Modal and mixed specifications: key decision problems and their complexities

ADAM ANTONIK, MICHAEL HUTH, KIM G. LARSEN, ULRIK NYMAN and ANDRZEJ WĄSOWSKI

Mathematical Structures in Computer Science / Volume 20 / Special Issue 01 / February 2010, pp 75 - 103
DOI: 10.1017/S0960129509990260, Published online: 26 February 2010

Link to this article: http://journals.cambridge.org/abstract_S0960129509990260

How to cite this article:

Request Permissions : Click here
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 $\square$ 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).
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 \( \Sigma \) 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 \((N, t_0) = ((S_N, R^\Box_N, R^\Diamond_N), t_0)\) refines another pointed, mixed specification \((M, s_0) = ((S_M, R^\Box_M, R^\Diamond_M), s_0)\) over the same alphabet \( \Sigma \), 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 \((s, a, s') \in R^\Box_M\), there exists some \((t, a, t') \in R^\Box_N\) with \((s', t') \in Q\); 
2. for all \((t, a, t') \in R^\Diamond_N\), there exists some \((s, a, s') \in R^\Diamond_M\) 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 \((s_0, \pi, s_2) \in R^\bigcap_M\), we can infer that there must be some \(x\) with \((t_0, \pi, x) \in R^\bigcap_N\) and \((x, s_2) \in Q\). In particular, \(x\) can only be \(t_1\) or \(t_2\). If \(x\) is \(t_1\), then since \((s_2, \pi, s_4) \in R^\bigcap_M\) and \((t_1, s_2) \in Q\), there has to be some \(R^\bigcap_N\) transition out of \(t_1\), which is not the case. If \(x\) is \(t_2\), then \((t_2, \pi, t_3) \in R^\bigcap_N\) and \((t_2, s_2) \in Q\) imply that there is some \(R^\bigcap_M\) 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 \(\Sigma\) 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 \(\mu\)-calculus formula \(\Psi_{(M,s)}\) (Huth 2005a) is satisfiable. In general, a transition system satisfying a modal \(\mu\)-calculus formula may be infinite. The small model theorem for \(\mu\)-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 \(\mu\)-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 \(\neg \Psi_{(N,t)} \lor \Psi_{(M,s)}\).

Validity checking of such vectorised modal \(\mu\)-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 \(A_{(M,s)}\) (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 \(A_{(M,s)}\). The formulae \(\Psi_{(M,s)}\) can be written as a system of recursive equations (Larsen 1989) \(X_s = \text{body}_s\) for each state \(s\) of \(M\). We can therefore construct all \(A_{(M,s)}\) in a compositional manner: whenever \(X_s\) refers in its \(\text{body}_s\) to some \(X_t\), we ensure \(A_{(M,s)}\) has a transition to the initial state of \(A_{(M,t)}\) at that point. This \(A_{(M,s)}\) 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 \(A_{(M,s)}\) 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 ATM_{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 $s^j_i, j = 1 \ldots i$, such that $(s^j_i, \pi, s^{j+1}_i) \in R^\square, R^\circ$ for $1 \leq j \leq i-1$ and $(s^i_i, \pi, s^i_i) \in R^\circ$;
- an extra deadlock state $d$ such that $(s^l_i, a_i, d) \in R^\circ$ if $l \in I - \{i\}$, while $(s^i_i, a_i, d) \in R^\circ, R^\square$. 
Fig. 3. Pointed specifications $(M_2, s_2)$, $(M_3, s_3)$ and $(M_5, s_5)$ 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 $\pi$, allowing the implementation to stop after $i - 1$ $\pi$-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 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 $T = (Q, \Gamma, \delta, q_0, \text{mode})$, where $Q$ is a non-empty finite set of control states, $\Gamma$ is an alphabet of
Fig. 4. The transition relation of an ATM as a labelled graph and as a function.

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

$$\delta : Q \times (\Gamma \cup \{\text{null}\}) \rightarrow \mathcal{P}(Q \times \Gamma \times \{l, r\})$$

is a transition relation, $q_0 \in Q$ is the initial control state and $\text{mode} : Q \rightarrow \{\text{Univ}, \text{Exst}\}$ 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 $\Gamma$. 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: $(q', a', r) \in \delta(q, a)$, 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 $\Gamma \cup \{\text{null}\}$.

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 $i$th cell from the left and $\tau \in (\Gamma \cup \text{null})^\omega$ is the current tape state.

2 For input $w \in \Gamma^*$, the initial configuration is $\langle q_0, 1, w\text{null}^\omega \rangle$.

3 The recursive and parallel execution of all applicable transitions $\delta$ from initial configuration $\langle q_0, 1, w\text{null}^\omega \rangle$ yields a computation tree $T_{(T, w)}$. We say that ATM $T$ accepts input $w$ if and only if the tree $T_{(T, w)}$ accepts, where the latter is defined recursively:

- Subtree $T_{(T, w)}$ with root $\langle q, i, \tau \rangle$ and $\text{mode}(q) = \text{Exst}$ accepts if and only if there is a successor $\langle q', i', \tau' \rangle$ of $\langle q, i, \tau \rangle$ in $T_{(T, w)}$ such that the sub-tree with root $\langle q', i', \tau' \rangle$ accepts.

† Transitions $(\_, \_, l)$ are not applicable in configurations $(\_, 1, \_)$ as the head cannot move over the left boundary of the tape, where we use _ as a wildcard.
Fig. 5. An accepting computation tree $T_{\langle T, 0101null^\omega \rangle}$ for the ATM $T$ of Example 4.

— Subtree $T_{\langle T, w \rangle}$ with root $\langle q, i, \tau \rangle$ and $\text{mode}(q) = \text{Univ}$ accepts if and only if for all successors $\langle q', i', \tau' \rangle$ of $\langle q, i, \tau \rangle$ in $T_{\langle T, w \rangle}$, 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 $0101null^\omega$.

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 $T_{\langle T, w \rangle}$ only contains configurations $\langle q, i, vnull^\omega \rangle$, 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 $T_{\langle T, w \rangle}$, and removing all the remaining successors and configurations unreachable from the root, one can create a smaller tree that only contains configurations with $\langle q, i, vnull^\omega \rangle$ where $|v| \leq |w|$. 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 $\text{ASPACE} = \text{EXPTIME}$ (Sipser 1996, Theorem 10.18).

Fact 1. Consider the formal language

$$\text{ATMLB} = \{ \langle T, w \rangle \mid w \in \Gamma^* \text{ is accepted by linearly bounded ATM } T \}.$$
The problem of deciding whether for an arbitrary linearly bounded ATM $T$ and an input $w$, the pair $\langle T, w \rangle$ is in $\text{ATM}_{LB}$ 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 $\Sigma$. 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 $\text{ATM}_{LB}$. Given an ATM $T$ and an input word $w$ of length $n$, we synthesise a collection of (pointed) modal specifications $\mathcal{M}^T_w = \{ M_i \mid 1 \leq i \leq n \} \cup \{ M_{\text{head}}, M_{\text{ctrl}}, M_{\text{exist}} \}$, (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 $\mathcal{M}^T_w$.

The specifications $M_i$, $M_{\text{head}}$, $M_{\text{ctrl}}$ and $M_{\text{exist}}$ 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 $\mathcal{M}^T_w$ both formally and through our running example in Figure 4. All specifications in $\mathcal{M}^T_w$ 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 $i$th 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 $\pi$, are used to encode logical constraints like disjunction and conjunction. The alphabet for our running example is

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

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 $\{p_{(i, a)}\}_{a \in \Gamma}$ and initial state $p_{(i, w_i)}$, representing the initial contents of the $i$th cell. There are no must-transitions:

$$R^\square = \emptyset.$$  

The may-transition relation connects any two states:

$$\text{for all symbols } a_1, a_2 \text{ in } \Gamma \text{ we have } (p_{(i, a_1)}, (a_1, i, a_2, -), p_{(i, a_2)}) \in R^\circ.$$
Changes in cells other than $i$ are also consistent with $M_i$:

for all $a \in \Gamma$ if $i \neq j$ with $1 \leq j \leq n$, then $(p_{(i,a)}, (_{-}, i, _{-}, r), p_{(i,a)}) \in R^\triangledown$.

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

$(p_{(i,a)}, \pi, p_{(i,a)}) \in R^\triangledown$ and $(p_{(i,a)}, \exists, p_{(i,a)}) \in R^\triangledown$ 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 $M_{\text{head}}$, 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 $M_{\text{head}}$. There are no must-transitions:

$R^\Box = \emptyset$.

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

for every position $1 \leq i < n$ we have $(p_i, (_{-}, i, _{-}, r), p_{i+1}) \in R^\triangledown$

for every position $1 < i \leq n$ we have $(p_i, (_{-}, i, _{-}, l), p_{i-1}) \in R^\triangledown$.

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

$(p_i, \pi, p_i) \in R^\triangledown$ and $(p_i, \exists, p_i) \in R^\triangledown$ for each position $1 \leq i \leq n$.

There are no other may-transitions in $M_{\text{head}}$. Note that the head of $T$ is only allowed to move between the first and $n$th cell in any computation. Figure 7 shows the specification $M_{\text{head}}$ for our running example.
Fig. 8. The specification $M_{\text{exist}}$, which enforces a $\pi$-transition after each $\exists$-transition.

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

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

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

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

\[
(q_s^{\exists}, \pi, (q_s a_{\text{old}} i a_{\text{new}} d q_t)) \in R^\Box
\]

\[
((q_s a_{\text{old}} i a_{\text{new}} d q_t), (a_{\text{old}}, i, a_{\text{new}}, d), q_t) \in R^\exists \cap R^\Box.
\]

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

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

\[
(q_s^{\exists}, \pi, (q_s a_{\text{old}} i)) \in R^\Box
\]

\[
((q_s a_{\text{old}} i), (a_{\text{old}}, i, a_{\text{new}}, d), q_t) \in R^\exists \cap R^\Box.
\]

The initial state of $M_{\text{ctrl}}$ 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 $M_{\text{ctrl}}$ specification for our running example is shown in Figure 11.

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

This concludes the description of all specifications from set $\mathcal{M}_T^w$ 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 $\mathcal{M}^T_w$ 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 $T_{(T,w)}$, one can construct a specification $N$ by structural induction on $T_{(T,w)}$. This $N$ effectively adds to $T_{(T,w)}$ some new states and labelled transitions so that the computation encoded in $T_{(T,w)}$ then interlocks with the action synchronisation of specifications in $\mathcal{M}^T_w$. Since $N$ is of the form...
Fig. 11. The entire $M_{\text{ctrl}}$ 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 $M_T^w$. This is a lengthy but routine argument.

For the converse, a common implementation of $M_T^w$ 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 $((S, R^\square, R^\boxplus), s_0)$, one such implementation is $(S, R^\square, R^\boxplus, s_0)$. 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 \((N, t) \prec_{th} (M', s')\).

\[N: \quad M':\]

\[
t \quad \prec_{th} \quad s' \quad \pi \quad s\]

\[
\text{M}\]

s\(_2\), two new \(\pi\)-transitions are added from a fresh state \(c_2\) to each of \(s_1\) and \(s_2\). One of the \(\pi\)-transitions is an \(R^\ominus \setminus R^\Box\) \(\pi\)-transition and the other is an \(R^\Box\) \(\pi\)-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 \(\pi\)-transitions to the newly conjoined systems. Observe that the resulting specification is properly mixed (not modal) since it contains \(\pi\)-transitions that are in \(R^\ominus \setminus R^\Box\). 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 \(k - 1\) \(\pi\)-transitions. Conversely, if the new mixed specification has an implementation, this implementation will contain at least a sequence of \(k - 1\) \(\pi\)-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 \(\Sigma\). 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 \((s', \pi, s) \in R^\ominus_{M'} R^\Box_{M'}\). 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 \((l', \pi, l)\). 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 \((l', \pi, l)\). 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. 

\[\square\]
### Table 1. Tabular summary of the results provided in this paper.

<table>
<thead>
<tr>
<th></th>
<th>Modal specifications</th>
<th>Mixed specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common impl.</td>
<td>EXPTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>Consistency</td>
<td>trivial</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>Thorough ref.</td>
<td>EXPTIME</td>
<td>EXPTIME-complete</td>
</tr>
</tbody>
</table>

**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 $\text{GMC}(M, s, \varphi)$ that are true if and only if there exists an implementation of $(M, s)$ satisfying $\varphi$. 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)), $\text{VAL}(M, s, \varphi)$, which is true if and only if all implementations of $(M, s)$ satisfy $\varphi$, 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 $\text{VAL}(N, t, \Psi_{(M, s)})$, it would be interesting to understand the exact complexity of $\text{VAL}(N, t, \varphi)$ for modal specifications $(N, t)$ when $\varphi$ 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 $\mathcal{M}_w^T$ of constructed modal specifications will have a common implementation; and, conversely, that if this set $\mathcal{M}_w^T$ 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 $\mathcal{M}_w^T$ 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 $T_{(T,w)}$ demonstrating that $T$ accepts $w$ in an exponentially bounded number of steps.

We will use $T_{(T,w)}$ to construct a modal specification

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

over $\Sigma$, where $N_{\text{states}}$ is a set of states, $R_N$ is a transition relation and $\Sigma$ is the alphabet of specifications in $\mathcal{M}_w^T$. The proof that $N$ is indeed an implementation of all specifications in $\mathcal{M}_w^T$ 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 $T_{(T,w)}$. More precisely, we distinguish three kinds of states:

— Type 1 states, indexed by a configuration of $T_{(T,w)}$ only, for example state $n_{(q_0,1,w)}$.
— Type 2 states, indexed by a configuration and an extra subscript $\exists$, as in $n_{(q_d,i,\exists)}$.
— Type 3 states, indexed by a configuration and an extra subscript $\pi$, as in $n_{(q_d,i,\pi)}$.

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

— If $\text{mode}(q) = \text{Univ}$, create two new states $n_{(q',i,\exists)}$ and $n_{(q',i,\pi)}$ and an $\exists$-transition from $n_{(q,i,\tau)}$ to $n_{(q',i,\exists)}$ and a $\pi$-transition from $n_{(q,i,\tau)}$ to $n_{(q',i,\pi)}$. Then, for each of the successors $\langle q',i',\tau' \rangle$ of $\langle q,i,\tau \rangle$, create a new state $n_{(q',i',\exists)}$ and a transition from $n_{(q,i,\tau)}$ to $n_{(q',i',\exists)}$ labelled by $(\tau_i,i',\tau',d)$ where $d = r$ if $i' = i + 1$ and $d = l$ otherwise. Then continue recursively for every successor $\langle q',i',\tau' \rangle$ of $\langle q,i,\tau \rangle$, and its corresponding state $n_{(q',i',\tau')}$. See Figure 14(a).

— If $\text{mode}(q) = \text{Ext}$, create two new states $n_{(q,i,\exists)}$ and $n_{(q,i,\pi)}$ and an $\exists$-transition from $n_{(q,i,\tau)}$ to $n_{(q,i,\exists)}$ and a $\pi$-transition from $n_{(q,i,\tau)}$ to $n_{(q,i,\pi)}$. Then, because $T_{(T,w)}$ 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 $n_{(q',i',\exists)}$ and a transition from $n_{(q,i,\tau)}$ to $n_{(q',i',\exists)}$ labelled by $(\tau_i,i',\tau',d)$ where $d = r$ if $i' = i + 1$ and $d = l$ otherwise. Then continue recursively with $\langle q',i',\tau' \rangle$ and $n_{(q',i',\tau')}$. 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 $T_{(T,w)}$ 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 $(\tilde{N},n_{(q_0,1,w)})$ refines each of the modal specifications in $\mathcal{M}^T_{w}$.

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

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

$$Q_1 = \{(x_1,n_{(q_d,i,\tau)}), n_{(q_d,i,\tau)} \in N_{\text{states}}\} \cup \{(x_2,n_{(q_d,i,\exists)}), n_{(q_d,i,\exists)} \in N_{\text{states}}\} \cup \{(x_3,n_{(q_d,i,\pi)}), n_{(q_d,i,\pi)} \in N_{\text{states}}\}.$$

$\dagger$ We write $\tau_i$ to mean the $i$th symbol of the tape state $\tau$. 

http://journals.cambridge.org Downloaded: 06 Jun 2014 IP address: 155.198.12.107
(a) Construction for a universal state \( q \). For \( l = 1..k \) we define \( d_l = r \) if \( i_l = i + 1 \) and \( d_l = 1 \) otherwise.

\[ \begin{array}{c}
\langle q, i, \tau \rangle \\
\langle q_1, i_1, \tau^1 \rangle \ldots \langle q_k, i_k, \tau^k \rangle
d\end{array} \begin{array}{c}
\sim \Rightarrow
\langle q, i, \tau \rangle \\
\langle q_1, i_1, \tau^1 \rangle \ldots \langle q_k, i_k, \tau^k \rangle
\end{array} \begin{array}{c}
n\langle q, i, \tau \rangle \\
\langle q, i, \tau \rangle \exists
\end{array} \begin{array}{c}
n\langle q, i, \tau \rangle \pi \\
n\langle q, i, \tau \rangle \pi
\end{array} \begin{array}{c}
\langle \tau_1, i, (\tau^1)_i, d_1 \rangle \\
\langle \tau, i, (\tau^i)_i, d_i \rangle
\end{array}
\]

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

\[ \begin{array}{c}
\langle q, i, \tau \rangle \\
\langle q_1, i_1, \tau^1 \rangle \ldots \langle q_k, i_k, \tau^k \rangle
d\end{array} \begin{array}{c}
\sim \Rightarrow
\langle q, i, \tau \rangle \\
\langle q_1, i_1, \tau^1 \rangle \ldots \langle q_k, i_k, \tau^k \rangle
\end{array} \begin{array}{c}
n\langle q, i, \tau \rangle \\
\langle q, i, \tau \rangle \exists
\end{array} \begin{array}{c}
n\langle q, i, \tau \rangle \pi \\
n\langle q, i, \tau \rangle \pi
\end{array} \begin{array}{c}
\langle \tau_1, i, (\tau^1)_i, d_1 \rangle \\
\langle \tau, i, (\tau^i)_i, d_i \rangle
\end{array}
\]

Fig. 14. Construction of a common implementation \( N \) from fragments of the accepting computation tree \( T_{\langle T, w \rangle} \).

We will show that \( Q_1 \) witnesses a refinement of \( (M_{\text{exist}}, x_1) \) 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 \( M_{\text{exist}} \) and \( N \) are related in \( Q_1 \). Then we check that \( Q_1 \) fulfills the conditions of Definition 2:

1. We need to show for all pairs \( (x, n) \in Q_1 \) that for all states \( x' \) of \( M_{\text{exist}} \), if \( (x, a, x') \in R^\boxplus_{M_{\text{exist}}} \) there exists a state \( n' \in N_{\text{states}} \) with \( (n, a, n') \in R^\boxplus_N \) and \( (x', n') \in Q_1 \). A must-transition occurs in \( R^\boxplus_{M_{\text{exist}}} \) only if \( x = x_2 \). In this case there is exactly one must \( \pi \)-transition going to \( x_3 \). We see from \( Q_1 \) that \( x_2 \) is paired only with states of form \( n = n_\langle q_0, i, \tau \rangle \exists \). By construction of \( N \), the latter state always has a must \( \pi \)-transition to some state \( n' = n_\langle q_0, i, \tau \rangle \pi \) 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 \( (n, a, n') \in R^\boxplus_N \), there exists a state \( x' \) of \( M_{\text{exist}} \) such that \( (x, a, x') \in R^\boxplus_{M_{\text{exist}}} \) with \( (x', n') \in Q_1 \). We consider three sub-cases:

   — \( n \) is of type 1, so \( n = n_\langle q, i, \tau \rangle \exists \):

   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 \( n' = n_\langle q_0, i, \tau \rangle \exists \).
This can be matched by \((x_1, 3, x_2) \in R_{M_{\text{exist}}}^\circ\) and, for \(x' = x_2\), we get \((x', n') \in Q_1\) by construction of \(Q_1\).

— \(n\) is of type 2, so \(n = n_{(q_i, i, t)3}\):

By \(Q_1\)’s construction, we have \(x = x_2\). By the construction of \(N\), there is exactly one may \(\pi\)-transition leaving \(n\). It targets a state \(n'\) of type 3, so \(n' = n_{(q_i, i, t)\pi}\). This can be matched by \((x_2, \pi, x_3) \in R_{M_{\text{exist}}}^\circ\), so taking \(x' = x_3\), we get \((x', n') \in Q_1\) by the construction of \(Q_1\).

— \(n\) is of type 3, so \(n = n_{(q_i, i, t)\pi}\):

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 \(n' = n_{(q_i, i, t)}\). All these transitions have labels in \((\_, \_, \_, \_\_)\). These can all be matched by \((M_{\text{exist}}, x_3)\), 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. \((M_i, p_{(i,w)}) < (N, n_{(q_0, 1, w)})\) for each tape cell \(1 \leq i \leq n\):

For any selection of \(i\) above, consider the following relation \(Q_2^i\) over the states of \(M_i\) and the states of \(N\):

\[
Q_2^i = \{(p_{(i,r)}, n) \mid n = n_{(q_i, i, t)} \text{ or } n = n_{(q_i, i, t)\pi} \text{ or } n = n_{(q_i, i, t)3}, \text{ for } 1 \leq j \leq n\}
\]

First note that the initial states of the two specifications are related in \(Q_2^i\). This is clearly the case since the initial state of each \(M_i\) is \(p_{(i,w)}\), so by the definition of \(Q_2^i\) it is related to \(n_{(q_0, 1, w)}\). We still need to show, given \((p, n) \in Q_2^i\), 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 \((p, n) \in Q_2^i\) that for all states \(n' \in N_{\text{states}}\) if \((n, a, n') \in R_{N}^\circ\), there exists a state \(p'\) of \(M_i\) such that \((p, a, p') \in R_{M_i}^\circ\) with \((p', n') \in Q_2^i\).

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 \((p', n') \in Q_2^i\).

3. \((M_{\text{head}}, p_1) < (N, n_{(q_0, 1, w)})\):

The relation \(Q_3\) witnessing this refinement is defined as follows:

\[
Q_3 = \{(p_i, n) \mid n = n_{(q_i, i, t)} \text{ or } n = n_{(q_i, i, t)\pi} \text{ or } n = n_{(q_i, i, t)3}\}
\]

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 \(M_{\text{head}}\). We now need to show that for any given \((p, n) \in Q_3\), the two refinement conditions of Definition 2 are preserved:
Consider the following binary relation \( Q \) on states of \( M_{\text{ctrl}} \) and \( N \):

\[
Q_4 = \{(q, n) \mid n = n_{(q,t)}\} \cup \\
\{(q, n) \mid n = n_{(q,t)}\} \cup \\
\{(q, i, \tau, n) \mid \text{mode}(q) = \text{Univ}\} \cup \\
\{(q, i, a, d, n) \mid \text{mode}(q) = \text{Ext} \text{ and} \\
n_{(q,t)}(\pi, i, a, d, n_{(q,t)}) \in R_N^\circ \}
\]

First observe that the initial states of the two specifications are in \( Q_4 \) since \( q_0 \) is the initial state of \( M_{\text{ctrl}} \) and \( n_{(q_0,1,w)} \) 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 \((q,a,n) \in R_{M_{\text{ctrl}}}^\circ\), there exists a state \( n' \in N_{\text{states}} \) such that \((n,a,n') \in R_N^\circ \) 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 \( q_{s3} \). This transition can be matched by an \( \exists \)-transition leaving \( n_{(q,t)} \) and targeting \( n_{(q,t)}\). These new target states remain in relation \( Q_4 \), as in the above definition.

- \( q = q_{s3} \) 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 \( \langle q_s\tau_i \rangle \), where \( q_s \) is a universal state of the ATM \( T \):

\( n \) has the form \( n_{(q,t)}\), but since \( n_{(q,t)} \) was constructed by our recursive procedure from a universal configuration of an accepting computation tree, we know that for all must-transitions leaving \( \langle q_s\tau_i \rangle \) to some state \( q_i \), there will be a matching must-transition in \( N \) leaving \( n_{(q,t)} \) and targeting \( n_{(q,t')} \), which is in relation with \( q_i \) as in the first summand in the definition of \( Q_4 \).
A. Antonik, M. Huth, K. Larsen, U. Nyman and A. Wąsowski

This concludes the argument that each specification in $M$:

(2) We need to show that if $(n, a, n') \in R_N^\uparrow$, there exists a state $q'$ of $M_{\text{ctrl}}$ such that $(q, a, q') \in R_{M_{\text{ctrl}}}^\uparrow$ with $(q', n') \in Q_4$.

We consider three cases according to the type of state $n$:

— $n$ is of type 1, so $n = n_{(q_i, i, \tau)}$:

By the construction of $N$, there is a may $\exists$-transition leaving $n$ targeting $n_{(q_i, i, \tau, \exists)}$. This is followed by $(q_i, \exists, q_{i, \exists}) \in R_{M_{\text{ctrl}}}^\uparrow$ and again gives us that $(q', n') \in Q_4$.

— $n$ is of type 2, so $n = n_{(q_i, i, \tau, \exists)}$:

By the construction of $Q_4$ (see the second summand), $q$ is of the form $q_{i, \exists}$. By the construction procedure for $N$, there is a single may $\pi$-transition leaving $n_{(q_i, i, \tau, \exists)}$ and targeting $n' = n_{(q_i, i, \tau, \pi)}$.

- If $\text{mode}(q_i) = \text{Univ}$, there is exactly one transition $(q_{i, \exists}, \pi, (q_i, \tau, i)) \in R_{M_{\text{ctrl}}}^\uparrow$ and its target state is related to $n_{(q_i, i, \tau, \pi)}$ in $Q_4$.

- If $\text{mode}(q_i) = \text{Exst}$, there can be many may $\pi$-transitions leaving $q_{i, \exists}$. We will choose which one to match with, based on the label of the single transition leaving $n_{(q_i, i, \tau, \pi)}$. We are, so to speak, looking one step ahead. Since $n_{(q_i, i, \tau, \pi)}$ says that the head is in position $i$ over a tape containing $\tau$, we choose to match our transition with the transition of $M_{\text{ctrl}}$ targeting the state whose name matches the prefix $'q_i, \tau, i'$'. Such a state always exists by construction of $M_{\text{ctrl}}$, and it is exactly the state that is related to $n_{(q_i, i, \tau, \pi)}$ in $Q_4$ (see the last summand).

— $n$ is of type 3, so $n = n_{(q_i, i, \tau, \pi)}$:

We consider two cases according to the mode of $q_i$ in the ATM $T$:

- $\text{mode}(q_i) = \text{Univ}$:

There may be several may-transitions leaving $n_{(q_i, i, \tau, \pi)}$. Since $N$ has been created from a legal computation tree, we know that any may-transition leaving $n_{(q_i, i, \tau, \pi)}$ and targeting $n' = n_{(q_i, i, \tau', r)}$ follows the transition relation $\delta$ of $T$. Moreover, by the construction of $M_{\text{ctrl}}$, its state $(q_i, \tau, i)$ will consequently be able to match this transition arriving in the state $q_i$ related to $n'$ in $Q_4$.

- $\text{mode}(q_i) = \text{Exst}$:

There is exactly one may-transition leaving $n_{(q_i, i, \tau, \pi)}$ and exactly one may-transition leaving $(q_i, \tau, i, a_2, dq_i)$. These transitions have the same label and have target states $n_{(q_i, i, \tau', r)}$ and $q_i$, respectively, which are related in $Q_4$.

This concludes the argument that each specification in $\mathcal{M}_w^T$ is refined by $N$. 
A.2. The existence of a common implementation implies acceptance

Let \( M_w^T \) 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 \( M_w^T \), we will construct a computation tree \( T_{(M,w)} \) demonstrating that \( T \) accepts \( w \).

Since \( U_{\text{new}} \) is a common implementation of \( M_w^T \), 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 \( S \in M_w^T \) that \( S \prec U_{\text{new}} \).

The construction of \( T_{(M,w)} \) 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 \( T_{(M,w)} \) 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) \( (u_x, p_{(k,w_k)}) \in Q_k \) 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 \( M_{\text{ctrl}} \). Specifically, we mean that \( q \) represents a label without any special suffixes. Label \( p_i \) refers to a particular state of \( M_{\text{head}} \), the one representing position \( i \). Similarly, \( p_{(k,w_k)} \) denotes the state of \( M_k \) that represents the fact that the \( k \)th symbol of \( \tau \) is stored in the \( k \)th 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 \( T_{(M,w)} \), then it is also a successor of \( \langle q, i, \tau \rangle \) in the ATM \( T \);
   - and, conversely:
   - (IH6) the tree \( T_{(M,w)} \) 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 \( T_{(M,w)} \) actually is an accepting computation tree of \( T \), witnessing acceptance of \( w \), after discussing the construction of \( T_{(M,w)} \), and after arguing that it satisfies the above inductive property.

Root (base case):

The root of \( T_{(M,w)} \) 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 \( U_{\text{new}} \) 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 \( M_{\text{head}} \), we know that \( (u_0, p_1) \in Q_{\text{head}} \).
- (IH4) Since \( w \) is the initial content of the tape, and thus \( p_{(k,w_k)} \) is an initial state of \( M_k \), the refinement \( M_k \prec U_{\text{new}} \) gives us that \( (u_0, p_{(k,w_k)}) \in Q_k \), so IH4 holds for \( \langle q_0, 1, w \rangle \).
(IH1) Since $M_{exist} \prec_{U_{new}} u_0$, we get that $(u_0, x_1) \in Q_{exist}$.

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 $T_{\langle M, w \rangle}$ 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 $\pi$ 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 $(u_x, x_1) \in Q_{exist}$, we know that $(u_x^\exists, x_2) \in Q_{exist}$, so $u_x^\exists$ must implement a $\pi$ transition to a new state, say $u_{x\pi}$. Since all $\pi$ 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 $q$:

---

**mode(q) = Ext:**

We know that $(M_{ctrl}, q)$ has to implement an $\exists$-transition followed by at least one $\pi$-transition reaching a state of the form $\langle q_\exists, i, a'dq' \rangle$. Also, because $u_x^\exists$ is related to $q_\exists$, it must be possible to choose $u_{x\pi}$ above such that $(u_{x\pi}, \langle q_\exists, i, a'dq' \rangle) \in Q_{ctrl}$, but then we know that $u_{x\pi}$ can take a transition labelled $(\tau_i, i, a', d)$ to some state $u'_x$ related to $q'$ in $Q_{ctrl}$.

So, if we extend $T_{\langle M, w \rangle}$ 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 $d = r$, 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 $(\tau_i, i, a', d)$, and then shown to witness fullfillment of the condition for $u'_x$ by the induction hypothesis (from refinement of $u_x$).

---

**mode(q) = 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 $(M_{ctrl}, \langle q_\pi, i \rangle)$.

The refinement relation with $M_{head}$ and $M_1$ 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 $\delta$ of $T$.

So we can extend $T_{\langle M, w \rangle}$ with new children $\langle q', i', \tau'[\tau_i \mapsto a'] \rangle$ for all $(q', i', \tau')$ such that $(M_{ctrl}, q')$ can be reached from $(M_{ctrl}, \langle q_\pi, i \rangle)$ in one step with a transition labelled $(\tau', i, a', d)$. Also $i' = i + 1$ if $d = r$, 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 $M_{\text{ctrl}}$, 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 $d = r$, and $l = i - 1$ if $d = l$. 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 $M_{\text{exist}}$, 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 $T_{(M, w)}$ 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, $T_{(M, w)}$ 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


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

Modal and mixed specifications: key decision problems and their complexities