EXPTIME-complete Decision Problems for Modal and Mixed Specifications[1]

Adam Antonik and Michael Huth[2,3]

Department of Computing, Imperial College London, United Kingdom

Kim G. Larsen and Ulrik Nyman[4,5]

Department of Computer Science, Aalborg University, Denmark

Andrzej Wąsowski[6]

IT University of Copenhagen, Denmark

Abstract

Modal and mixed transition systems are formalisms that allow mixing of over- and under-approximation in a single specification. We show EXPTIME-completeness of three fundamental decision problems for such specifications: whether a set of modal or mixed specifications has a common implementation, whether a sole mixed specification has an implementation, and whether all implementations of one mixed specification are implementations of another mixed or modal one. These results are obtained by a chain of reductions starting with the acceptance problem for linearly bounded alternating Turing machines.

1 Introduction

Behavioral models capture actual, desired or required system behavior and can so serve as documentation, specification or as the basis of analysis and validation activities. Formal behavioral models — of which we mention process algebras, Petri nets and labelled transition systems — bring a high degree of rigor and dependability to validation and verification activities.

[1] Huth and Antonik were 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*. Wąsowski was partially funded by CISS—Center for Embedded Software Systems, Aalborg University

[2] aa1001@doc.imperial.ac.uk

[3] mrh@doc.imperial.ac.uk

[4] kgl@cs.aau.dk

[5] ulrik@cs.aau.dk

[6] wasowski@itu.dk

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Often one has to deal with more than one behavioral model at a time. For example, in requirement elaboration one may have several versions of a model, in component-based design one may have models that each focus on a different aspect of the system, and in formal verification one may have a system model accompanied by models that represent either desired features or genuinely faulty behavior. In each of these cases the modeller may want to have assurance that this collection of models is consistent. If versions of models are inconsistent with each other, this may reveal important implementation trade-offs. If all aspect models are inconsistent, their combination is not implementable. If a system model is inconsistent with all members of a given set of fault models, the system will not exhibit any of these flaws. Finally if a system model is consistent with a set of feature models, then the system will be able to actually implement all these features.

A related concept is the consistency of a *single* behavioral model. If models serve as specifications, their inconsistency suggests that the specification cannot be implemented. Conversely, a consistent model boosts our confidence in implementability and may even allow code-generation of such an implementation.

The stepwise-refinement paradigm proposes to write specifications as models and to then repeatedly refine such models until an implementation has been realized. In a *thorough* interpretation, refinement is decreasing the set of possible implementations: only implementations that were possible before the refinement step are still possible thereafter, but not necessarily all of them anymore.

This paper is devoted to studying the exact computational complexity of these three decision problems; whether finitely many models are consistent, whether a single model is consistent, and whether one model thoroughly refines another. The actual models we study are *mixed specifications* — stateful models with allowed and required transitions, well recognized as a formal foundation for system specification and abstraction alike [23,18,24,5,21,22,8,9,20,19]. We show that

- deciding whether finitely many modal or mixed specifications are consistent is EXPTIME-complete in the sum of the sizes of these specifications
- deciding whether one mixed specification is consistent is EXPTIME-complete in the size of that specification
- deciding whether one mixed specification thoroughly refines another mixed specification is EXPTIME-complete in the sum of their sizes.

Interestingly, checking the consistency of 100 mixed specifications with a few states each can be dramatically more complex than checking the consistency of a few mixed specifications with 100 states each. This is in striking contrast to the situation when all mixed specifications are fully refined (have identical required and allowed behaviors). In that case, consistency checks reduce to pairwise bisimilarity checks, which can be performed in polynomial time.

Our complexity results motivate future research that aims to either approximate these three decision problems soundly and efficiently, or that identifies sub-classes of specifications for which these decision problems are less complex.

We proceed by introducing the necessary background on alternating Turing machines, specifications, and their decision problems in Section 2. In Section 3 state-of-the-art bounds for these problems are reported. The new EXPTIME-completeness

results are given in Section 4. Section 5 reflects on a remaining open complexity gap for a special kind of mixed specifications, modal ones. We conclude in Section 6.

Related work

We refer to our recent overview [2] for a full account of related work. The present paper primarily improves on the results of [3], which are discussed in detail in Section 3. The relation of this work to generalized model checking [4] is detailed in Section 5.

In [13] a superpolynomial algorithm is given, which establishes common implementation for $k > 1$ modal specifications. The algorithm is exponential in k, but polynomial if k is fixed. It computes a common implementation if one exists. These upper bounds follow also from the polynomial algorithm for consistency checking of a conjunction of disjunctive modal transition systems, as studied in [24].

In [14] Hussain and Huth present an example of two modal specifications that have a common implementation but no greatest common implementation.

Fischbein et al. [10] use modal specifications for behavioral conformance checking of products against specifications of product families. They propose a new thorough refinement whose implementations are defined through a generalization of branching bisimulation. The thorough refinement obtained in this manner is finer than weak refinement, and argued to be more suitable for conformance checking.

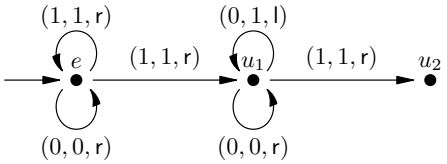
2 Background

Let us begin with a definition of the decision problem used in the main proof of this paper. An *Alternating Turing Machine* [6], 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 tape symbols, $\mathsf{null} \notin \Gamma$ is a special symbol denoting empty cell contents, $\delta\colon Q \times (\Gamma \cup \{\mathsf{null}\}) \to \mathcal{P}(Q \times \Gamma \times \{\mathsf{l}, \mathsf{r}\})$ is a transition relation, $q_0 \in Q$ is the initial control state, and $\mathsf{mode}\colon Q \to \{\mathsf{Univ}, \mathsf{Exst}\}$ is a labeling of control states as respectively universal or existential. 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 a single cell at a time, which can then be read or written to. The head can then move to the left or right: $(q', a', \mathsf{r}) \in \delta(q, a)$, e.g., 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 $\Gamma \cup \{\mathsf{null}\}$.

Figure 1 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 u_i control states are universal.

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. For input $w \in \Gamma^*$, the initial configuration is $\langle q_0, 1, w\mathsf{null}^\omega \rangle$. The recursive and parallel execution of all applicable [7] transitions δ from initial

[7] Transitions $(\_, \_, \_, \mathsf{l})$ are not applicable in configurations $\langle \_, 1, \_ \rangle$ as the head cannot move over the left



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

Fig. 1. The transition relation of an ATM as a labelled graph and a function.

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 iff the tree $\mathsf{T}_{\langle T,w\rangle}$ accepts, where the latter is a recursive definition:

- $\mathsf{T}_{\langle T,w\rangle}$ with root $\langle q, i, \tau\rangle$ and $\mathsf{mode}(q) = \mathsf{Exst}$ accepts iff 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
- $\mathsf{T}_{\langle T,w\rangle}$ with root $\langle q, i, \tau\rangle$ and $\mathsf{mode}(q) = \mathsf{Univ}$ accepts iff 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)

The ATM of Figure 1 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 last 1 is ignored. Note that the computation forks in u_1 whenever a 0 is seen. However, the top branch would reach the earlier 1 eventually and accept.

An ATM T is *linearly bounded* iff 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 by 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 $|v| \leq |w|$. We refer to such pruned computation trees simply as "computations".

Our notion of "linear boundedness" follows [17] 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. [8]

Let $\mathrm{ATM}_{\mathrm{LB}} = \{\langle T, w\rangle \mid w \in \Gamma^*$ accepted by linearly bounded ATM $T\}$. The problem of deciding if 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 [6].

Let us now define the basic models of interest in our study [18,8,7]:

Definition 2.1 For a finite alphabet of actions Σ, a *mixed specification* M is a triple

boundary of the tape.

[8] This is possible because ASPACE = EXPTIME [27, Thm. 10.18].

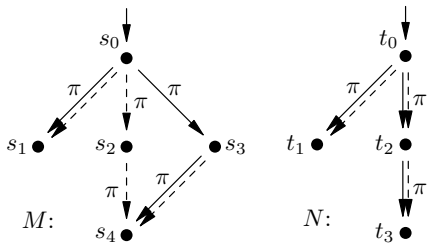


Fig. 2. Mixed $((M, s_0))$ and modal $((N, t_0))$ specifications with $I(M, s_0) = I(N, t_0)$ but not $(N, t_0) \prec (M, s_0)$.

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

Refinement [18,8,7], called "modal refinement" in [20], is a co-inductive relationship between two mixed specifications that verifies that one such specification is more abstract than the other. This generalizes the co-inductive notion of bisimulation [25] to mixed specifications:

Definition 2.2 A mixed specification $(N, t_0) = ((S_N, R^{\Box}_N, R^{\Diamond}_N), t_0)$ *refines* another mixed specification $(M, s_0) = ((S_M, R^{\Box}_M, R^{\Diamond}_M), s_0)$ over the same alphabet Σ, written $(M, s_0) \prec (N, t_0)$, iff there is a relation $Q \subseteq S_M \times S_N$ containing (s_0, t_0) and whenever $(s, t) \in Q$ then

(i) for all $(s, a, s') \in R^{\Box}_M$ there exists some $(t, a, t') \in R^{\Box}_N$ with $(s', t') \in Q$
(ii) 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 one finite-state mixed specification refines another one is in P. For mixed specification (N, t_0) and modal specification (M, s_0) in Figure 2 we have $(M, s_0) \prec (N, t_0)$, given by $Q = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_2), (s_4, t_3)\}$. Note that throughout figures, solid arrows denote $R^{\Box}$-transitions, and dashed arrows denote $R^{\Diamond}$-transitions. 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 2.2. Then from $(s_0, \pi, s_2) \in R^{\Diamond}_M$ we infer that there must be some x with $(t_0, \pi, x) \in R^{\Diamond}_N$ and $(x, s_2) \in Q$. In particular, x can only be t_1 or t_2. If x is t_1, then since $(s_2, \pi, s_4) \in R^{\Diamond}_M$ and $(t_1, s_2) \in Q$ there has to be some $R^{\Diamond}_N$ transition out of t_1, which is not the case. If x is t_2, then $(t_2, \pi, t_3) \in R^{\Box}_N$ and $(t_2, s_2) \in Q$ imply that there is some $R^{\Box}_M$ transition out of s_2, which is not the case. In conclusion, there cannot be such a Q and so $(N, t_0) \not\prec (M, s_0)$.

Labeled 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 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 2.3 Let (N,t) and (M,s) be pointed mixed specifications. As in [20] 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 [16,28,26]; Figure 2 provides a counterexample.

We shall now formally define the decision problems informally stated above:

Common implementation (CI): given $k > 1$ modal or mixed specifications (M_i, s_i), is the set $\bigcap_{i=1}^{k} I(M_i, s_i)$ non-empty?

Consistency (C): Is $I(M,s)$ non-empty for a modal or mixed specification (M,s)?

Thorough refinement (TR): Does a mixed specification (N,t) thoroughly refine a mixed specification (M,s), i.e., 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, causes no loss of generality, as already explained in [3]. A mixed specification (M,s) is consistent in the infinite sense iff its characteristic modal mu-calculus formula $\Psi_{(M,s)}$ [15] is satisfiable. Appealing to the small model theorem for mu-calculus, $\Psi_{(M,s)}$ is satisfiable iff it is satisfiable over finite-state implementations. We can reason in a similar manner about common implementation, which justifies the restriction to finite-state specifications and implementations.

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

3 Current Bounds

In [3], the three decision problems CI, C, and TR were studied for mixed and modal specifications. The results of [3] are summarized in Table 1. Two reductions were given in [3] that we appeal to here:

- a reduction of CI for modal specifications to C for *mixed* specifications
- a reduction of C for mixed specifications to TR for *mixed* specifications.

EXPTIME-hardness of CI for modal specifications would thus render EXPTIME-completeness of the decision problems CI, C, and TR for mixed specifications. We

Table 1
A summary given in [3] of the results provided in [3].

	Modal specifications	Mixed specifications
Common impl.	PSPACE-hard, EXPTIME	PSPACE-hard, EXPTIME
Consistency	trivial	PSPACE-hard, EXPTIME
Thorough ref.	PSPACE-hard, EXPTIME	PSPACE-hard, EXPTIME

turn to this EXPTIME-hardness proof in the next section.

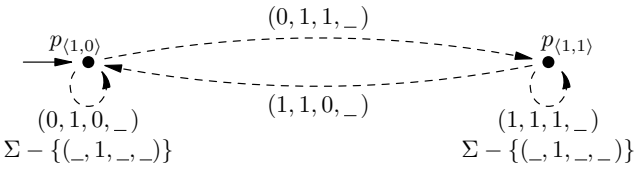


Fig. 3. Specification M_1 of the first tape cell in our running example, assuming $w_1 = 0$.

4 EXPTIME-Completeness Results

Theorem 4.1 *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 4.1 by demonstrating a PTIME reduction from $\mathrm{ATM}_{\mathrm{LB}}$. Given an ATM T and an input word w of length n we synthesize a collection of (pointed) modal specifications $\mathcal{M}_w^T = \{M_i \mid 1 \leq i \leq n\} \cup \{M_{\mathrm{head}}, M_{\mathrm{ctrl}}, M_{\mathrm{exist}}\}$ whose sum of sizes is polynomial in n and in the size of T, such that T accepts w iff there exists an (pointed) implementation I refining all members of $\mathcal{M}_w^T$.

Specifications M_i, M_{head}, M_{ctrl}, and M_{exist} model tape cell i, the current head position, the finite control of T, and acceptance (respectively). Common implementations of these specifications model action synchronization to agree on what symbol is read from the tape, what is the head position, what is the symbol written to the tape, in what direction the head moves, and what are the transitions taken by the finite control, and whether a computation is accepting. The achieved effect is that a common refinement of these specifications corresponds to an accepting computation of T on input w. More precisely, any common implementations 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 $\mathcal{M}_w^T$ both formally and through our running example in Figure 1. All specifications in $\mathcal{M}_w^T$ have the same alphabet [9]

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

where $\exists$ and π are fresh symbols whose transitions encode logical constraints like disjunction and conjunction. All other actions are of the form (a_1, i, a_2, d) and denote 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. The alphabet for our running example is

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

Encoding Tape Cells.

For each tape cell i, 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

[9] A stricter and more complex reduction to CI of modal specifications over a *binary* alphabet is possible by encoding actions in binary form.

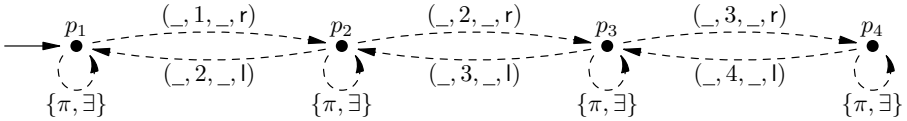


Fig. 4. Example of the head specification M_{head} assuming $|w| = 4$.

of the ith cell. There are no must-transitions:

$$R^{\Box} = \emptyset$$

The may-transition relation connects any two states:

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

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

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

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 more may-transitions in M_i.

Figure 3 presents a specification M_1 for the leftmost cell of an ATM over a binary alphabet. In figures we visualize multiple transitions with the same source and target as single arrows labeled with sets of actions. Several labels placed by 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.

Encoding The Head.

Specification M_{head}, which tracks the current head position, has n states labeled 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:

$$R^{\Box} = \emptyset$$

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

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

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

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

$$(p_i, \pi, p_i) \in R^{\diamond} \text{ and } (p_i, \exists, p_i) \in R^{\diamond} \text{ for each } 1 \leq i \leq n$$

There are no more 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 4 shows specification M_{head} for our running example.

Encoding The Finite Control.

Specifications M_{ctrl} and M_{exist} model the finite control of the ATM T. Specification M_{exist} is indepenendent of the ATM T. It is defined in Figure 5. It ensures that a π-transition is taken after every $\exists$-transition. 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

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

- If $\mathsf{mode}(q_s) = \mathsf{Exst}$: for each $1 \leq i \leq n$, $a_{\text{old}} \in \Gamma$, and for each transition $(q_t, a_{\text{new}}, d) \in \delta(q_s, a_{\text{old}})$ add a may π-transition from $q_{s\exists}$ to a new intermediate state uniquely named $\langle q_s a_{\text{old}} i a_{\text{new}} d q_t \rangle$, and add a must-transition labeled $(a_{\text{old}}, i, a_{\text{new}}, d)$ 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 6 shows this encoding for the state e of our running example.

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

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

The initial state of M_{ctrl} is its state named q_0, where q_0 is the initial state of T. Figure 7 demonstrates the encoding of the state u_1 of the ATM in Figure 1. The complete specification M_{ctrl} for our running example is shown in Figure 8.

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 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).

The complete family of specifications $\mathcal{M}_w^T$ contains all the specifications described above:

$$\mathcal{M}_w^T = \{M_i \mid 1 \leq i \leq n\} \cup \{M_{\text{head}}, M_{\text{ctrl}}, M_{\text{exist}}\}$$

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, it remains to prove the following lemma:

Lemma 4.2 *For each linearly bounded ATM T and an input w, T accepts w iff the set of modal specifications $\mathcal{M}_w^T$ has a common implementation.*

The proof of Lemma 4.2 will appear in the final version of the paper. We 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 labeled transitions so that the computation encoded in $\mathsf{T}_{\langle T,w\rangle}$ then interlocks with the action synchronization of specifications in $\mathcal{M}_w^T$. Since N is of the form (S, R, R) it suffices to show that N is a common refinement of all members in $\mathcal{M}_w^T$. This is a lengthy but routine argument.

For the converse, a common implementation of $\mathcal{M}_w^T$ is cycle-free by our assumption that T never repeats a configuration. So that pointed common implementation is a DAG and we use structural induction on that DAG to synthesize 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 initialized by input w.

Further results.

Theorem 4.1 states EXPTIME-hardness of CI for *modal* specifications. Together with the upperbound given in [3] we conclude that this bound is tight: CI is EXPTIME-complete. Moreover, by applying the reduction of CI for modal specifications to C for mixed specifications [3] we conclude that C for mixed specifications is EXPTIME-complete. Furthermore by appealing to the reduction of C for mixed specifications to TR for mixed specifications [3], we obtain that TR for mixed specifications is EXPTIME-complete as well.

Corollary 4.3 *The complexities shown in Table 2 are correct.*

5 Discussion

One complexity gap remains in Table 2, that for thorough refinement of *modal* specifications. Despite having made an extensive effort we can presently show neither EXPTIME-hardness nor membership in PSPACE for this problem.

In this context, it is useful to state that thorough refinement can be reduced to certain validity checks. First, as observed in [3], mixed and modal specifications (M, s) have characteristic formulæ $\Psi_{(M,s)}$ [15] in the modal μ-calculus such that pointed labeled transition systems (L, l) are implementations of (M, s) iff (L, l) satisfies $\Psi_{(M,s)}$. This was already observed in [18] for such formulæ written in vectorized form. So the thorough refinement problem of whether $(M, s) \prec_{th} (N, t)$ reduces to a validity check of $\neg\Psi_{(N,t)} \vee \Psi_{(M,s)}$. This raises the question of whether the validity problem for formulae given in the vectorized form of [18] is in PSPACE or whether it is EXPTIME-hard; that problem is known to be in EXPTIME (see

Table 2
Tabular summary of the results provided in this paper (in bold).

	Modal specifications	Mixed specifications
Common impl.	**EXPTIME-complete**	**EXPTIME-complete**
Consistency	trivial [23]	**EXPTIME-complete**
Thorough ref.	PSPACE-hard, EXPTIME [3]	**EXPTIME-complete**

for example [3]).

Second, we can reduce thorough refinement to a universal version of generalized model checking [4]. In loc. cit. Bruns and Godefroid consider judgments $\mathsf{GMC}(M, s, \varphi)$ which are true iff there exists an implementation of (M, s) satisfying φ. They remark that this generalizes both model checking (when (M, s) is an implementation) and satisfiability checking (when (M, s) is such that all labeled transition systems refine it). This existential judgment has a universal dual (see e.g. [1]), $\mathsf{VAL}(M, s, \varphi)$ which is true iff all implementations of (M, s) satisfy φ, thus generalizing model checking and validity checking. The former judgment is useful for finding counter-examples, the latter one for verification; e.g. both uses can be seen in the CEGAR technique for program verification of [11]. Since $(M, s) \prec_{th} (N, t)$ directly reduces to $\mathsf{VAL}(N, t, \Psi_{(M,s)})$, it would be of interest to understand the exact complexity of $\mathsf{VAL}(N, t, \varphi)$ for modal specifications (N, t) when φ ranges over characteristic formulæ $\Psi_{(M,s)}$ in vectorized form.

We remark that by translations and completeness results presented in [12] it follows that all complexity bounds presented here carry over to partial Kripke structures and Kripke modal transition systems.

6 Conclusion

We have discussed three fundamental decision problems for modal and mixed specifications: common implementation, consistency, and thorough refinement. For *modal* specifications, consistency is trivially true, while thorough refinement was previously shown to be PSPACE-hard and in EXPTIME [3]. For the remaining decision problems we have shown here that they are all EXPTIME-complete in the sum of the sizes of mixed or modal specifications.

We have appealed to known reductions between some of these problems [3] and, crucially, to a new reduction of input acceptance for linearly bounded alternating Turing machines to the existence of a common implementation for modal specifications – sketched in this extended abstract. The exact complexity of thorough refinement for modal specifications is subject to further investigation.

References

[1] Antonik, A. and M. Huth, *On the complexity of semantic self-minimization*, in: *Proc. AVOCS 2007*, to appear in ENTCS.

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

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

[4] Bruns, G. and P. Godefroid, *Generalized model checking: Reasoning about partial state spaces*, in: C. Palamidessi, editor, *CONCUR*, Lecture Notes in Computer Science **1877** (2000), pp. 168–182.

[5] Cerans, K., J. C. Godskesen and K. G. Larsen, *Timed modal specification - theory and tools*, in: *CAV '93: Proceedings of the 5th International Conference on Computer Aided Verification* (1993), pp. 253–267.

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

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

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

[9] Dams, D., R. Gerth and O. Grumberg, *Abstract interpretation of reactive systems*, ACM Trans. Program. Lang. Syst. **19** (1997), pp. 253–291.

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

[11] Godefroid, P. and M. Huth, *Model checking vs. generalized model checking: Semantic minimizations for temporal logics*, in: *LICS* (2005), pp. 158–167.

[12] Godefroid, P. and R. Jagadeesan, *On the expressiveness of 3-valued models*, in: L. D. Zuck, P. C. Attie, A. Cortesi and S. Mukhopadhyay, editors, *VMCAI*, Lecture Notes in Computer Science **2575** (2003), pp. 206–222.

[13] Hussain, A. and M. Huth, *On model checking multiple hybrid views*, Technical report, Department of Computer Science, University of Cyprus (2004), TR-2004-6.
URL http://pubs.doc.ic.ac.uk/hybrid-logic-multiple-views/

[14] Hussain, A. and M. Huth, *Automata games for multiple-model checking*, Electr. Notes Theor. Comput. Sci. **155** (2006), pp. 401–421.

[15] Huth, M., *Labelled transition systems as a Stone space*, Logical Methods in Computer Science **1** (2005), pp. 1–28.
URL http://pubs.doc.ic.ac.uk/labelled-systems-metrics-Stone/

[16] Hüttel, H., "Operational and Denotational Properties of Modal Process Logic," Master's thesis, Computer Science Department. Aalborg University (1988).

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

[18] Larsen, K. G., *Modal specifications.*, in: J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, Lecture Notes in Computer Science **407** (1989), pp. 232–246.

[19] Larsen, K. G., U. Nyman and A. Wąsowski, *Modal I/O automata for interface and product line theories.*, in: R. D. Nicola, editor, *ESOP*, Lecture Notes in Computer Science **4421** (2007), pp. 64–79.

[20] Larsen, K. G., U. Nyman and A. Wąsowski, *On modal refinement and consistency*, in: L. Caires and V. T. Vasconcelos, editors, *CONCUR*, Lecture Notes in Computer Science **4703** (2007), pp. 105–119.

[21] Larsen, K. G., B. Steffen and C. Weise, *A constraint oriented proof methodology based on modal transition systems*, in: *Tools and Algorithms for Construction and Analysis of Systems*, 1995, pp. 17–40.
URL citeseer.ist.psu.edu/article/larsen95constraint.html

[22] Larsen, K. G., B. Steffen and C. Weise, *Fischer's protocol revisited: a simple proof using modal constraints*, Lecture Notes in Computer Science **1066** (1996), pp. 604–615.
URL citeseer.ist.psu.edu/larsen96fischers.html

[23] Larsen, K. G. and B. Thomsen, *A modal process logic*, in: *Third Annual IEEE Symposium on Logic in Computer Science (LICS)* (1988).

[24] Larsen, K. G. and L. Xinxin, *Equation solving using modal transition systems*, in: *Fifth Annual IEEE Symposium on Logics in Computer Science (LICS), 4–7 June 1990, Philadelphia, PA, USA*, 1990, pp. 108–117.

[25] Park, D., *Concurrency and automata on infinite sequences*, in: *Proceedings of the 5th GI-Conference on Theoretical Computer Science* (1981), pp. 167–183.

[26] Schmidt, H. and H. Fecher, *Comparing disjunctive modal transition systems with a one-selecting variant*, Submitted for publication (2007).
URL http://www.informatik.uni-kiel.de/~hf/papers/Fecher07CDMTSO-Sub.pdf

[27] Sipser, M., "Introduction to the Theory of Computation," International Thomson Publishing, 1996.

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

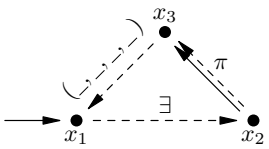


Fig. 5. Specification M_{exist} enforces a π-transition after each $\exists$-transition.

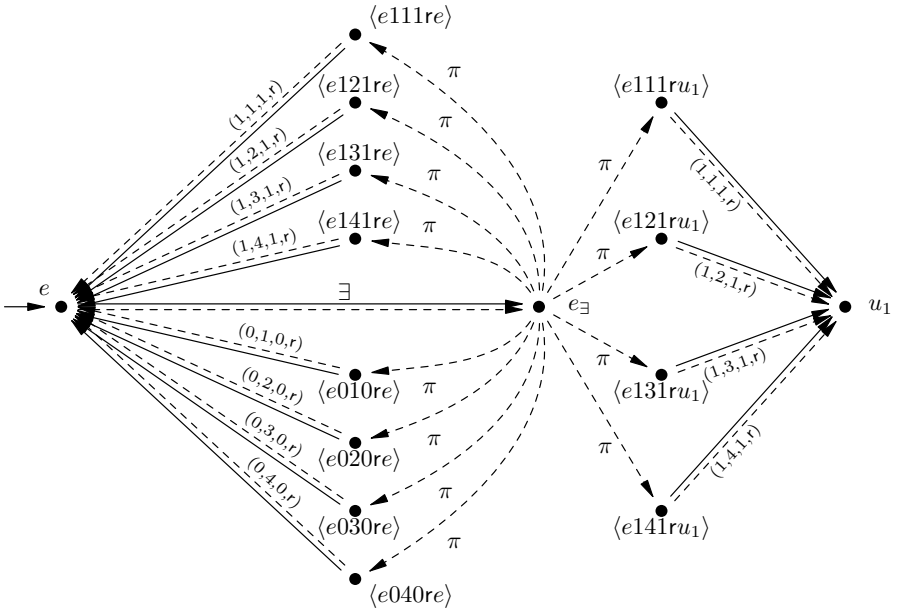


Fig. 6. Encoding for the existential state of the running example, assuming $|w| = 4$.

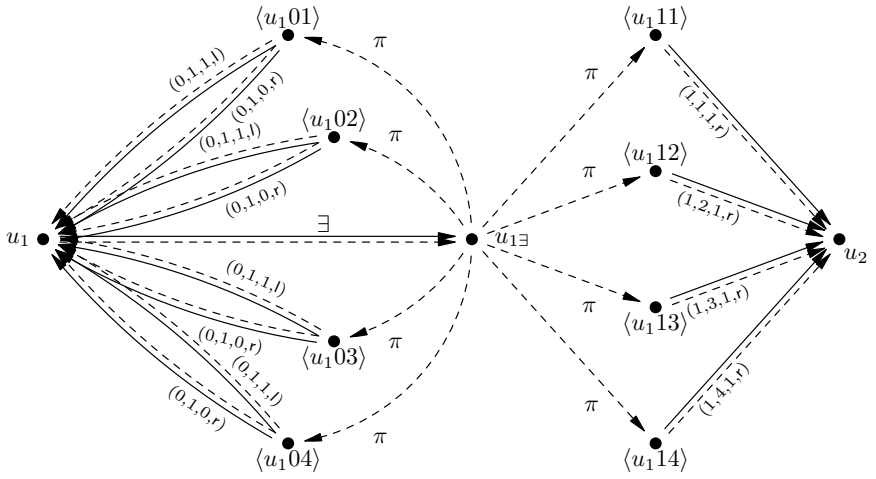


Fig. 7. Encoding for the universal state u_1 of the running example, assuming $|w| = 4$.

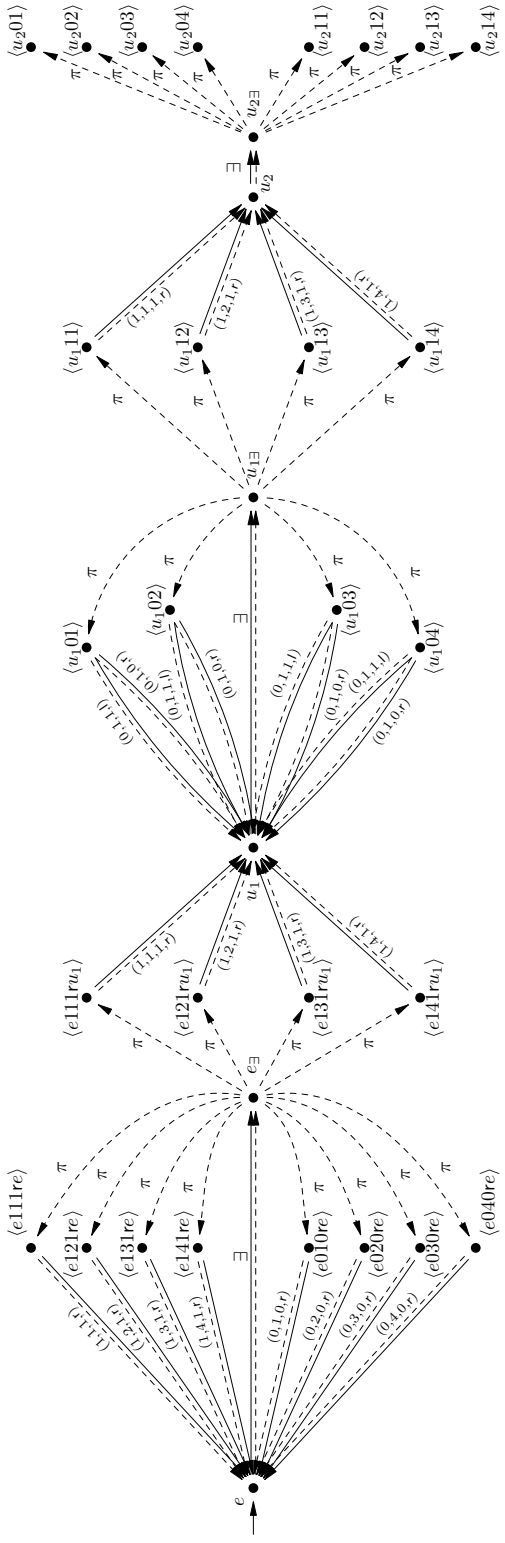


Fig. 8. The entire specification M_{ctrl} for the example of Figure 1 assuming $|w| = 4$.