Complexity of Decision Problems for Mixed and Modal Specifications*

Adam Antonik[1], Michael Huth[1],
Kim G. Larsen[2], Ulrik Nyman[2], and Andrzej Wąsowski[2,3]

[1] Department of Computing, Imperial College London, United Kingdom
{aa1001,mrh}@doc.imperial.ac.uk
[2] Department of Computer Science, Aalborg University, Denmark
{kgl,ulrik}@cs.aau.dk
[3] IT University of Copenhagen, Denmark
wasowski@itu.dk

Abstract. We consider decision problems for modal and mixed transition systems used as specifications: the *common implementation* problem (whether a set of specifications has a common implementation), the *consistency* problem (whether a single specification has an implementation), and the *thorough refinement* problem (whether all implementations of one specification are also implementations of another one). *Common implementation* and *thorough refinement* are shown to be PSPACE-hard for modal, and so also for mixed, specifications. *Consistency* is PSPACE-hard for mixed, while trivial for modal specifications. We also supply upper bounds suggesting strong links between these problems.

1 Introduction

Bisimulation equivalence [1] is widely accepted as a correctness criterion for realizations of abstract specifications. Bisimulation is, however, a rather strong relation that severely, and often unnecessarily, limits the choices of designers in how specifications should be realized. At the same time, the main alternative, bisimulation's sister preorder *simulation* [1], is often too weak to use in this context as it only limits faulty behaviours, without enforcing any correct ones.

In order to address these shortcomings, Larsen and Thomsen [2] have proposed *modal transition systems* and the accompanying *modal refinement*, in this paper referred to simply as refinement. Modal transition systems feature required and allowed transitions able to simultaneously describe an under- and over-approximation of behavior within a single specification. Modal refinement generalizes both simulation and bisimulation, letting the specifier choose the required level of strictness in the spectrum between the two. In [3] Larsen argued that any sufficiently expressive specification language necessarily must accommodate inconsistent specifications, akin to inconsistent logical formulæ, and thus

* 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)*

lifted the consistency requirement. The same type of systems were independently reintroduced by Dams as *mixed transition systems* [4, 5].

Here we establish complexities of several decision procedures for this family of specification languages, addressing several long outstanding open problems:

- CI Deciding whether $k > 1$ modal transition systems have a *common implementation* is PSPACE-hard in the sum of the sizes of these k systems.
- C Deciding whether a mixed transition system is *consistent*, i.e. whether it has an implementation, is PSPACE-hard in the size of that system.
- TR Deciding whether one modal transition system *thoroughly refines* another modal transition system is PSPACE-hard in the size of these systems.

We show quite strong links between these problems. In particular we efficiently reduce problems of type CI to problems of type C, and problems of type C to problems of type TR—though *mixed*, not necessarily modal, transition systems are the targets of that latter reduction. All three problems C, CI, and TR are shown to be in EXPTIME.

We begin with discussing the related work in Section 2 and introducing the basic concepts in Section 3. The hardness results and the aforementioned problem reductions for common implementation, consistency, and thorough refinement are the subject of Sections 4, 5, and 6 respectively. A general discussion, including the provision of upper bounds, is given in Section 7. We conclude in Section 8.

2 Related Work

Our terminology differs from that used in [6]: what we call "modal transition systems" and "mixed transition systems" are called respectively "syntactically consistent modal transition systems" and "modal transition systems" therein.

In [7] a superpolynomial algorithm was given for deciding CI for $k > 1$ modal specifications. The algorithm is exponential in k, but polynomial if k is fixed. In particular, it computes a common implementation if there is one. These upper bounds also follow easily from the polynomial algorithm for consistency checking of a conjunction of disjunctive modal transition systems, as studied in [8].

Larsen et al. [6] show that TR is coNP-hard, while C is NP-hard. We strengthen both of these bounds here. They also hint at exponential upper bounds for both problems, without arguing how these can be achieved. We elaborate on how to attain these bounds, by giving precise reductions in Section 7.

Hussain and Huth [9] 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 with specifications of product families. They propose a new thorough refinement whose implementations are defined through a refinement notion that generalizes branching bisimulation. The thorough refinement obtained in this manner is finer than weak refinement, and argued to be more suitable for conformance checking. In the light of the present work it is very likely that this refinement can be shown to be PSPACE-hard in the size of the specifications.

3 Background

Let us begin with defining the basic objects of interest in our study [11, 4, 12]:

Definition 1. *For an alphabet Σ, a* mixed transition system *M is a triple $(S, R^{\Box}, R^{\Diamond})$, where S is a set of states and $R^{\Box}, R^{\Diamond} \subseteq S \times \Sigma \times S$ are must- and may-transitions relations respectively. A* modal transition system *is a mixed transition system satisfying $R^{\Box} \subseteq R^{\Diamond}$; all its must-transitions are also may-transitions. A* pointed *mixed (respectively modal) transition system (M, s) is a mixed (modal) transition system M with a designated initial state $s \in S$. The size $|M|$ of a mixed (modal) transition system M is defined as $|S| + |R^{\Box} \cup R^{\Diamond}|$. All transition systems considered here are finite, i.e. Σ and S are always finite sets.*

Throughout this paper we refer to pointed modal (mixed) transition systems as modal (mixed) *specifications*. Throughout figures, solid arrows denote $R^{\Box}$-transitions, dashed arrows denote $R^{\Diamond}$-transitions. Arrows without labels have an implicit $\star$-label, where $\star$ is an action with context-dependent meaning. Two examples of modal specifications are depicted in Fig. 1, while a mixed specification that is not a modal specification can be seen in Fig. 5.

Modal refinement [11, 4, 12] is a refinement relationship for mixed specifications that allows verifying that one such specification is more abstract than another. It generalizes bisimulation [13] to underspecified models:

Definition 2. *A mixed specifications $(N, t_0) = ((S_N, R^{\Box}_N, R^{\Diamond}_N), t_0)$* refines *another mixed specification $(M, s_0) = ((S_M, R^{\Box}_M, R^{\Diamond}_M), s_0)$ over the same alphabet, 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*

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 one finite-state mixed specification refines another one is in P.

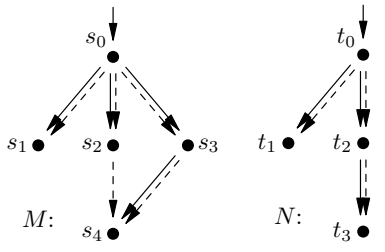

Fig. 1. Specifications (M, s_0), (N, t_0) with $I(M, s_0) = I(N, t_0)$ (so $I(M, s_0) \subseteq I(N, t_0)$), but not $(N, t_0) \prec (M, s_0)$

Labeled transition systems over an alphabet Σ are pairs (S, R) where S is a set of states and $R \subseteq S \times \Sigma \times S$ is a transition relation. We identify labeled transition systems (S, R) with modal transition systems (S, R, R). The set of implementations $I(M, s)$ of a mixed specification (M, s) are all pointed labeled 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 transition system.

Example. (Due to Harald Fecher) Figure 1 shows modal specifications (M, s) and (N, t) over alphabet $\{\star\}$. Relation $Q = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_2), (s_4, t_3)\}$ witnesses that (N, t_0) refines (M, s_0), but (M, s_0) does not refine (N, t_0).

As in [6] we define the *thorough refinement* $(M,s) \prec_{th} (N,t)$ to be the predicate $I(N,t) \subseteq I(M,s)$. Transitivity of refinement ensures that refinement soundly characterizes thorough refinement: $(M,s) \prec (N,t)$ implies $(M,s) \prec_{th} (N,t)$. But the converse does not hold: completeness of refinement for thorough refinement, is known to be false [14–16]; Figure 1 provides a counterexample.

We shall now formally define the problems that we study, and briefly discuss their significance.

Common implementation (CI): given $k > 1$ mixed specifications (M_i, s_i), is the set $\bigcap_{i=1}^{k} I(M_i, s_i)$ non-empty? For example, (M_1, s_1) could be our system model and all other (M_i, s_i) could be definitions of faulty behavior (respectively features). Common implementations are then possible implementations of our model that can exhibit all $k-1$ faults (features).

Consistency (C): Is $I(M,s)$ non-empty for a mixed specification (M,s)? Specification formalisms need the ability to express inconsistencies so that conflicts in systems or their design are detectable. Equally, inconsistent specifications may well result from the composition of consistent specifications.

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 refinement is only sound but not complete for thorough refinement, the question arises of whether thorough refinement has an efficient, e.g. co-inductive, definition that can be integrated in refinement tools.

We assume that specifications are finite-state, given their abstract nature. But implementations may (have to) be infinite-state as we otherwise cannot express important features, e.g. unbounded ranges of data types. For the three decision problems studied in this paper, it turns out that they won't change if we restrict implementations to finite-state ones. For example, a mixed specification (M,s) is consistent in the infinite sense iff its characteristic modal mu-calculus formula $\Psi_{(M,s)}$ [17] is satisfiable. Appealing to the small model theorem for that logic, $\Psi_{(M,s)}$ is satisfiable iff it is satisfiable over finite-state implementations. We can reason in a similar manner about common implementation, through the formula $\bigwedge_i \Psi_{(M_i,s_i)}$. Finally, $(M,s) \prec_{th} (N,t)$ is false iff $\Psi_{(N,t)} \wedge \neg\Psi_{(M,s)}$ is satisfiable. This justifies that we consider only 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 PSPACE-complete.

4 Common Implementation

We show that the CI problem is PSPACE-hard for modal specifications, which then automatically renders the same hardness result for mixed specifications.

Theorem 3. *Let $\{(M_i, s_i) \mid 1 \leq i \leq k\}$ with $k > 1$ be a finite family of modal specifications over the same action alphabet Σ. Deciding emptiness of the set $\bigcap_{i=1}^{k} I(M_i, s_i)$ is PSPACE-hard in $\sum_{i=1}^{k} |M_i|$.*

We argue for this by reduction from the Generalized Geography game [18, 19].

Definition 4. *A rooted, directed graph is a structure $G = (V, E, v_0)$, where V is a finite set of vertices, $E \subseteq V \times V$ is a set of edges and $v_0 \in V$ is the root. For an edge $e = (u, v) \in E$ we write $\mathbf{tgt}\, e$ for v and $\mathbf{src}\, e$ for u, and we define $Follow(e) := \{f \in E \mid \mathbf{tgt}\, e = \mathbf{src}\, f\}$ and $Init := \{e \in E \mid \mathbf{src}\, e = v_0\}$.*

For $G = (V, E, v_0)$ the two-player *Generalized Geography* game on G is played according to the following rules:

> "The two players alternate choosing a new edge from E. The first edge chosen (by player 1) must have its source at v_0 and each subsequently chosen edge must have its source at the vertex that was the target of the previous edge and must not have been previously chosen in the game. The first player unable to choose such a new edge loses." [18, p. 254]

The generalized geography problem (GenGeo) is whether given a rooted directed graph G does there exist a winning strategy for player 1 in the *Generalized Geography* game played on G? GenGeo is PSPACE-complete [18].

Proof (of Theorem 3). We reduce GenGeo to checking CI of k modal specifications $\{(M_i, s_i)\}$, where both k and each $|M_i|$ are at most polynomial in the size of G. The reduction should be such that a common implementation of all (M_i, s_i), if it exists, will explicitly give the winning strategy for Player 1.

We will create a set of modal specifications for each kind of conditions imposed by the game. All specifications will share an alphabet $\Sigma = E \cup \{\star\}$, where $\star$ is a fresh name such that $\star \notin E$. Choosing an edge in the game corresponds to taking a transition in these specifications.

Let us begin with modal specifications (P_1, s_1) and (P_2, s_2) presented in Figure 2, which ensure that Player 1 can always continue – a necessary condition for obtaining a winning strategy. Transitions with labels $X \subseteq \Sigma$ denote sets of transitions, one for each $e \in X$. We keep track of whose turn it is in the game by distinguishing Player 1 states from Player 2 states, labeling states with Player numbers for the sake of clarity. Observe that both P_1 and P_2 oscillate between Player 1 and Player 2 decisions. Each Player 2 move is modeled directly by a single transition, while a Player 1 move is modeled by exactly two transitions; a $\star$-transition followed by a regular edge transition. As will be seen later, disjunctive choices will only occur in Player 1 mode, so $\star$-transitions used to encode disjunctions are there only for Player 1 states. Specification P_1 limits choices of Player 1 to a disjunction of all legal actions, while P_2 enforces that at least one of these choices is indeed taken.

Let us continue with the remaining GenGeo game rules. We can enforce that an edge e is played at most once using a modal specification (M_e, s_e) shown in the left part of Figure 3. This specification models a flag that disallows any further e-transitions once e has been used. Similarly, for each edge e create a modal specification (N_e, t_e), as shown in the right part of Figure 3, to constrain the moves following an e move to edges directly following it. N_e has a $\star$-labeled

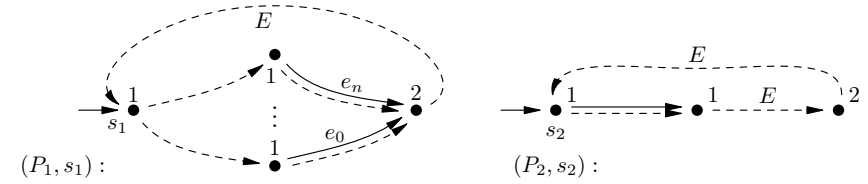

Fig. 2. Modal specifications (P_1, s_1) and (P_2, s_2) together ensuring that Player 1 can always continue playing. Assume $E = \{e_0, \dots e_n\}$

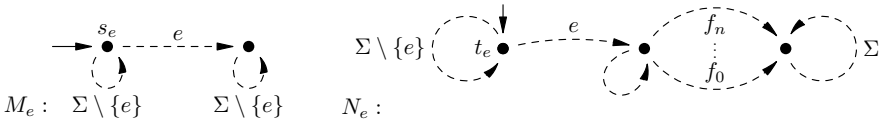

Fig. 3. Specifications M_e, N_e instantiated for each $e \in E$ and $\{f_0, \dots, f_n\} = Follow(e)$

loop on its middle state to account for both Player 1 and Player 2 moves. Recall that if e was played by Player 2, then in our encoding it will be first followed by a $\star$ before Player 1 plays any subsequent edge. The requirement that Player 1 should choose one of the transitions leaving the root as the first move is enforced by (P_0, s_0) as shown in the left part of Figure 4.

We are left with the last and the most complex game rule, namely that whenever Player 1 makes a choice then Player 2 has to be able to respond with any so far unused edge f following that choice. Our implementation, which directly represents the strategy, should thus have all transitions representing possible choices in such a state. We model this by creating a specification (M_{ef}, s_{ef}) for every pair of edges e and f such that $f \in Follow(e) \setminus \{e\}$. The idea is that each modal transition system M_{ef} enforces an f transition after an e transition has been chosen by Player 1, unless f has already been used (either by Player 1 or Player 2), or e has been used by Player 2. See the right part of Figure 4.

The answer to GenGeo(V, E, v_0) is yes iff the answer to CI is yes for

$$\left(\bigcup_{i=0..2} \{(P_i, s_i)\} \right) \cup \bigcup_{e \in E} \left(\{(M_e, s_e), (N_e, t_e)\} \cup \bigcup_{f \in Follow(e) \setminus \{e\}} \{(M_{ef}, s_{ef})\} \right) . \quad (1)$$

The size of each of these $O(|E|^2)$ specifications is $O(|E|)$. □

Corollary 5. *The common implementation problem for $k > 1$ mixed specifications is PSPACE-hard in the size of these specifications.*

Proof. This follows from Theorem 3 and the fact that the set of mixed specifications is a superset of the set of modal specifications. □

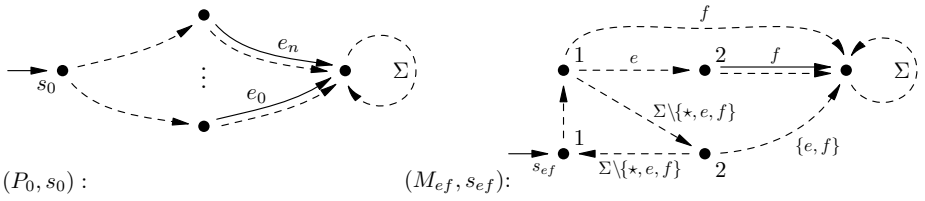

Fig. 4. Specifications (P_0, s_0) and (M_{ef}, s_{ef}) assuming that $\mathit{Init} = \{e_0, \ldots, e_n\}$

5 Consistency

Let us now show that consistency of a single mixed specification is PSPACE-hard in its size. We achieve this by appealing to Theorem 3, and reducing CI for several modal specifications to the C for a single mixed specification.

Theorem 6. *Consistency of a mixed specification is PSPACE-hard.*

Proof. By Theorem 3, 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 iff all (M_i, s_i) have a common implementation.

Figure 5 illustrates the construction, which originates in [6], 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 s_2, two new $\star$-transitions are added from a fresh state c_2 to each of s_1, s_2. One of the $\star$-transitions is a may $\star$-transition and the other is a must $\star$-transition. Only two states can be conjoined directly in this way, but the process can be iterated as many times as needed, as seen in the figure, by adding a corresponding number of $\star$-transitions to the newly conjoined systems. Observe that the resulting specification is properly mixed (not modal). Its size is linear in $\sum_i |\,M_i\,|$ and quadratic in k, which itself is $O(\sum_i |\,M_i\,|)$.

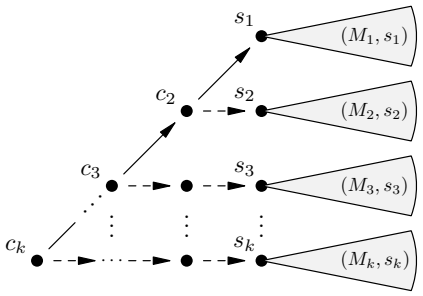

Fig. 5. Conjunction of k mixed specifications into one mixed specification

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

6 Thorough Refinement

We show PSPACE-hardness of TR for mixed specifications by appeal to Theorem 6 and a reduction of consistency checks to thorough refinement checks.

Theorem 7. *Thorough refinement of mixed specifications is PSPACE-hard in the size of these specifications.*

Proof. By Theorem 6 deciding C for a mixed specification is PSPACE-hard. Therefore it suffices to reduce C to TR. Let (M, s) be a mixed specification over Σ. Consider a modal specification (N, t) over $\Sigma \cup \{\star\}$ with $N = (\{t\}, \{\}, \{\})$, which only has a single state and no transitions. From (M, s) construct the mixed specification (M', s') over $\Sigma \cup \{\star\}$ by prefixing s with a new state s' and a single transition $(s', \star, s) \in R^{\diamond}_{M'} \setminus R^{\Box}_{M'}$. Then (M', s') is a mixed specification that has (N, t) as an implementation, where $Q = \{(s', t)\}$ is the witnessing refinement relation. We show that (M, s) is consistent iff not $(N, t) \preceq_{th} (M', s')$.

1° If (M, s) is consistent, then 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 $(l', \star, l)$. But then (M', s') has an implementation that is not allowed by (N, t) and so $I(M', s') \not\subseteq I(N, t)$.
2° Conversely, if $I(M', s') \not\subseteq I(N, t)$ then there exists an implementation (L, l') of (M', s'), which is not an implementation of (N, t) – and so (L, l') has a transition $(l', \star, l)$. 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.

Remark: Observe that the first argument above would also work for refinement instead of thorough refinement. However we would not be able to get the second implication for refinement, due to its incompleteness. □

Let us now strengthen Theorem 7 to the subclass of *modal* specifications, by a polynomial reduction from the PSPACE-complete decision problem QUANTIFIED 3SAT [18, pp. 171-2] of computing the truth value of closed quantified Boolean formulæ in 3CNF. These formulæ are of the form $Qx_1 \ldots Qx_n . \chi$, where each Q is $\exists$ or $\forall$ and χ is a propositional formula over $x_1, \ldots, x_n$ in 3CNF. We refer to them as QCNF formulæ in here. We can assume without loss of generality that our formulæ do not contain any clauses with duplicate literals, nor vacuously true clauses. We use $\forall x \, \exists y \, (\neg x \vee y) \wedge (\neg y \vee x)$ as a running example.

We present the semantics of QCNF formulæ in a style that will facilitate our proof. Each formula φ can be rewritten into a set of *valuation trees*. The non-deterministic rewrite system for this is depicted in Figure 6. Universal quantification rewrites into branching, existential quantification into a choice, and the 3CNF kernel χ into the set of variables selected to be true on the path from the tree root to that kernel node. The terminals of this rewrite system for term $(\varphi, \emptyset)$ are valuation trees of φ. One such valuation tree for the formula $\forall x \, \exists y \, (\neg x \vee y) \wedge (\neg y \vee x)$ can be seen in Figure 7. Each leaf of a valuation tree T contains all those x_i that are true in the respective model for the propositional

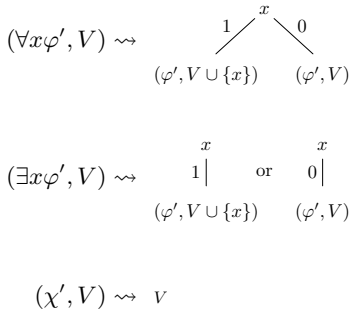

Fig. 6. Semantics of QCNF as a non-deterministic rewrite system

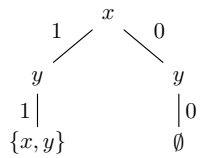

Fig. 7. Valuation tree witnessing the truth of $\forall x\, \exists y\, (\neg x \vee y) \wedge (\neg y \vee x)$

kernel formula χ. We define $T \models \varphi$ to mean that all models of leaves of T satisfy the kernel χ of φ. Finally, φ is defined to be true iff there is a valuation tree T for φ such that $T \models \varphi$. For example, $T \models \varphi$ for the valuation tree T in Fig. 7, as the CNF kernel $(\neg x \vee y) \wedge (\neg y \vee x)$ is true in both models $\{x, y\}$ (x and y are true) and $\emptyset$ (x and y are false). Thus, φ is true.

In Figure 8 we present a second non-deterministic rewrite system whose terminals are *potential valuation trees*. In this new system there is no path context, existential quantification has two more rewrite rules, and the CNF kernel may rewrite into any subset of its variables. The terminals of this rewrite system are *potential* valuation trees of φ. By construction, every valuation tree is a potential valuation tree. A potential valuation tree that is not a valuation tree is called a *flawed valuation tree*. Figure 9 shows a valuation tree for our running example with three kinds of flaws: the leftmost y node has no successor, the rightmost y node has two successors, and the leaf set $\{x, y\}$ is inconsistent with the 0 label for x on its path.

Our reduction constructs for any φ of QCNF two modal specifications (N_φ, t_φ) and (M_φ, s_φ) such that

$$I(N_\varphi, t_\varphi) \subseteq I(M_\varphi, s_\varphi) \qquad \text{iff} \qquad \varphi \text{ is false.} \tag{2}$$

The intuition behind the construction is that (N_φ, t_φ) models potential valuation trees and (M_φ, s_φ) models flawed, and only flawed, valuation trees of φ.

More precisely, these modal specifications are such that any valuation tree T with $T \models \varphi$ can be transformed into an implementation of (N_φ, t_φ) that is not an implementation of (M_φ, s_φ) and, conversely, that any element of $I(N_\varphi, t_\varphi) \setminus I(M_\varphi, s_\varphi)$ can be transformed into such a valuation tree T with $T \models \varphi$.

Both models are defined over the following alphabet

$$\Sigma_\varphi = \{\star\} \cup \{v_{x_i}, v_{\neg x_i} \mid 1 \leq i \leq n\} \tag{3}$$

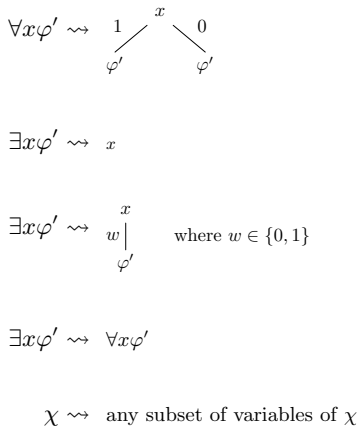

Fig. 8. Non-deterministic rewrite system for QCNF deriving *potential* valuation trees

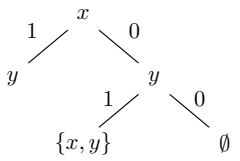

Fig. 9. Flawed valuation tree for formula $\forall x\, \exists y\, (\neg x \vee y) \wedge (\neg y \vee x)$

where $x_1, \ldots, x_n$ is the set of variables of φ.[1] Specification (N_φ, t_φ) is defined by structural induction on φ according to the rules presented in Figure 10. The initial state t_φ has a must $\star$-transition to the continuation of the compilation of N_φ. Each quantifier Qx_i gets translated into a diamond shaped model of $\star$-transitions, where the upper half consists of must and may transitions for quantifiers $\forall$ and $\exists$ (respectively). The corners of diamonds have "spikes", transitions labeled with a "truth value" v_{x_i} or $v_{\neg x_i}$, for quantifier variable x_i, to a dead-end state. After all quantifiers have been compiled in this manner, conjunction is compiled as a fork of two must $\star$-transitions, disjunction as a fork of two may $\star$-transitions, and literals compiled as spikes of truth values. See the result of this compilation for our running example in Figure 11.

Refinement, as defined for modal specifications, does not guarantee that a fork of may $\star$-transitions (present in the compilation of $\exists x_i$ and $\vee$) will implement at least one of these may $\star$-transitions. Also, an implementation may be inconsistent as to its choice of truth values v_{x_i} or $v_{\neg x_i}$. Each path through a sequence of diamonds corresponds to a choice of such truth values, recorded in the respective spike transition. When such a path reaches the compilation of a propositional literal, that literal may well be inconsistent with the spike for that literal encountered en route. In total, these are then the static criteria for corresponding to a flawed valuation tree, and hence drive the construction of specification (M_φ, s_φ), whose architecture is depicted in Fig. 12. Initial state s_φ has may $\star$-transitions to modal specifi-

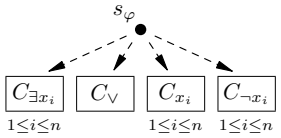

Fig. 12. Structure of modal specification (M_φ, s_φ): $\star$-transitions lead from s_φ to components that detect possible flaws in potential valuation trees of φ

[1] A stronger, albeit more complicated, reduction is possible to TR of specifications over a singleton alphabet. We show the simpler variant here for the sake of clarity.

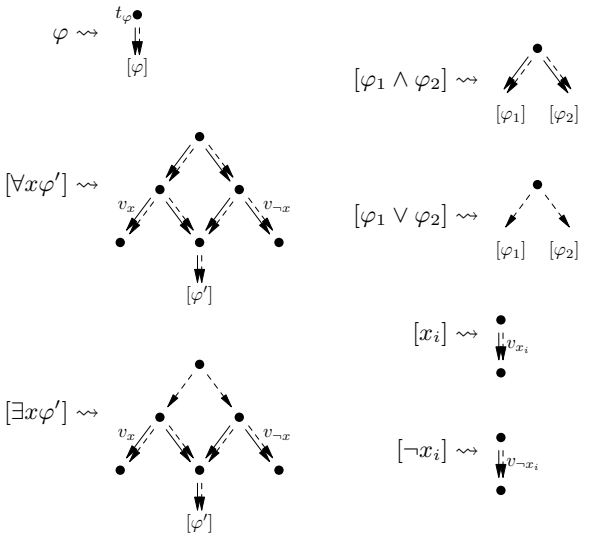

Fig. 10. Deterministic rules rewriting a QCNF formula φ into a specification (N_φ, t_φ)

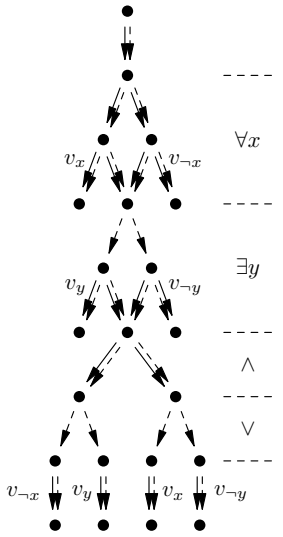

Fig. 11. Modal specification (N_φ, t_φ) for $\varphi = \forall x \, \exists y \, (\neg x \vee y) \wedge (\neg y \vee x)$

cations, components that each encode a potential flaw for a valuation tree. For each variable x_i of φ we have a component

- $C_{\exists x_i}$, whose M_φ-implementations have no "witness" for $\exists x_i$, i.e., no may transitions on the top of the diamond encoding the quantifier.
- C_{x_i}, whose M_φ-implementations have a path on which there is some v_{x_i} spike but where, on that same path, a $v_{\neg x_i}$-transition occurs subsequently
- $C_{\neg x_i}$, whose M_φ-implementations have a path on which there is some $v_{\neg x_i}$ spike but where, on that same path, a v_{x_i}-transition occurs subsequently.

Finally there is a component $C_\vee$ whose M_φ-implementations all have a path of 3n $\star$-transitions to a dead-end state, and so no such implementation can encode all disjunctions of φ correctly.

Based on the constructions we can present the following theorem.

Theorem 8. *Thorough refinement between modal specifications is PSPACE-hard in the size of these specifications.*

Since the modal transition systems N_φ and M_φ can be constructed in polynomial time in the size of φ, it suffices to show that (3) holds. The proof of this fact can be found in the Appendix.

Note that, by construction, $((\{s_\varphi\}, \emptyset, \emptyset), s_\varphi)$ is an implementation of (M_φ, s_φ) but not of (N_φ, t_φ). So the result also applies to strict thorough refinement.

Corollary 9. *Strict thorough refinement, whether $I(N,t) \subset I(M,s)$, is PSPACE-hard in $|M|$ and $|N|$ for modal and thus also for mixed specifications.*

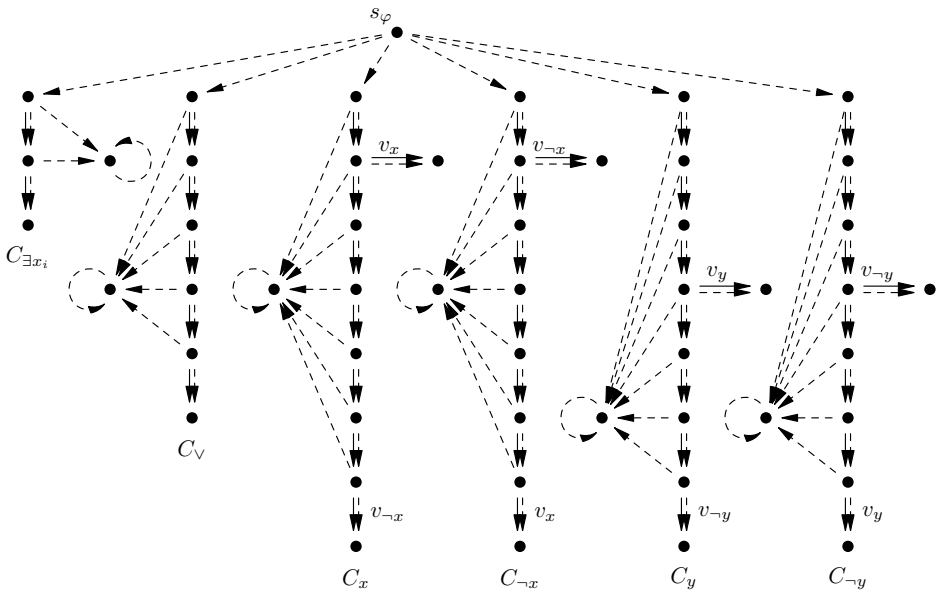

Fig. 13. Modal specification (M_φ, s_φ) for $\varphi = \forall x \, \exists y \, (\neg x \vee y) \wedge (\neg y \vee x)$. All incoming and outgoing transitions of all loop states are labeled with Σ_φ (omitted for clarity)

7 Discussion

First, we relate our results to the complexity of related problems. Second, we discuss and derive our upper bounds.

In [20] efficient translations are given between various classes of 3-valued models such that these translations preserve and reflect the respective refinement notions. These classes of models are all consistent and one of them subsumes modal transition systems. Therefore our complexity results for common refinement and thorough refinement for modal transition systems transfer to these model classes if we define our three concepts in the same manner for each respective notion of refinement. In particular, our complexity results apply to partial Kripke structures and Kripke modal transition systems.

It is likely that our results extend to "weak" refinement notions that generalize weak bisimulation. This, however, requires a further study. Such refinement notions were systematically studied in [6].

The "conjunction" gadget used in reducing the common implementation problem for modal transition systems to consistency of a mixed transition system (Section 5) is able to identify states uniquely based on the may/must pattern of transitions encountered en route from the initial state. Nominals, used in hybrid logic [21], are a well known mechanism for identifying states uniquely. One can show NP-hardness of the common implementation problem for *two* modal transition systems already if such systems are enriched with nominals [22].

If specifications are "closed under negation" in that $\neg(M,s)$ has the complement of $I(M,s)$ as set of implementations, then thorough refinement reduces to common implementation: $(M,s) \prec_{th} (N,t)$ is false iff (M,s) and $\neg(N,t)$ have a common implementation. From the results in [17] it follows easily that modal transition systems do not have such a negation. Support of negation for specifications should require more structure than that found in mixed transition systems. Another open problem is whether non-empty languages $I(M,s)$ accepted by mixed specifications (M,s) can also be accepted by modal specifications; in other words—if a mixed specification is consistent, is it refinement-equivalent to a modal specification?

Generalized model checking [23] considers judgments $\mathsf{GMC}(M,s,\varphi)$ which hold iff there is an implementation of (M,s) that satisfies φ. For pointed modal specifications (M,s) and Hennessy-Milner formulae φ this is PSPACE-complete in the size of φ [23, 20]. For each such φ there are $1 \leq m < \infty$ pointed modal specifications (M_i,s_i) such that $\mathsf{GMC}(M,s,\varphi)$ is false iff $I(M,s) \subseteq \bigcup_{i=1}^{m} I(M_i,s_i)$ [17]. Intuitively, the union on the right-hand side is the set of implementations that satisfy $\neg\varphi$. In general, $m > 1$ so there seems to be no natural and direct reduction of generalized model checking to thorough refinement. For φ in CTL, $\mathsf{GMC}(M,s,\varphi)$ is EXPTIME-complete [23, 20] but $1 < m$ or $m = \infty$ may hold.

We finally discuss what upper bounds we can provide for the decision problems presented in this paper. Mixed and modal specifications (M,s) have characteristic formulæ $\Psi_{(M,s)}$ [17] in the modal μ-calculus such that pointed labeled transition systems (L,l) are implementations of (M,s) iff (L,l) satisfies $\Psi_{(M,s)}$. The common implementation and consistency problem reduce to satisfiability checks of $\bigwedge_i \Psi_{(M_i,s_i)}$ and $\Psi_{(M,s)}$, respectively. The thorough refinement problem of whether $(M,s) \prec_{th} (N,t)$ reduces to a validity check of $\neg\Psi_{(N,t)} \vee \Psi_{(M,s)}$.

Validity checking of such vectorized modal μ-calculus formulæ is in EXPTIME (an unpublished popular wisdom, for which we give a formal argument here). One way in which this membership in EXPTIME can be seen is by translating the problem into alternating tree automata. It is well known that formulæ $\Psi_{(M,s)}$ can be efficiently translated [24] into alternating tree automata $\mathsf{A}_{(M,s)}$ (with parity acceptance condition) that accept exactly those pointed labeled transition systems that satisfy $\Psi_{(M,s)}$. Since non-emptiness, intersection, and complementation of languages is in EXPTIME for alternating tree automata, we get our EXPTIME upper bounds if these automata have size polynomial in $|M|$. Since the size of $\Psi_{(M,s)}$ may be exponential in $|M|$ we require a direct translation from (M,s) into a version of $\mathsf{A}_{(M,s)}$. The formulæ $\Psi_{(M,s)}$ can be written as a system of recursive equations [3] $X_s = \mathit{body}_s$ for each state s of M. We can therefore construct all $\mathsf{A}_{(M,s)}$ in a compositional manner: whenever X_s refers in its body_s to some X_t, then $\mathsf{A}_{(M,s)}$ has a transition to the initial state of $\mathsf{A}_{(M,t)}$ at that point. This $\mathsf{A}_{(M,s)}$ generates the same language as the one constructed from $\Psi_{(M,s)}$, by appeal to the existence of memoryless winning strategies in parity games. The system of equations is polynomial in $|M|$, and so the compositional version of $\mathsf{A}_{(M,s)}$ is polynomial in the size of that system of equations. We summarize:

Theorem 10. *The common implementation, consistency and thorough refinement problems are all in EXPTIME for modal and mixed specifications.*

8 Conclusion

We studied modal and mixed specifications and their fundamental decision problems: consistency (a form of realizability), common implementations (a conjunctive form of consistency), and thorough refinement (a form of implication) of specifications. We established that all these decision problems are in EXPTIME and PSPACE-hard for mixed as well as for modal specifications – keeping in mind that all modal specifications are consistent by construction. These results showed that some of these decision problems are at least as hard as others studied here. This raises the question of whether they in fact have the same complexity.

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

	Modal specifications	Mixed specifications
Common implementation	PSPACE-hard, EXPTIME	PSPACE-hard, EXPTIME
Consistency	trivial	PSPACE-hard, EXPTIME
Thorough refinement	PSPACE-hard, EXPTIME	PSPACE-hard, EXPTIME

Acknowledgments. Harald Fecher made us aware of the counterexample for incompleteness of refinement used in this paper. This then led to the rediscovery of a history of such counterexamples. Nir Piterman helped in improving the presentation of the proof for Theorem 8. We thank Igor Walukiewicz, Wolfgang Thomas and Dietmar Berwanger for independently confirming that validity of vectorized μ-calculus formulæ is in EXPTIME.

References

1. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
2. Larsen, K.G., Thomsen, B.: A modal process logic. In: Third Annual IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society, IEEE Computer Society (1988) 203–210
3. Larsen, K.G.: Modal specifications. In Sifakis, J., ed.: Automatic Verification Methods for Finite State Systems. Volume 407 of Lecture Notes in Computer Science., Springer (1989) 232–246
4. Dams, D.: Abstract Interpretation and Partition Refinement for Model Checking. PhD thesis, Eindhoven University of Technology (July 1996)

5. Dams, D., Gerth, R., Grumberg, O.: Abstract interpretation of reactive systems. ACM Trans. Program. Lang. Syst. **19**(2) (1997) 253–291
6. Larsen, K.G., Nyman, U., Wąsowski, A.: On modal refinement and consistency. In Caires, L., Vasconcelos, V.T., eds.: CONCUR. Volume 4703 of Lecture Notes in Computer Science., Springer (2007) 105–119
7. Hussain, A., Huth, M.: On model checking multiple hybrid views. Technical report, Department of Computer Science, University of Cyprus (2004) TR-2004-6.
8. Larsen, K.G., Xinxin, L.: 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) 108–117
9. Hussain, A., Huth, M.: Automata games for multiple-model checking. Electr. Notes Theor. Comput. Sci. **155** (2006) 401–421
10. Fischbein, D., Uchitel, S., Braberman, V.: A foundation for behavioural conformance in software product line architectures. In: ROSATEA '06 Proceedings, New York, NY, USA, ACM Press (2006) 39–48
11. Larsen, K.G.: Modal specifications. In Sifakis, J., ed.: Automatic Verification Methods for Finite State Systems. Volume 407 of Lecture Notes in Computer Science., Springer (1989) 232–246
12. Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. ACM Trans. Program. Lang. Syst. **16**(5) (1994) 1512–1542
13. Park, D.: Concurrency and automata on infinite sequences. In: Proceedings of the 5th GI-Conference on Theoretical Computer Science, London, UK, Springer-Verlag (1981) 167–183
14. Hüttel, H.: Operational and denotational properties of modal process logic. Master's thesis, Computer Science Department. Aalborg University (1988)
15. Xinxin, L.: Specification and Decomposition in Concurrency. PhD thesis, Department of Mathematics and Comnputer Science, Aalborg University (April 1992)
16. Schmidt, H., Fecher, H.: Comparing disjunctive modal transition systems with a one-selecting variant. Submitted for publication to JLAP (2007)
17. Huth, M.: Labelled transition systems as a Stone space. Logical Methods in Computer Science **1**(1) (January 2005) 1–28
18. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
19. Jonsson, B., Larsen, K.G.: On the complexity of equation solving in process algebra. In Abramsky, S., Maibaum, T.S.E., eds.: TAPSOFT, Vol.1. Volume 493 of Lecture Notes in Computer Science., Springer (1991) 381–396
20. Godefroid, P., Jagadeesan, R.: On the expressiveness of 3-valued models. In Zuck, L.D., Attie, P.C., Cortesi, A., Mukhopadhyay, S., eds.: VMCAI. Volume 2575 of Lecture Notes in Computer Science., Springer (2003) 206–222
21. Franceschet, M., de Rijke, M.: Model checking hybrid logics (with an application to semistructured data). J. Applied Logic **4**(3) (2006) 279–304
22. Antonik, A.: MPhil/PhD transfer report (January 2007) Imperial College London, United Kingdom.
23. Bruns, G., Godefroid, P.: Generalized model checking: Reasoning about partial state spaces. In Palamidessi, C., ed.: CONCUR. Volume 1877 of Lecture Notes in Computer Science., Springer (2000) 168–182
24. Wilke, Th.: Alternating tree automata, parity games, and modal μ-calculus. Bull. Soc. Math. Belg. **8**(2) (May 2001)