-Verlag Lecture Notes in Computer Science* **4962** 112–126.

Antonik, A., Huth, M., Larsen, K. G., Nyman, U. and Wąsowski, A. (2008c) Exptime-complete decision problems for modal and mixed specifications. In: 15th International Workshop on Expressiveness in Concurrency. *Electronic Notes in Theoretical Computer Science* **242** 19–33.

Beneš, N., Křetínský, J., Larsen, K. G. and Srba, J. (2009) Checking thorough refinement on modal transition systems is EXPTIME-complete. In: Proceedings of the 6th International Colloquium on Theoretical Aspects of Computing. *Springer-Verlag Lecture Notes in Computer Science* **5684** 112–126.

Berwanger, D., Chatterjee, K., Doyen, L., Henzinger, T. A. and Raje, S. (2008) Strategy construction for parity games with imperfect information. In: Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08). *Springer-Verlag Lecture Notes in Computer Science* **5201** 325–339.

Berwanger, D. and Doyen, L. (2008) On the power of imperfect information. In: Proceedings of the 28th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'08), Bangalore, India, December 2008. Available at `http://drops.dagstuhl.de/portals/FSTTCS08/`.

Børjesson, A., Larsen, K. G. and Skou, A. (1993) Generality in design and compositional verification using tav. In: *FORTE '92 Proceedings*, North-Holland Publishing Co. 449–464.

Brunet, G., Chechik, M. and Uchitel, S. (2006) Properties of behavioural model merging. In: Misra, J., Nipkow, T. and Sekerinski, E. (eds.) FM. *Springer-Verlag Lecture Notes in Computer Science* **4085** 98–114.

Bruns, G. (1997) An industrial application of modal process logic. *Sci. Comput. Program.* **29** (1-2) 3–22.

Bruns, G. and Godefroid, P. (2000) Generalized model checking: Reasoning about partial state spaces. In: Palamidessi, C. (ed.) CONCUR. *Springer-Verlag Lecture Notes in Computer Science* **1877** 168–182.

Chandra, A. K., Kozen, D. and Stockmeyer, L. J. (1981) Alternation. *J. ACM* **28** (1) 114–133.

Chebyshev, P. (1852) La totalité des nombres premiers inférieurs a une limite donnée. *Journal de Mathematiques Pures et Appliques* **17** 341–365.

Clarke, E. M., Grumberg, O. and Long, D. E. (1994) Model checking and abstraction. *ACM Trans. Program. Lang. Syst.* **16** (5) 1512–1542.

Dams, D. (1996) *Abstract Interpretation and Partition Refinement for Model Checking*, Ph.D. thesis, Eindhoven University of Technology.

Fischbein, D., Uchitel, S. and Braberman, V. (2006) A foundation for behavioural conformance in software product line architectures. In: *ROSATEA '06 Proceedings*, ACM Press 39–48.

Godefroid, P. and Huth, M. (2005) Model checking vs. generalized model checking: Semantic minimizations for temporal logics. In: *Proceedings of the Twentieth Annual IEEE Symp. on Logic in Computer Science, LICS 2005*, IEEE Computer Society Press 158–167.

Godefroid, P., Huth, M. and Jagadeesan, R. (2001) Abstraction-based model checking using modal transition systems. In: Larsen, K. G. and Nielsen, M. (eds.) CONCUR 2001 – concurrency theory: 12th international conference, Aalborg, Denmark. *Springer-Verlag Lecture Notes in Computer Science* **2154** 426–440.

Gurfinkel, A., Wei, O. and Chechik, M. (2006) Yasm: A software model-checker for verification and refutation. In: Ball, T. and Jones, R. B. (eds.) CAV. *Springer-Verlag Lecture Notes in Computer Science* **4144** 170–174.

Huth, M. (2005a) Labelled transition systems as a Stone space. *Logical Methods in Computer Science* **1** (1) 1–28.

Huth, M. (2005b) Refinement is complete for implementations. *Formal Asp. Comput.* **17** (2) 113–137.

Huth, M., Jagadeesan, R. and Schmidt, D. (2001) Modal transition systems: A foundation for three-valued program analysis. *Springer-Verlag Lecture Notes in Computer Science* **2028**.

Hüttel, H. (1988) Operational and denotational properties of modal process logic. Master's thesis, Computer Science Department, Aalborg University.

Kozen, D. (1988) A finite model theorem for the propositional μ-calculus. *Studia Logica* **47** (3) 233–241.

Landweber, P. S. (1963) Three theorems on phrase structure grammars of type 1. *Information and Control* **6** (2) 131 – 136.

Laroussinie, F. and Sproston, J. (2007) State explosion in almost-sure probabilistic reachability. *Inf. Process. Lett.* **102** (6) 236–241.

Larsen, K. G. (1989) Modal specifications. In Sifakis, J. (ed.) Automatic Verification Methods for Finite State Systems. *Springer-Verlag Lecture Notes in Computer Science* **407** 232–246.

Larsen, K. G., Nyman, U. and Wąsowski, A. (2007a) Modal I/O automata for interface and product line theories. In: Nicola, R. D. (ed.) ESOP. *Springer-Verlag Lecture Notes in Computer Science* **4421** 64–79.

Larsen, K. G., Nyman, U. and Wąsowski, A. (2007b) On modal refinement and consistency. In: Caires, L. and Vasconcelos, V. T. (eds.) CONCUR 2007. *Springer-Verlag Lecture Notes in Computer Science* **4703** 105–119.

Larsen, K. G., Steffen, B. and Weise, C. (1995) A constraint oriented proof methodology based on modal transition systems. In: Proceedings of the First International Workshop on Tools and Algorithms for Construction and Analysis of Systems. *Springer-Verlag Lecture Notes in Computer Science* **1019** 17–40.

Larsen, K. G. and Thomsen, B. (1988) A modal process logic. In: *Third Annual IEEE Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society 203–210.

Larsen, K. G. and Xinxin, L. (1990) Equation solving using modal transition systems. In: *Fifth Annual IEEE Symposium on Logics in Computer Science (LICS)*, IEEE Computer Society 108–117.

Nyman, U. (2008) *Modal Transition Systems as the Basis for Interface Theories and Product Lines*, Ph.D. thesis, Department of Computer Science, Aalborg University.

Park, D. (1981) Concurrency and automata on infinite sequences. In: Proceedings of the 5th GI-Conference on Theoretical Computer Science. *Springer-Verlag Lecture Notes in Computer Science* **104** 167–183.

Raclet, J.-B. (2008) Residual for component specifications. *Electronic Notes in Theoretical Computer Science* **215** 93–110.

Schmidt, D. (2001) From trace sets to modal-transition systems by stepwise abstract interpretation.

Schmidt, H. and Fecher, H. (2007) Comparing disjunctive modal transition systems with a one-selecting variant. (Submitted for publication.)

Sipser, M. (1996) *Introduction to the Theory of Computation*, PWS Publishing Company.

Uchitel, S. and Chechik, M. (2004) Merging partial behavioural models. In: Taylor, R. N. and Dwyer, M. B. (eds.) *SIGSOFT Software Engineering Notes* **29** (6) 43–52.

Wilke, Th. (2001) Alternating tree automata, parity games, and modal μ-calculus. *Bull. Soc. Math. Belg.* **8** (2).

Xinxin, L. (1992) *Specification and Decomposition in Concurrency*, Ph.D. thesis, Department of Mathematics and Comnputer Science, Aalborg University.

Zielonka, W. (1998) Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.* **200** (1-2) 135–183.