The Concurrency Column

by

Luca Aceto

BRICS, Department of Computer Science
Aalborg University, 9220 Aalborg Ø, Denmark
luca@cs.auc.dk, http://www.cs.auc.dk/~luca/BEATCS

Modal transition systems are a variation on the classic model of transition systems where transitions come in two flavours: those that any refinement of the given specification *must* possess, and those that it *may, but is not required to,* have. This model of computation was introduced about twenty years ago by Kim Guldstrand Larsen and Bent Thomsen. Since then, it has been the subject of investigation by several groups of researchers, and interest in this model and in its sibling that goes by the name of *mixed specifications* has grown over the last few years. In the light of the recent rapid growth in the research literature on modal and mixed specifications, and their applications, I thought that it was appropriate to devote an installment of the Concurrency Column to a survey of recent results and open problems in the field. I am very happy to be in a position to offer the readers of the Concurrency Column this excellent overview paper by some of the prime movers in the development of the theory and applications of modal and mixed specifications. I trust that this piece will be of general interest, and I hope that it will entice several researchers to contribute to the on-going work on these models. Enjoy!

20 Years of Modal and Mixed Specifications

Adam Antonik & Michael Huth[*]
Kim G. Larsen & Ulrik Nyman[†] Andrzej Wąsowski[‡]

Abstract

Twenty years ago, modal and mixed specifications were proposed as abstract models of system behavior. In this paper, we explain the nature and utility of such specifications, relate them to other formalisms, showcase some of their established applications, and mention some existing tool support. We also present some recent complexity results for decision problems underlying such applications and list some remaining open problems.

1 Introduction

After one of the presentations at the FOSSACS 2008 conference questions had been asked about modal and mixed specifications:

> What are they? What popular and simple model of computation do they resemble? To what other formalisms can they be related?

One could easily continue asking:

> What kind of applications of mixed or modal specifications are known? Do these applications have tool support? Etc.

The 20th anniversary of the publication [37] that proposed modal specifications seems to be a suitable occasion for addressing such questions. We attempt to do this below by summarizing what is known, and what is unknown about modal and mixed specifications and the rich family of their associated models.

Modal specifications, often referred to as *modal transition systems*, were introduced by Larsen and Thomsen in [37]. They comprise two kinds of parts:

P1 a non-empty set of states,

P2 a transition relation R governing the temporal evolution of states, where transitions are labeled with actions.

Figure 1(a) illustrates such a specification. It models behavior of a simple measurement system, which after receiving a request, polls its sensor (poll) for a value, and reports the value back to the requester (report).

Already in [37] the authors aptly point out that the value of specifications consisting of a single P1 and a single P2 part is limited in that they cannot clearly distinguish between behaviors *insisted upon* and those merely *allowed*. From a

[*]Department of Computing, Imperial College London, `aa1001,mrh@doc.imperial.ac.uk`

[†]Department of Computer Science, Aalborg University, `kgl,ulrik@cs.aau.dk`

[‡]IT University of Copenhagen, `wasowski@itu.dk`

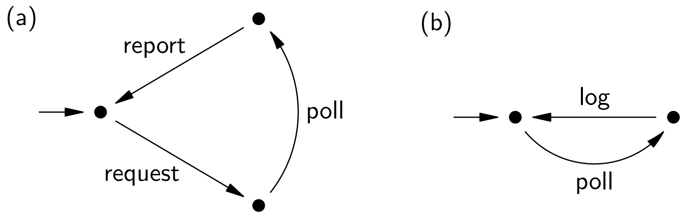

Figure 1: (a) Specifying behavior of a measurement system by labeling transitions with actions. (b) Similar specification of another aspect of the same sensor system.

theoretical perspective this means that a specification can only represent implementations to which it is bisimilar. So if a specification is concentrating on a specific aspect of a system it models (e.g. security), it will not be consistent with implementations that exhibit aspects not captured in that specification (e.g. log creation). For example, in Figure 1(b), another aspect of our measurement system is specified, namely that all polled values should be logged locally in the system for future reference. Clearly, the models in Fig. 1(a) and Fig. 1(b) are inconsistent in the sense that there exists no single implementation bisimilar to both: the former does not allow any log actions required by the latter, while the latter does not allow any of the request and report actions, required by the former. This situation effectively prevents specifications of different parts or aspects of a system from being reconciled within a single implementation through any refinement based on simulation relations.

In order to differentiate between required and allowed aspects of a specification we simply furnish the state set with two sets of transitions, one for the required part ($R^\Box$), and one for the allowed part ($R^\Diamond$). So a model now consists of a single P1 part and two P2 parts over the same sets of states and actions. *Modal specifications* are such models satisfying $R^\Box \subseteq R^\Diamond$. Figure 2 shows two examples of modal specifications, for the respective parts of the measurement system considered in Fig. 1.

But what does such a modal specification mean or what does it represent? This question is addressed by the associated refinement notion $<$ defined in [37]. If a modal specification N refines a modal specification M (written $M{<}N$), then N is consistent with M, meaning that it is possible to implement both. In the extreme case, when $R^\Box = R^\Diamond$ in specification N and $M{<}N$, then N is consistent with M but cannot be refined any further up to bisimulation equivalence; it is therefore (a model of) an *implementation* of M. In particular the model of Figure 1(a) can now be considered to be an implementation, which is refining the modal specification from Figure 2(a), but is not refining the one from Figure 2(b) – since it has no ability to perform the required logging. Also, it is not hard to conclude that both

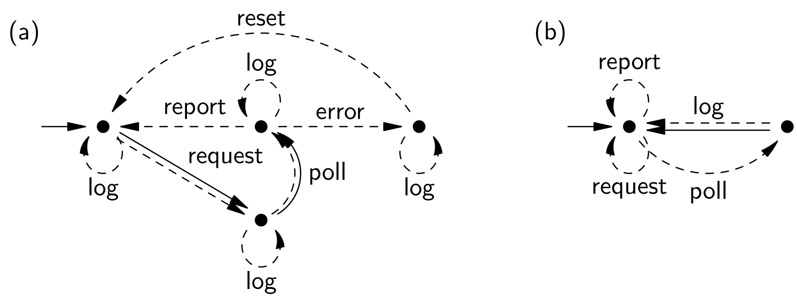

Figure 2: Two modal specifications of a measurement system: (a) basic polling and request behavior with allowed logs and error handling; (b) a log creation aspect. Throughout figures elements of required transitions ($R^{\Box}$) are denoted by solid lines, allowed transitions ($R^{\Diamond}$) by dashed lines.

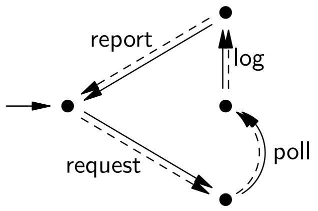

Figure 3: Common implementation of both modal specifications in Fig. 2.

models of Figure 2 are consistent with each other, i.e. it is possible to satisfy both requirements in a single implementation such as the one in Figure 3.

Mixed specifications were independently reintroduced by Dams [15, 16] as *mixed transition systems* in 1996 in the context of abstract interpretation. In mixed specifications we cannot assume that every required behavior is also allowed and so $R^{\Box} \subseteq R^{\Diamond}$ may not hold. Although this may seem strange, there are two good reasons for considering violations of that inclusion.

- Conflicting aspects of behavior may be expressed within a single specification, and so one would like to judge these conflicts as being irreconcilable if the specification has no implementations [33]. But $R^{\Box} \subseteq R^{\Diamond}$ always guarantees the existence of implementations, e.g. implement $R^{\Box}$ and discard $R^{\Diamond} \backslash R^{\Box}$.

- When a specification is the abstraction of a computer program, one can increase the precision of that abstraction so that it satisfies more properties that are true of the computer program. One way of doing this is to remove redundant elements of $R^{\Diamond}$ and to discover new elements of $R^{\Box}$ [15, 16].

In the remainder of the paper we will provide a more formal discussion of modal and mixed specifications and refinement; present the decision problems of specifications in the context of consistency, model variants, and property validation; discuss what is known about the complexity of these decision problems; and feature some applications and extensions of these decision problems. The selection of this material is subjective, so we also provide some references – again, subjectively selected – for further recommended reading on topics not covered in this paper.

2 Basics

Let us now formally define modal and mixed specifications and their refinement. As we shall also discuss variations of these models that include atomic propositions (specifications used in [15, 16] as program abstractions), we will add a third part out of which specifications may be built:

P3 a labeling function $L\colon S \to \mathbb{P}(\mathsf{AP})$ that determines which atomic propositions $q \in \mathsf{AP}$ are true at what states s, the subset $L(s)$ of AP.

A complete specification will comprise an P1 part, two P2 parts, and two P3 parts:

Definition 1. *Let Σ be a non-empty set of actions and AP a non-empty set of atomic propositions. A* mixed specification *M is a tuple $(S, R^{\Box}, R^{\Diamond}, L^{\Box}, L^{\Diamond})$ such that $(S, R^{\Box}, L^{\Box})$ and $(S, R^{\Diamond}, L^{\Diamond})$ are specifications as defined in P1-P3 above. In particular, $R^{\Box}$, $R^{\Diamond}$ are subsets of $S \times \Sigma \times S$, and $L^{\Box}$, $L^{\Diamond}$ are members of $S \to \mathbb{P}(\mathsf{AP})$.*

Thus a mixed specification consists of two separate parts: $(S, R^{\Box}, L^{\Box})$ specifies the *required* propositions and behavior over states in S, while $(S, R^{\Diamond}, L^{\Diamond})$ expresses *allowed* propositions and behavior over S. Throughout figures depicting mixed specifications in this paper, at state s, $!q$ denotes $q \in L^{\Box}(s)$, and $?q$ denotes $q \in L^{\Diamond}(s)$. Figure 4 shows three mixed specification with $\mathsf{AP} = \{x_{odd}, y_{odd}\}$ and a single action (whose labels are therefore omitted).

Example 1.

1. *A specification $(S, R^{\Box}, L^{\Box})$ of required properties and behavior can be made into a modal specification M by defining $R^{\Diamond} = S \times \Sigma \times S$ and setting $L^{\Diamond}(s) = \mathsf{AP}$ for all $s \in S$. This modal specification retains the required part and allows any property or behavior at any state.*

2. *A specification $(S, R^{\Diamond}, L^{\Diamond})$ of allowed properties and behavior can be made into a modal specification M by defining $R^{\Box} = \{\}$ and setting $L^{\Box}(s) = \{\}$ for all $s \in S$. This modal specification retains the allowed part and does not insist on any property or behavior at any state.*

Given a mixed specification, such as those constructed in Example 1, Fig. 2, or Fig. 4 we need to be able to determine whether an implementation satisfies that mixed specification. In particular, we need to determine what implementations are. Of course, genuine implementations will have structure and aspects that are not captured in mixed specifications (e.g. non-functional requirements). So implementations will have to be abstractions of real systems. Still implementations should be completely specified, which intuitively means that they contain no optional behavior and no optional propositions in states. The specifications consisting of one P1, one P2, and one P3 part will serve us nicely in that respect. The appropriateness of that choice of implementations is corroborated by the fact that implementations (S, R, L) are simply another representation of mixed specifications M whose required and allowed part are equal, i.e. $M = (S, R, R, L, L)$.

Returning to the question of whether an implementation satisfies (that is to say "implements") a mixed specification, the answer is traditionally given through an implementation relation. The refinement relation in [37] stipulates which specifications refine which specifications, but since our implementations are just special specifications this refinement serves as an implementation relation at the same time. Intuitively, refinement states that all required behavior has to be preserved in refining states, and that refining states can only exhibit allowed behavior that is also allowed in the states that they refine. Additionally, these demands are applied co-inductively to successor states. Formally:

Definition 2. *Let $M = (S_M, R^{\square}_M, R^{\diamond}_M, L^{\square}_M, L^{\diamond}_M)$ and $N = (S_N, R^{\square}_N, R^{\diamond}_N, L^{\square}_N, L^{\diamond}_N)$ be two mixed specifications and $(s_0, t_0) \in S_M \times S_N$. Then t_0* refines *s_0 iff there is a relation $Q \subseteq S_M \times S_N$ such that $(s_0, t_0) \in Q$, and whenever $(s, t) \in Q$ then*

1. *for all $(s, \alpha, s') \in R^{\square}_M$ there is some $(t, \alpha, t') \in R^{\square}_N$ with $(s', t') \in Q$,*
2. *for all $(t, \alpha, t') \in R^{\diamond}_N$ there is some $(s, \alpha, s') \in R^{\diamond}_M$ with $(s', t') \in Q$,*
3. *$L^{\square}_M(s) \subseteq L^{\square}_N(t)$, and*
4. *$L^{\diamond}_N(t) \subseteq L^{\diamond}_M(s)$.*

We write $s_0 \prec t_0$ whenever t_0 refines s_0 — or $(M, s_0) \prec (N, t_0)$ if we wish to emphasize the mixed specifications.

Example 2 ([27])**.** *The three mixed specifications M, N_1, and N_2 in Figure 4 consist of a sequence of three states and a self-loop on the last state. There is only a single action symbol which is therefore omitted. The set AP is $\{x_{odd}, y_{odd}\}$. If $\mathtt{x}$ and $\mathtt{y}$ are integer variables, and if x_{odd} is true at a state if the value of $\mathtt{x}$ is odd, and similarly for y_{odd}, then the specification M is an abstraction of the state space for the following program:*

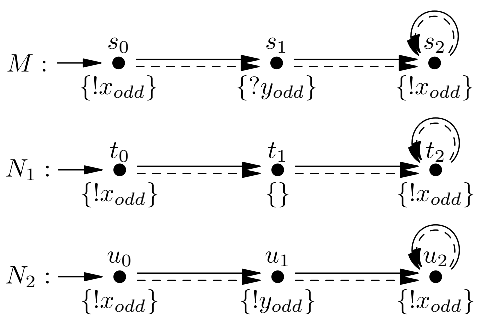

Figure 4: Three modal specifications with a single action (omitted) and two atomic propositions x_{odd} and y_{odd}. State s_0 is refined by state t_0 and by state u_0; but t_0 does not refine u_0, and u_0 does not refine t_0.

```
(x = 1 || y = 0);
(x = 2*f(x) || y = f(y));
(x = 1 || y = 0);
```

where f is an unspecified function of type `int → int` *and* `c1 || c2` *denotes the parallel execution of commands* `c1` *and* `c2`*. The specifications N_1 and N_2 show two possible implementations of M, corresponding to whether y_{odd} at state s_1 is refined to being false or true. Thus we have $(M, s_0) \prec (N_1, t_0)$ and $(M, s_0) \prec (N_2, u_0)$; but neither $(N_1, t_0) \prec (N_2, u_0)$ nor $(N_2, u_0) \prec (N_1, t_0)$ hold.*

With a refinement notion in hand we can now define the set of implementations of a mixed specification: Let M be a mixed specification with state s_0. Then $I(M, s_0)$, the set of *implementations of* (M, s_0), is defined to contain exactly those pairs of implementations and states (J, j) such that $(M, s_0) \prec (J, j)$. For example, in Fig. 4 (N_1, t_0) and (N_2, u_0) are in $I(M, s_0)$, but (N_2, u_0) is not in $I(N_1, t_0)$.

It is not difficult to see that $\prec$ is a transitive relation. So if $(M, s_0) \prec (N, t_0)$, then $I(N, t_0) \subseteq I(M, s_0)$. The converse is, surprisingly, false. Figure 5 shows two modal specifications that have the same set of implementations but where one does not refine the other. This is why refinement is sometimes referred to as "modal" refinement and why the relationship $I(N, t_0) \subseteq I(M, s_0)$ is called the "thorough" refinement (e.g. in [36]). So thorough refinement is defined as an inclusion of sets of implementations, such that one specification refines the other if every implementation of the former is also an implementation of the latter.

Incidentally, modal refinement is strikingly similar to the two-player case [2] of alternating simulation [3], a game-theoretic notion of refinement. In alternating simulation inputs are treated as uncontrollable actions, while outputs are consid-

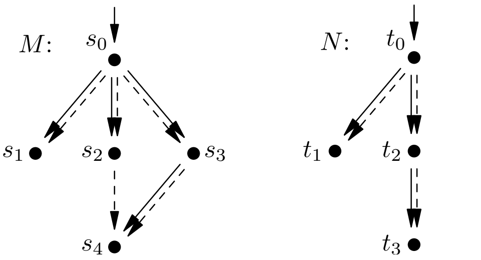

Figure 5: (Due to Harald Fecher.) Two mixed specifications M and N with $I(M, s_0) = I(N, t_0)$ but where (N, t_0) is not refined by (M, s_0).

ered to be controllable. Similarly, in modal refinement determinstic implementations have no control on providing required transitions, while they have a certain freedom of choosing allowed transitions in modal refinement. In [35] it has been argued that modal refinement is strictly more expressive than alternating simulation for deterministic input/output games, since the latter are a special case of the former. In [35] a selection of weak (observational) modal refinements are discussed with the agenda of aligning modal refinement better with alternating simulation in the nondeterministic case as well.

Another form of incompleteness of modal refinement occurs in the context of composition operators of modal and mixed specifications, e.g. parallel composition. Given a set of inference rules that define such a composition operator for implementations, for instance the CCS [41] parallel composition $||$, one can obtain a new operator $||_m$ for modal specifications by simply doubling every inference rule for $||$ in a *may* ($R^\diamond$) and a *must* ($R^\Box$) version. This can be seen in Figure 6, which illustrates the definition of the synchronization rule for modal specifications. Assuming that refinement is a precongruence with regards to $||$ – meaning that $M_i \prec N_i$ for $i = 1, 2$ implies $(M_1 \,||\, N_1) \prec (M_2 || N_2)$ – one can expose an incompleteness of the corresponding composition operator $||_m$ for modal specifications.

Firstly, note that $||$ and $||_m$, when restricted to implementations are the same operators. Secondly, a reasonable expectation is that the set of implementations of $M \,||_m\, N$ would equal the set of implementations $I \,||\, J$ with I an implementation of M, and J an implementation of N. In short, any implementation of a composed specification should be representable as the composition of component implementations, and the composition of component implementations should be an implementation of the composed specification. The latter holds since refinement is a precongruence for $||_m$, so we know that $I \,||\, J = I \,||_m\, J$ is an implementation of $M \,||_m\, N$ for all $I \in I(M)$ and $J \in I(N)$. But the converse is false: there are implementations of the composed specifications that are not representable as

$$\frac{N \xrightarrow{a} N' \qquad M \xrightarrow{a} M'}{N \,||\, M \xrightarrow{a} N' \,||\, M''} \quad \cdots \qquad \begin{array}{c} \dfrac{N \xrightarrow{a}_{\Box} N' \qquad M \xrightarrow{a}_{\Box} M'}{N \,||_m\, M \xrightarrow{a}_{\Box} N' \,||_m\, M'} \quad \cdots \\[2em] \dfrac{N \overset{a}{\dashrightarrow}_{\Diamond} N' \qquad M \overset{a}{\dashrightarrow}_{\Diamond} M'}{N \,||_m\, M \overset{a}{\dashrightarrow}_{\Diamond} N \,||_m\, M'} \quad \cdots \end{array} \tag{1}$$

Figure 6: Structural operational semantics for synchronous parallel composition operator. On the left: a sole composition rule for labeled transition systems. On the right: two copies of that very composition rule for modal transition systems, using respectively *must* and *may* transitions. The parallel composition of two modal specifications using $||_m$ has a *must* transition if both specifications require it; equally, the parallel composition has a *may* transition if both specifications allow it.

compositions of component implementations, see Figure 7. In fact it is not even known, whether the following problem is decidable: establish if a given implementation of a composition of two modal specifications can be implemented by a composition of implementations of these specifications.

3 Decision Problems

Suppose a modeler has provided a set of mixed specifications $\{M_i \mid 0 < i \leq k\}$ with $k \geq 1$, where s_i is a designated initial state for M_i and each (M_i, s_i) models a particular aspect of a system. A fundamental question is whether these aspects can interact and co-exist such that they are all realizable in a single implementation. We can phrase this question in terms of our refinement notion.

CI A finite set of mixed specifications with initial states $\{(M_i, s_i) \mid 0 < i \leq k\}$ with $k \geq 1$ is said to have a *common implementation* iff there is an implementation (J, j) such that $(M_i, s_i) \prec (J, j)$ for all $0 < i \leq k$.

Figure 2(a) shows (M_1, s_1) with initial state s_1, Figure 2(b) shows (M_2, s_2) with initial state s_2, and Figure 3 depicts a common implementation (J, j) of the two.

A positive answer to the common implementation problem verifies the consistency of finitely many models of a system. Observe that we may equally well use the common implementation problem to investigate or expose flaws in a system model. Suppose that the mixed specification (M, s_0) is a desired model of a system and (F, t_0) is a mixed specification that models a particular flaw. Now, if we can show that there does not exist any common implementation of both mixed

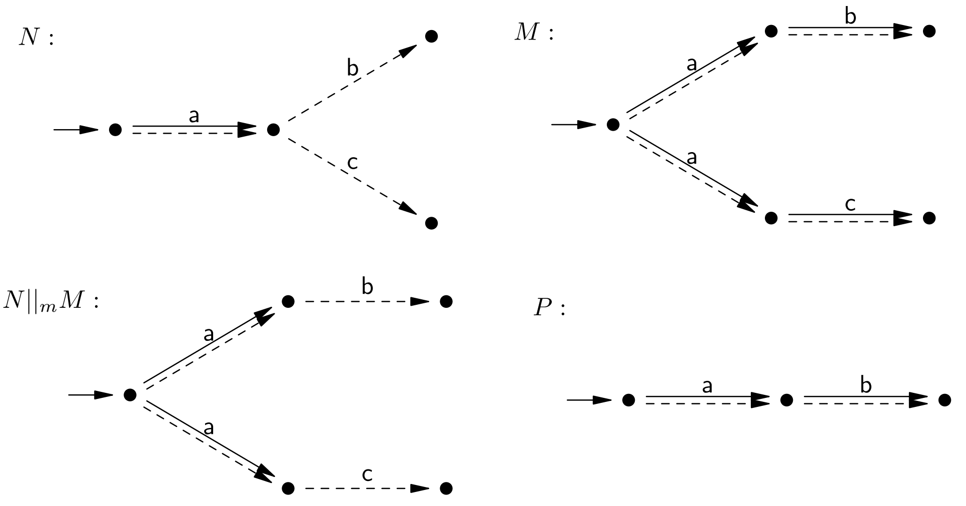

Figure 7: Illustration of the incompleteness of parallel composition $\|_m$ for modal specifications: M and N are modal specifications, $N \|_m M$ is their modal parallel composition. But P is a legal implementation of $N \|_m M$ that does not arise as a parallel composition $I \| J$ of implementations of N and M, respectively.

specifications, then the model is inconsistent with the flaw. Consequently, any system that implements this model cannot exhibit the respective error. Conversely, a common implementation of (M, s_0) and (F, t_0) may provide important clues on how the mixed specification M needs to be revised in order to eliminate the flaw.

One often considers a special case of the common implementation problem, namely for $k = 1$, which asks whether (M_0, s_0) has any implementations, or, in other words, whether (M_0, s_0) is *consistent*:

C A mixed specification M with initial state s_0 is *consistent* iff $I(M, s_0)$ is non-empty, i.e., iff there exists an implementation (J, j) with $(M, s_0) \prec (J, j)$.

Not every mixed specification is consistent. Figure 8 shows a mixed specification that does not have any implementations.

There is a simple static way in which one can ensure that a mixed specification $M = (S, R^\Box, R^\Diamond, L^\Box, L^\Diamond)$ is consistent, by demanding that everything that is required is also allowed. This is achieved through the inclusions

$$R^\Box \subseteq R^\Diamond \tag{2}$$

$$L^\Box(s) \subseteq L^\Diamond(s) \qquad (\forall s \in S) \tag{3}$$

A mixed specification $M = (S, R^\Box, R^\Diamond, L^\Box, L^\Diamond)$ is also called a *modal specification* if it satisfies all inclusions in (2–3). It is easy to see that a modal specification $M =$

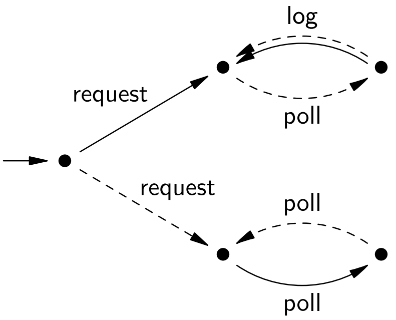

Figure 8: A mixed specification that is not consistent.

$(S, R^{\square}, R^{\diamond}, L^{\square}, L^{\diamond})$ is consistent at all of its states. For $s \in S$, one implementation of (M, s) is $((S, R^{\square}, R^{\square}, L^{\square}, L^{\square}), s)$.

The common implementation problem CI and the consistency problem C correspond to satisfiability problems, as they appeal to an existential quantification over implementations. But there is also a natural decision problem for mixed specifications that corresponds to a validity problem.

TR A mixed specification M with state s_0 is thoroughly refined by a mixed specification N with state t_0 iff $I(N, t_0) \subseteq I(M, s_0)$.

We already saw one technique for establishing positive instances of TR: to show $I(N, t_0) \subseteq I(M, s_0)$ show that $(M, s_0) \prec (N, t_0)$ holds. We also saw that this technique cannot be used for showing negative instances of thorough refinement $I(N, t_0) \not\subseteq I(M, s_0)$ as the non-existence of any refinement relation between s_0 and t_0 says nothing about the relationship of $I(M, s_0)$ and $I(N, t_0)$ in general, as demonstrated in Figure 5. Consequently, direct algorithms are needed for TR in order to determine with certainty whether all implementations of one mixed specification are also implementations of another mixed specification.

The last decision problem considered here comes in a satisfiability and a validity flavor. Suppose a mixed specification M with initial state s_0 models a system. We want to validate or explore properties of that system, expressible in some temporal logic, through the validation or exploration of the mixed specification M. If φ is such a property, we can ask – as originally proposed in [11] – whether there *exists* an implementation of M that satisfies φ.

GMC Given a mixed specification M with initial state s_0 and a temporal logic formula φ, the *generalized model checking* problem asks whether there is an implementation (J, j) of (M, s_0) that satisfies φ.

Logics typically considered for model checking include linear-time temporal logics such as LTL, and branching-time temporal logics such as CTL and the modal mu-calculus (see e.g. [21]). For these logics GMC can be seen as a kind of restricted satisfiability problem, which establishes whether properties φ are consistent with a mixed specification. We can turn GMC into a validity problem in the usual fashion: decide GMC for $\neg\varphi$ and (M, s_0). If the answer is negative, then *all* implementations of (M, s_0) satisfy φ. We study issues associated with such validation in more detail in Section 7.

4 Complexity

Having presented the essential decision problems CI, C, TR, and GMC we shall now discuss their computational complexity. Let us start by introducing special classes of specifications, used in the determination of lower bounds.

A modal specification $(S, R^{\Box}, R^{\Diamond}, L^{\Box}, L^{\Diamond})$ with $R^{\Box} = R^{\Diamond}$ and Σ being a singleton is called a *partial Kripke structure*. All three modal specifications in Figure 4 are examples of partial Kripke structures. As shown in [30], this definition of partial Kripke structures is equivalent to the one that introduced these models in the literature [10]. In this manner, the implementations of partial Kripke structures correspond to the familiar Kripke structures, and all partial Kripke structures are consistent.

A mixed (resp. modal) specification $(S, R^{\Box}, R^{\Diamond}, L^{\Box}, L^{\Diamond})$ with empty set AP is called a mixed (resp. modal) *transition system*, and abbreviated as $(S, R^{\Box}, R^{\Diamond})$. Their implementations are called labeled transition systems, and often denoted simply by a pair (S, R). Figure 2 shows modal transition systems and Fig. 3 depicts a labeled transition system.

We will use partial Kripke structures for a discussion of lower bounds for generalized model checking, and modal and mixed transition systems for a discussion of lower bounds for the other three decision problems. That choice simply echoes the respective choices made in the extant literature [11, 6].

Lower bound for CI for modal specifications. We establish a lower bound for the common implementation problem for *modal* transition systems. This lower bound will therefore also apply to modal and mixed specifications in general. We record that the size $|M|$ of a mixed specification $M = (S, R^{\Box}, R^{\Diamond}, L^{\Box}, L^{\Diamond})$ with finite state set S is defined as

$$|S| + |R^{\Box} \cup R^{\Diamond}| + \sum_{s \in S} |L^{\Box}(s) \cup L^{\Diamond}(s)| \tag{4}$$

Generalized Geography (GENGEO) is known to be a PSPACE-complete decision problem [24, 32]. An instance of GENGEO considers a directed graph G with a finite set of vertices V, an edge relation $E \subseteq V \times V$, and a designated vertex v_α. Two players, Verifier and Refuter, play a game on G where each play of the game results in a finite sequence $(v_0, v_1)(v_1, v_2) \dots (v_{n-2}, v_{n-1})(v_{n-1}, v_n)$ of non-repeating edges (v_i, v_{i+1}) from E, determined as follows:

1. Initially, Verifier chooses an edge $(v_0, v_1) \in E$ with $v_0 = v_\alpha$.
2. Refuter chooses an edge $(v_1, v_2) \in E$ not chosen already in that play.
3. Verifier chooses an edge $(v_2, v_3) \in E$ not chosen already in that play.
4. Refuter and Verifier strictly alternate their moves in this fashion until a player faces a vertex v_n that has no outgoing edge or only outgoing edges that were already chosen in that play. Then that player looses that play.

The decision problem for G is whether Verifier has a strategy that, if followed by her, will ensure her that she wins all plays in that game.

Example 3 (quoted after [53])**.** *Consider the graph G with vertices v_0, ..., v_8 shown in Figure 9. Verifier has a winning strategy from vertex v_0 in G:*

- *Clearly, the initial move of Verifier has to be from v_0 to either v_1 or v_2.*
- *If Verifier were to move to v_2, Refuter could then move to v_8 and so Verifier would lose. So Verifier moves from v_0 to v_1.*
- *Refuter moves to v_3 as there is only one edge with v_1 as source.*
- *Verifier then moves to v_4, leaving Refuter only moves to either v_2 or v_6.*
- *Regardless of Refuter's next move, Verifier can then move to v_8 and wins.*

In order to reduce GENGEO to CI, for a given graph G and initial state v_α we construct a finite set of modal transition systems such that

- the sum of their sizes is polynomial in the size of G, and
- they have a common implementation iff Verifier has a winning strategy in the GENGEO game for G from v_α.

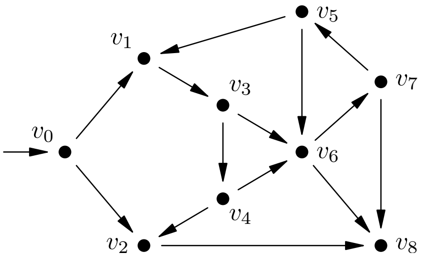

Figure 9: An instance of GENGEO with a winning strategy for the Verifier, an edited version of the example found in [53].

The construction is such that a winning strategy can be translated with relative ease into a common implementation of these modal transition systems. The harder part of the reduction proof is, for a given common implementation of these modal transition systems, to synthesize a winning strategy for GENGEO in G? One may think of the former as a soundness and the latter as a completeness argument.

The intuition behind this reduction is that transitions in $R^{\Box}$ are used to encode all possible game moves of Refuter, forcing an implementation to consider every choice of Refuter in that game. Dually, choices of Verifier are encoded in $R^{\Diamond}$ and so any implementation is allowed to make such a choice for Verifier. The less straightforward part is the design of additional modal transition systems that act as "monitors" and ensure that no responses may be repeated, and that Verifier must always make some response when it is his turn. This reduction therefore establishes PSPACE-hardness of CI for modal transition systems.

Example 4. *Figures 10-12 show the set of modal transition systems whose CI instance the instance G of* GENGEO *reduces to. Unlabeled transitions have an implicit label $\pi \notin E$, transitions labeled with sets of labels denote one such transition for each such label.*

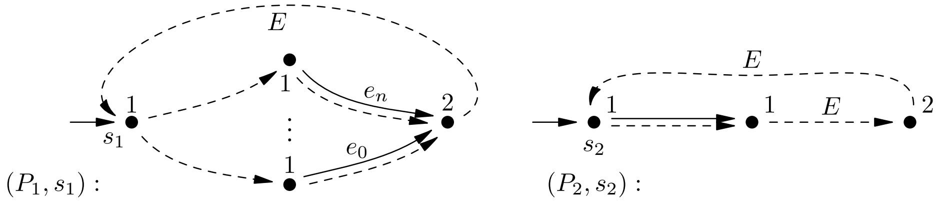

Figure 10: Modal specifications (P_1, s_1) and (P_2, s_2) together ensure that Verifier can always continue to play. Assume $E = \{e_0, \dots e_n\}$; a state tag 1 (resp. 2) indicates that Verifier (resp. Refuter) can make a next move. All unlabeled transitions are labeled by an implicit π label.

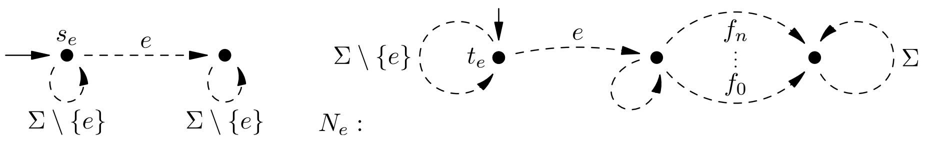

Figure 11: Specifications M_e, N_e instantiated for each $e \in E$ and the set $\{f_0, \ldots, f_n\}$ of edges whose source is the target of e. Specification M_e ensures that edge e is visited at most once; N_e ensures that only edges that follow e directly will be visited immediately after a visit of e.

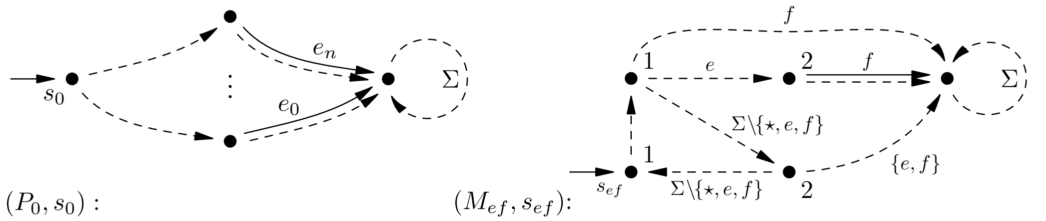

Figure 12: Specification P_0 ensures that Verifier begins by choosing one of the edges $\{e_0, \ldots, e_n\}$, whose source is v_0. For each pair of adjacent edges e and f we instantiate a specification M_{ef}, which ensures that CI explores for all moves e those responses f whose sources are the target of e, unless e was already chosen or f was already chosen by Refuter.

Lower bound for C for mixed specifications. We exploit the above PSPACE-hardness result to reduce CI for modal transition systems to C for mixed transition systems. Figure 13 shows the construction for a finite set of modal transition systems $\{(M_i, s_i) \mid 0 < i \leq k\}$ with $k \geq 1$. The size of that mixed transition system is quadratic in the size of the instance of CI. Therefore it suffices to show that the mixed transition system (M, c_k) has an implementation iff the instance of CI has a common implementation. Given a common implementation (J, j) of all (M_i, s_i) we can replace each (M_i, s_i) in (M, c_k) with (J, j) and implement all outgoing transitions from all states c_i. It is easily seen that this provides an implementation of (M, c_k). Conversely, let $(\hat{J}, \hat{j})$ be an implementation of (M, c_k). Then the pattern of required and allowed transitions on all paths from c_k to any s_i in M is such that the relation witnessing $(M, c_k) \prec (\hat{J}, \hat{j})$ enforces existence of a state j in $\hat{J}$ such that $(\hat{J}, j)$ is a common refinement – and so a common implementation – of all (M_i, s_i).

Combining the reduction of GENGEO to CI for modal transition systems with the reduction of the latter to C for mixed transition systems implies that C for mixed transition systems is PSPACE-hard.

Lower bound for TR for mixed specifications. With PSPACE-hardness of C for mixed specifications at hand, we can prove PSPACE-hardness of TR for mixed

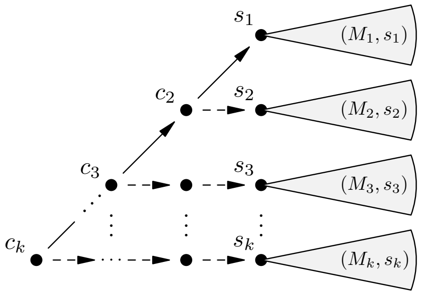

Figure 13: A mixed specification that conjoins mixed specifications M_i with initial states s_i together. All outgoing transitions from all c_i have labels with a special action that does not occur in the action set for any M_i.

specifications as follows: Given a mixed transition system M with initial state s, we claim that (M, s) is consistent iff not $I(N, t) \supseteq I(\hat{M}, \hat{s})$, where N is the labeled transition system with state set $\{t\}$ and no transitions whatsoever, and $\hat{M}$ extends M with a fresh state $\hat{s}$ and a single transition $(\hat{s}, \pi, s) \in R^{\diamond} \setminus R^{\Box}$ where π is an action not present in M.

First, if (M, s) is consistent, any implementation (J, j) thereof results immediately into an implementation of $(\hat{M}, \hat{s})$ by adding a fresh state $\hat{j}$ with a transition $(\hat{j}, \pi, j)$ to J matching $(\hat{s}, \pi, s)$. But this is then not an implementation of (N, t) and so $I(N, t) \not\supseteq I(\hat{M}, \hat{s})$.

Second, from $I(N, t) \not\supseteq I(\hat{M}, \hat{s})$ one gets an implementation $(\hat{J}, \hat{j})$ of $(\hat{M}, \hat{s})$ that is not an implementation of (N, t). From the latter we infer the existence of a transition out of $\hat{j}$ to some state j in $\hat{J}$, which then has to match $(\hat{s}, \pi, s)$ as the only outgoing transition from $\hat{s}$ in $\hat{M}$. But then the co-inductive nature of refinement tells us that $(\hat{J}, j)$ is an implementation of (M, s) and so the latter is consistent.

Lower bound for TR for modal specifications. The PSPACE-hardness argument for TR with mixed specifications does not apply directly to TR for *modal* specifications. That reduction required M to be a mixed specification since C for modal specifications is a trivial problem. Therefore we require a bespoke argument for the lower bound of TR for modal specifications, which we now provide for modal transition systems.

This reduction exploits the fact that deciding the truth of QCNF formulae, a closed quantified Boolean formulae with propositional kernel in 3CNF form, is PSPACE-complete [24, pp. 171-2]. Consider, e.g., the following formula

$$\varphi_Q = \forall x\, \exists y\, (\neg x \lor y) \land (\neg y \lor x) \tag{5}$$

This formula reads that for every Boolean value x, there exists a Boolean value y such that x is true iff y is true. Clearly, φ_Q is true. One way of determining

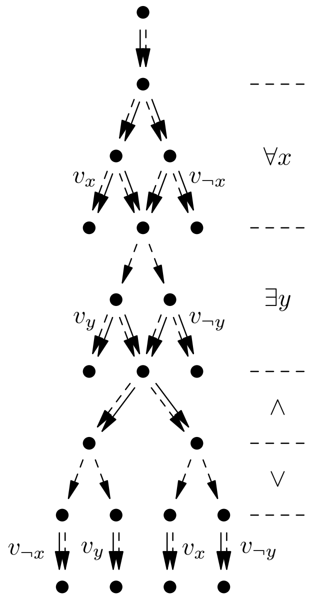

Figure 14: Modal transition system N_{φ_Q} representing the evaluation game graph of the formula in (5).

this is to turn φ_Q into a DAG with branches for choices of quantifiers, on which Verifier and Refuter play a strictly alternating game, and where truth corresponds to the existence of a winning strategy for Verifier. This approach will not work as a reduction, though, since the game graph of formulae φ may be exponential in the size of φ. Another problem is how to relate a winning strategy in this game to an instance of thorough refinement for modal transition systems. Remember that TR involves two and not just one graph-like object.

The first problem is overcome by letting a decision point for variable *x* branch to two successors which then do two things:

- they merge to a common state for the next layer of decisions (thus avoiding the exponential blow-up), but
- they also provide "spike transitions" to dead end states that can act as "stores" for values chosen for variable *x*.

Universally quantified *x* first branch out with required transitions, existentially quantified *y* first branch out with allowed transitions. Conjunction, disjunction, and literals are then modeled accordingly. For formula φ, this specifies a modal transition system N_φ with initial state s_φ, shown in Figure 14 for φ_Q.

The second problem is overcome by thinking of the implementations of (N_φ, s_φ) as strategies that may win and so witness truth of φ, and to devise another

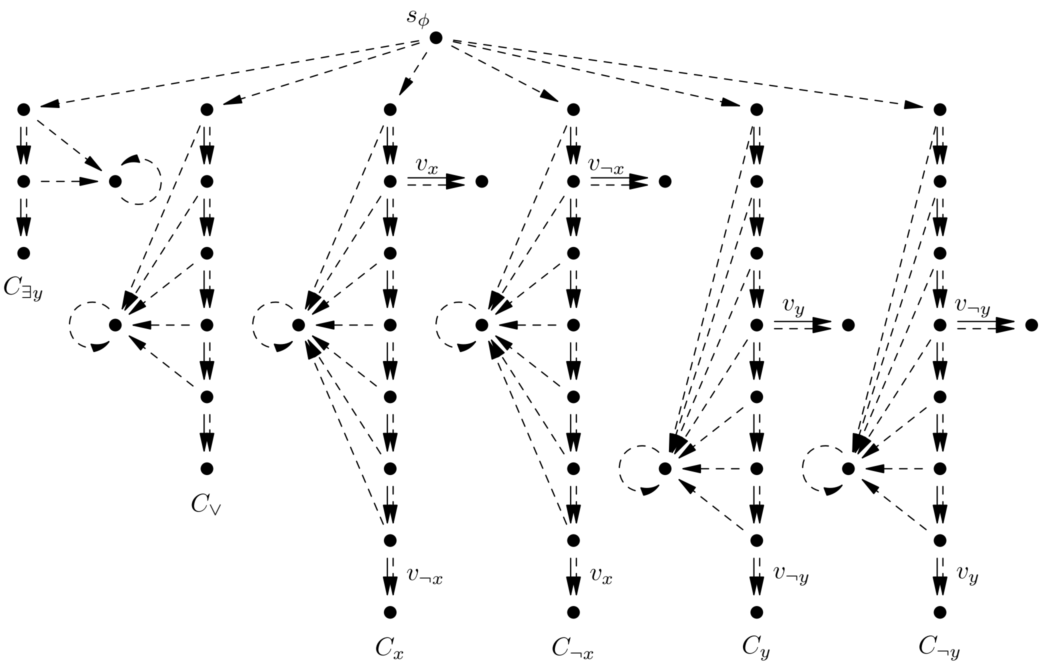

Figure 15: Modal transition system M_{φ_Q} representing all flawed evaluations of the formula in (5) in a game semantics.

modal transition system M_φ with initial state t_φ such that the implementations of (M_φ, t_φ) correspond to, and cover all, non-winning strategies for witnessing the truth of φ. We can then say that formula φ is true iff $I(N_\varphi, t_\varphi) \not\subseteq I(M_\varphi, s_\varphi)$, i.e., iff there is a strategy that is not loosing and so winning.

The construction of M_φ is somewhat involved as it needs to capture all possible failures of potential strategies for the truth of φ in a game semantics, given in the form of implementations of (M_φ, s_φ). Figure 15 illustrates this for formula (5). Each of the columns represent a certain category of erroneous implementations. The left most column, for instance represents all those implementations that are so short that they do not select a value for the existentially quantified variable y.

Since $|M_\varphi|$ and $|N_\varphi|$ are polynomial in the size of φ, we obtain the desired PSPACE-hardness of TR for modal transition systems.

Lower bound for GMC for partial Kripke structures The computational complexity of GMC for such models is a function of the size of the model M and of the size of the formula φ for the temporal logic that φ belongs to. Here we only consider the complexity in terms of the size of φ, and only for the branching-time logics CTL and modal mu-calculus. As pointed out in [11], generalized model checking is a generalization of both model checking and satisfiability checking.

To see how GMC generalizes model checking, note that any instance of GMC for an implementation J and property φ simply asks whether the model check of φ on J is true. This is so since any refinement between implementations is just bisimulation, and bisimilar models satisfy the same formulae of CTL and of the modal mu-calculus.

To appreciate that GMC generalizes satisfiability checking, consider the partial Kripke structure $M_\perp$ with state set $\{s_\perp\}$, transition relation $\{(s_\perp, s_\perp)\}$, and labeling functions satisfying $L^\Box(s_\perp) = \{\}$ and $L^\diamond(s_\perp) = \mathsf{AP}$. It is straightforward to argue that *all* states of all partial Kripke structures with set of atomic propositions AP refine $(M_\perp, s_\perp)$. In particular, $I(M_\perp, s_\perp)$ is the set of all partial Kripke structures over AP. Thus the satisfiability of φ over partial Kripke structures is equivalent to a positive answer to GMC applied to $(M_\perp, s_\perp)$ and φ.

Since satisfiability for CTL, as well as for the modal mu-calculus, is EXPTIME-complete, it was inferred in [11] that GMC is EPXTIME-hard for both logics in the size of the formula.

We summarize the results on lower bounds in Table 1.

Table 1: Tabular summary of the lower bounds provided.

	Modal specifications	Mixed specifications
Common implementation	PSPACE-hard	PSPACE-hard
Consistency	trivial	PSPACE-hard
Thorough refinement	PSPACE-hard	PSPACE-hard
GMC of CTL and mu-calculus	EXPTIME-hard	EXPTIME-hard

Upper bounds. One can show that the decision problems CI, C, and TR are in EXPTIME. For mixed specification M, we associate with each of its states s a recursion variable X_s and a system of recursive equations, one for each $s \in S$ [33]:

$$X_s = \Big(\bigwedge_{(s,\alpha,s')\in R^\Box} \langle\alpha\rangle X_{s'} \Big) \wedge \Big(\bigwedge_{\alpha\in\Sigma} [\alpha]\Big(\bigvee_{(s,\alpha,s')\in R^\diamond} X'_s \Big)\Big) \wedge \Big(\bigwedge L^\Box(s) \Big) \wedge \neg \Big(\bigvee (\mathsf{AP} \setminus L^\diamond(s)) \Big) \quad (6)$$

Each X_s denotes a subset of S and one solves (6) simultaneously for all s such that each solution set X_s is maximal. This set X_s then contains exactly those states t of M for which $(M, s) \prec (M, t)$. One can also unwind this recursion syntactically in the modal mu-calculus so that the solution set X_s is represented by a *characteristic formula* [1] ψ_s with greatest fixed-points only. In that manner we can reduce CI and C to checking the satisfiability of $\bigwedge_{i=1}^{k} \psi_{s_i}$ and ψ_{s_0} (respectively). Similarly, TR reduces to checking the validity of $\neg\psi_{t_0} \to \psi_{s_0}$. Unfortunately, the size of each ψ_{s_i} may be exponential in $|M_i|$ and so this reduction places these decision problems into 2EXPTIME. To lower these upper bounds, one can translate the recursive equations for X_s compositionally into alternating tree automata A_s. The reductions to satisfiability and validity, as observed for characteristic formulae ψ_s above, still apply. But the size of A_s is linear in the size of $|M|$. This EXPTIME upper bound is folklore knowledge but does not seem to have been published prior to [6].

A similar upper bound for GMC can be obtained by converting φ into an equivalent alternating tree automata A_φ, a linear transformation in the size of φ, and noting that there is an implementation of (M, s) satisfying φ iff the alternating tree automata $A_s \wedge A_\varphi$, which accepts the intersection of the languages that are accepted by A_s and A_φ, has a non-empty language. Since non-emptiness is in EXPTIME and since intersection has linear overhead for these automata, we see that GMC is in EXPTIME in the size of φ for CTL and for the modal mu-calculus. That upper bound was originally given in [11].

5 Extensions

Historically modal transition systems [37] predate mixed transition systems [15], although the latter were introduced independently. As already stated, mixed transition systems extend modal transition systems by dropping the static consistency requirement, that $R^\Box \subseteq R^\Diamond$.

One of the first attempts to apply modal transition systems included the solution of equation systems [38, 54] involving bisimulation [43, 40] constraints with CCS-like contexts embedding an unknown process X. It turns out that the set of all solutions of such an equation system is characterized by a *disjunctive modal transition system*, which differs from a regular modal transition system by including must transitions which model disjunctions of target states and actions. Technically the must transition relation $R^\Box$ is a subset of $S \times 2^{\Sigma\times S}$. A given implementation has the choice of implementing at least one of the branches in the disjunction. The modal refinement relation has been extended accordingly [38].

Figure 16 shows a simple disjunctive modal transitions system, a more precise model of our measurement system from Section 1. The specification has one disjunctive *must* transition, which specifies that at least one of the two actions log and report must be implemented by the target state of the poll action. The complete specification thus ensures that after a poll it is optional to log the measured value before having to report it.

One-selecting modal transition systems, proposed by Schmidt and Fecher [48,

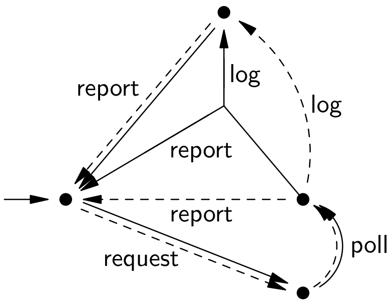

Figure 16: A disjunctive modal transition system modeling a measurement system with optional logging.

49, 47], are a strengthening of disjunctive modal transition systems and their refinement. More precisely one-selecting modal transition systems interpret disjunctive transitions according to an exclusive-or semantics such that exactly one (and not at least one) of the target states can be implemented. They also include disjunctive *may* transitions, which would not bring any additional expressive power to disjunctive modal transition systems, but do constitute a nontrivial extension. In [49] Schmidt and Fecher show that disjunctive modal transition systems and one-selecting modal transition systems can express the same sets of implementations, by supplying transformations between both kinds of models that preserve the respective sets of implementations. At the same time, they demonstrate that one-selecting modal transition systems have a strictly more expressive refinement preorder when compared to disjunctive modal transition systems. This follows from a proof that no transformation from one-selecting modal transition systems to disjunctive modal transition systems is monotone with respect to the respective refinement notions. Such a monotone transformation is possible in the other direction, though.

For example, in interpreting the specification of Figure 16 as a one-selecting modal transition system we decrease the allowed set of implementations. Now it is strictly specified that either reporting should be done immediately, or after performing a log. In the original interpretation as a disjunctive modal transition system, implementations were able to provide both choices.

In 1993 Čerāns and colleagues presented *timed modal specifications* – an extension of modal specification with real time [13], or, in other words, a generalization of real time process calculi to allow loose specifications. The theory was accompanied by a corresponding verification tool, EPSILON.

Modal transition systems and their modal refinement can be seen as an extension themselves, namely of *extended transition systems* and of their partial bisimulation [39, 52, 10]. Extended transition systems $(S, R, \uparrow)$ extend transition systems (S, R) with a divergence predicate $s \uparrow \alpha$ of type $S \times \Sigma$, saying that not all transitions with source s and labeled with α may be present. One can represent $(S, R, \uparrow)$ as a modal transition system by setting $R^{\diamond} = R$ and $R^{\square} = R \cup \{(s, \alpha, s') \in S \times \Sigma \times S \mid s \uparrow \alpha\}$. It is then a pleasant surprise that partial bisimulations between two extended transition systems are captured by modal refinement of their corresponding representations as modal transition systems. See [30] for details.

6 Applications in Software Engineering

In Section 2 we have mentioned the considerable overlap of expressive power between refinement and alternating simulation for deterministic 2-player games. Since alternating simulation and 2-player games are a popular base to model rich behavioral interfaces for components [2], it seems natural to consider building interface theories around modal specifications and refinement [35, 7, 44]. Modal specifications bring their usual advantage to interface theories, allowing to model required and optional ways of interacting with a component. Alternating simulation enforces a more rigid structure than modal specifications in this case, by making all input steps required transitions, and output steps allowed transitions. In the context of interface theories the question of completeness of refinement and complexity of thorough refinement has been less pressing so far, since only deterministic and consistent interfaces were proposed (refinement is complete for such modal specifications). When multiple interfaces are associated with a single component [7, 44] in order to express multiple viewpoints, the common implementation question seems particularly relevant.

An application, which is closely related to interfaces, is modeling *variability* in families of software products. In *software product line* engineering one considers *configurable* components, which may exhibit different behavior, depending on configuration. As already argued modal specifications fit well in this usage scenario, as they are naturally equipped with means of expressing variation in behavior using may transitions, which are not required [23, 35, 22]. For example the specification of Figure 2(a) can be interpreted as a model of a family of measurement systems, with some of them being able to do error handling, and some not providing this functionality. Questions that naturally arise with these specifications are TR (corresponding to partial specialization of a configurable component), CI (whether there exist configurations of components which make them compatible), and GMC (whether all implementations guarantee safety properties, and whether there exist implementations fulfilling some requirements).

The line of work on *behavioral model merging* (e.g. [51, 9, 50]) considers a very interesting problem of combining partial models constructed by multiple stakeholders, for example during requirements engineering. The basic choice here is that merging of several models should result in a common refinement of all of them, which is minimal with respect to $\prec$. It turns out that in general a minimum common refinement is not uniquely defined [29]. As CI is trivially reducible to the common refinement problem, we note that the latter must be at least PSPACE-hard (and indeed EXPTIME-complete as we will explain later on). Nonetheless, a heuristic for merging models may construct an informative common refinement that can aid system comprehension. Also, in [50] an algorithm is presented that synthesizes from a safety property φ, expressible in a 3-valued variant of linear-time temporal logic (whose "fluents" allow state-based model checking for event-based models), a modal transition system $M(\varphi)$ such that the modal transition systems that satisfy φ in a 3-valued semantics defined in loc. cit. are exactly those modal transition systems that refine $M(\varphi)$. So $M(\varphi)$ can be seen as a characteristic model for φ. The MTSA tool provides functionality for model merging and synthesis, and is available as an Eclipse plug-in [19].

Early on a verification system called TAV [12, 8] was developed for modal transition systems. The TAV system supported compositional modeling and efficient checking of refinement using a local verification technique [34] that avoids building the complete state space.

7 Model Checking and Semantic Minimization

A mixed or modal specification represents all of its implementations, and can therefore satisfy properties in at least two ways:

- $\mathsf{Sat}\,(M, s, \varphi)$ holds iff there is an implementation (J, j) of (M, s) satisfying φ,
- $\mathsf{Val}\,(M, s, \varphi)$ holds iff all implementations (J, j) of (M, s) satisfy φ.

The second judgment illustrates another need for consistency checking: if (M, s) is inconsistent, then $\mathsf{Val}\,(M, s, \varphi)$ is true for all φ! Therefore, a consistency check of (M, s) is needed to rule out or to detect such undesired vacuity. The duality, that $\mathsf{Sat}\,(M, s, \varphi)$ holds iff $\mathsf{Val}\,(M, s, \neg\varphi)$ does not, remains to be true even for inconsistent specifications (M, s).

Of course, $\mathsf{Sat}\,(M, s, \varphi)$ is just asking whether GMC holds for (M, s) and φ. So both judgments $\mathsf{Sat}\,(M, s, \varphi)$ and $\mathsf{Val}\,(M, s, \varphi)$ are EXPTIME-complete in the size of φ when φ is in CTL or the modal mu-calculus [11]. This prompts the question of whether these judgments cannot be computed by cheaper means – such as model checking – for property patterns φ that occur in practice, and this is indeed the case for the patterns documented at the pattern repository site `patterns.projects.cis.ksu.edu` [4].

$$
\begin{array}{rll}
(M,s) \models^{p} q & \text{iff} & q \in L^{\Box}(s) \\
(M,s) \models^{o} q & \text{iff} & q \in L^{\Diamond}(s) \\
(M,s) \models^{m} \neg\varphi & \text{iff} & \text{not } (M,s) \models^{\neg m} \varphi \\
(M,s) \models^{m} \varphi_1 \wedge \varphi_2 & \text{iff} & (M,s) \models^{m} \varphi_1 \text{ and } (M,s) \models^{m} \varphi_2 \\
(M,s) \models^{m} \mathsf{EX}\varphi & \text{iff} & \exists\,(s,s') \in R \text{ with } (M,s') \models^{m} \varphi \\
(M,s) \models^{m} \mathsf{EG}\,\varphi & \text{iff} & \exists \text{ infinite path } \pi \text{ beginning in } s \\
 & & \text{with } (M,t) \models^{m} \varphi \text{ for all states } t \text{ on } \pi \\
(M,s) \models^{m} \mathsf{E}\,[\varphi_1\,\mathsf{U}\,\varphi_2] & \text{iff} & \exists \text{infinite path } \pi = s_0 s_1 s_2 \ldots \text{ and } j \geq 0 \text{ with } s = s_0, \\
 & & (M,s_j) \models^{m} \varphi_2, \text{ and } (M,s_i) \models^{m} \varphi_1 \text{ for all } 0 \leq i < j
\end{array}
$$

Figure 17: Satisfaction relations $\models^{p}$ and $\models^{o}$ for computation tree logic (CTL) over partial Kripke structure $M = (S, R, L)$, where $m \in \{o, p\}$ and $\neg o = p$ and $\neg p = o$.

For φ from CTL or the modal mu-calculus and a partial Kripke structure M with state s one can introduce two judgments $(M,s) \models^{o} \varphi$ and $(M,s) \models^{p} \varphi$ whose semantics is defined as for ordinary Kripke structures, except that $\models^{o}$ has an optimistic view of atomic propositions and $\models^{p}$ interprets them pessimistically. We illustrate this for computation tree logic (CTL):

$$\varphi ::= q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{EG}\,\varphi \mid \mathsf{E}\,[\varphi\,\mathsf{U}\,\varphi] \;, \tag{7}$$

where $q \in \mathsf{AP}$, and other propositional connectives and modalities are derived in the usual manner. The semantics for $\models^{p}$ and $\models^{o}$ is depicted in Figure 17. Note that the pessimistic semantics considers $L^{\Box}(s)$ as those atomic propositions true at state s, whereas the optimistic semantics uses the larger set $L^{\Diamond}(s)$. Also, both semantics interpret conjunctions compositionally. The treatment of negation reveals that both semantics are mutually recursive, reflecting the duality of satisfiability and validity. The semantics of modalities are like the standard ones for Kripke structures, since $R^{\Box} = R^{\Diamond}$ for partial Kripke structures.

The connection between these semantics and the judgments $\mathsf{Sat}\,(M,s,\varphi)$ and $\mathsf{Val}\,(M,s,\varphi)$ is easily established: for all M, s, and φ we have

(UA) $(M,s) \models^{p} \varphi$ implies $\mathsf{Val}\,(M,s,\varphi)$

(OA) $\mathsf{Sat}\,(M,s,\varphi)$ implies $(M,s) \models^{o} \varphi$

The first item (UA) states that $(M,s) \models^{p} \varphi$ under-approximates $\mathsf{Val}\,(M,s,\varphi)$, the second item (OA) expresses that $(M,s) \models^{o} \varphi$ over-approximates $\mathsf{Sat}\,(M,s,\varphi)$.

In practice, this means that a positive model check $(M, s) \models^p \varphi$ certifies that all implementations of (M, s) satisfy φ. On the other hand, if $(M, s) \models^p \varphi$ turns out to be false, we do not know whether all implementations of (M, s) satisfies φ or not.

Example 5. *[11] Consider the partial Kripke structure M from Figure 4. We have $(M, s_0) \models^o \mathsf{AF}\, y_{odd}$ since $y_{odd} \in L^\diamond(s_1)$, noting that AF stands for $\neg\mathsf{EG}\neg$.*

The expression $\mathsf{AG}\,\varphi$ stands for $\neg\mathsf{E}\,[\mathsf{tt}\,\mathsf{U}\,\neg\varphi]$ where tt is a special atomic proposition contained in $\bigcap_{s\in S} L^\diamond(s)$. So $\mathsf{AG}\,(x_{odd} \vee \neg y_{odd})$ states that all states on all infinite paths satisfy that either x is odd or y is even. We have $(M, s_0) \models^o \mathsf{AG}\,(x_{odd} \vee \neg y_{odd})$ since there is only one infinite path $s_0 s_1 s_2^\omega$ from s_0 and $(M, s_i) \models^o x_{odd} \vee \neg y_{odd}$ holds for $i = 0, 1, 2$. The latter is clear for $i = 0, 2$ since then $x_{odd} \in L^\diamond(s_i)$, and for $i = 1$ we do not have $(M, s_1) \models^p y_{odd}$ (since $y_{odd} \notin L^\square(s_1)$) and so $(M, s_1) \models^o \neg y_{odd}$ follows.

Combining both facts, we get $(M, s_0) \models^o (\mathsf{AF}\ y_{odd}) \wedge \mathsf{AG}\,(x_{odd} \vee \neg y_{odd})$. But we do not have $\mathsf{Sat}\,(M, s_0, (\mathsf{AF}\ y_{odd}) \wedge \mathsf{AG}\,(x_{odd} \vee \neg y_{odd}))$. To see this, let us suppose by way of contradiction that there is an implementation (J, j_0) of (M, s_0) satisfying $(\mathsf{AF}\ y_{odd}) \wedge \mathsf{AG}\,(x_{odd} \vee \neg y_{odd})$. In particular, j_0 satisfies $\mathsf{AF}\ y_{odd}$ and so there is a path from j_0 to some j_k in J such that j_k satisfies y_{odd}. Since s_1 is the only state s in M with $y_{odd} \in L^\diamond(s)$ we infer that s_1 has to be refined by j_k. Since j_0 also satisfies $\mathsf{AG}\,(x_{odd} \vee \neg y_{odd})$, we know that j_k satisfies $x_{odd} \vee \neg y_{odd}$ as it is reachable from j_0. Therefore j_k satisfies $y_{odd} \wedge x_{odd}$. But then s_1 must satisfy $x_{odd} \in L^\diamond(s_1)$ since j_k refines s_1, a contradiction.

Although the semantics $\models^p$ and $\models^o$ can be computed with the same complexity as the familiar semantics $\models$ for Kripke structures [11], Example 5 reveals their potential weakness: the compositional treatment of conjunction may not detect dependencies within such conjuncts; in this instance, the conflicting needs of having to make y_{odd} true and false (respectively) at state s_1.

This raises the question of whether there are formulae that do not exhibit such weakness for any model, or whether we can transform such formulae in order to eradicate such weaknesses. We call φ^p a *pessimistic semantic minimization* of φ [25] iff for all partial Kripke structures M and states s we have that $\mathsf{Val}\,(M, s, \varphi)$ iff $(M, s) \models^p \varphi^p$. That is to say, the validity judgment for φ has the same result as the model check for φ^p with respect to $\models^p$. Dually, φ^o is an *optimistic semantic minimization* of φ [25] iff for all partial Kripke structures M and states s we have that $\mathsf{Sat}\,(M, s, \varphi)$ iff $(M, s) \models^o \varphi^o$.

Of particular interest is whether we can choose φ^p to be φ (we then call φ *pessimistically self-minimizing*), or whether we can choose φ^o to be φ (we then call φ *optimistically self-minimizing*). Note that φ is pessimistically self-minimizing iff $\neg\varphi$ is optimistically self-minimizing. Dually, φ is optimistically self-minimizing iff $\neg\varphi$ is pessimistically self-minimizing.

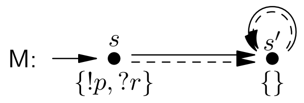

Figure 18: A partial Kripke structure such that $\mathsf{Val}\,(M, s, \mathsf{A}\,[(\neg p \vee \mathsf{AG}\,\neg r)\ \mathsf{W}\,r]\,)$ holds but $(M, s) \models^{p} \mathsf{A}\,[(\neg p \vee \mathsf{AG}\,\neg r)\ \mathsf{W}\,r]$ does not.

Pessimistic self-minimization of a φ of interest is extremely handy, as we can then model check $(M, s) \models^{p} \varphi$ cheaply and know that the result is that for $\mathsf{Val}\,(M, s, \varphi)$. It turns out that many practically important CTL formulae are pessimistically self-minimizing. For example,

$$\mathsf{AG}\ (q \wedge \neg r \rightarrow \mathsf{A}\,[\neg p\ \mathsf{W}\ r]) \tag{8}$$

the CTL encoding of the specification pattern *"Absence of p, Between q and r."* [20] is pessimistically self-minimizing. This can be shown by means of compositional proof rules developed in [25, 4]. One such rule is:

> **(AU Rule)** If φ and ψ are pessimistically self-minimizing, ψ is a universal path formula, and φ and ψ do not share any atomic propositions, then $\mathsf{A}\,[\varphi\ \mathsf{U}\ \psi]$ is pessimistically self-minimizing.

These rules can establish the pessimistic self-minimization of rather complex properties, such as

$$\neg\mathsf{E}\,[\neg q\,\mathsf{U}\ (q \wedge \mathsf{EF}\,(p \wedge (\mathsf{EG}\,(\neg s) \vee\ \mathsf{EF}\,(s \wedge (\mathsf{E}\,[\neg t\,\mathsf{U}\,z] \vee \mathsf{EX}\,(\mathsf{EG}\,(\neg t)))))))] \tag{9}$$

the CTL encoding of *"Constrained Chain after q"* [20].

Not all patterns from `patterns.projects.cis.ksu.edu` are pessimistically self-minimizing. But, surprisingly, all have pessimistic semantic minimizations that transform the original formula in subtle and minimal ways. For example, the pattern *"Absence of p, Before r."* is encoded in CTL as

$$\mathsf{A}\,[(\neg p \vee \mathsf{AG}\,\neg r)\ \mathsf{W}\,r] \tag{10}$$

and is not pessimistically self-minimizing, as can be seen by considering the partial Kripke structure in Figure 18. But

$$\mathsf{A}\,[(\neg p \vee \mathsf{AX}\ (\mathsf{AG}\,\neg r))\ \mathsf{W}\,r] \tag{11}$$

is a pessimistic semantic minimization of the formula in (10). One can conclude this by first writing the formula in (10) as the equivalent

$$\neg\mathsf{E}\,[\neg r\,\mathsf{U}\;(\neg r \wedge \neg\,(\neg p \vee \mathsf{AG}\,\neg r))] \tag{12}$$

and then, by using the aforementioned compositional rules, determine whether the unnegated $\mathsf{E}\,[\neg r\,\mathsf{U}\;(\neg r \wedge \neg\,(\neg p \vee \mathsf{AG}\,\neg r))]$ can be subject to equivalence transformations that result in an optimistically self-minimizing formula. Although (11) results from (10) by unfolding the AG modality once, this heuristic will not always work in trying to compute optimistic or pessimistic semantic minimizations.

Even worse, some CTL formulae do not have optimistic or pessimistic semantic minimizations even within CTL* (which contains CTL). One such example is the formula $\mathsf{A}\,[(\mathsf{EX}q)\;\mathsf{U}\;(\neg q \vee r)]$ [25]. Note that this AU formula does not satisfy the preconditions of the (AU Rule) since its two sub-formulae share an atomic proposition q. One can show, though, that all formulae of the modal mu-calculus have optimistic semantic minimizations as well as pessimistic semantic minimizations [25]. One possible proof [42] observes that distributive formulae with non-redundant propositional clauses, a normal form for mu-calculus enjoying linear-time satisfiability checks [31], are optimistically self-minimizing.

Since self-minimization is such an attractive property, one wonders about the complexity of deciding whether a given φ is optimistically, or pessimistically, self-minimizing. First of all it is easy to see that all valid formulae are optimistically self-minimizing, and that unsatisfiable formulae are pessimistically self-minimizing but not optimistically so. We list the relevant results from [5]:

- For propositional logic, deciding whether φ is optimistically self-minimizing is coNP-complete, and so is the same question for the pessimistic case. Deciding whether φ is optimistically self-minimizing but neither valid nor pessimistically self-minimizing is in DP and coNP-hard. The same complexity applies to deciding whether φ is pessimistically self-minimizing but neither unsatisfiable nor optimistically self-minimizing.

- For propositional modal logic, obtained from (7) by removing the clauses for EG and EU, these three decision problems all turn out to be PSPACE-hard and in EXPSPACE.

- For the modal mu-calculus, all these three decision problems are EXPTIME-hard and in 2EXPTIME.

For the latter two logics it would be of interest to see whether these complexity gaps can be narrowed. For all three logics, it would also be nice to gain a better understanding of the complexity of deciding whether a φ is pessimistically *and* optimistically self-minimizing. The known upper bounds for this are coNP, EXPSPACE, and 2EXPTIME (respectively) [5].

We conclude this section by revisiting characteristic formulae ψ_{s_0} for states s_0 of partial Kripke structures M. It is not hard to show that

$$(N, t_0) \models^p \psi_{s_0} \quad \text{iff} \quad (M, s_0) \prec (N, t_0) \tag{13}$$

We claim that ψ_{s_0} is pessimistically self-minimizing iff for all partial Kripke structures N and states t_0 we have that thorough refinement $I(N, t_0) \subseteq I(M, s_0)$ equals modal refinement $(M, s_0) \prec (N, t_0)$.

First, assume that ψ_{s_0} is pessimistically self-minimizing. Let $(M, s_0) \prec (N, t_0)$ be false. We need to show that $I(N, t_0) \subseteq I(M, s_0)$ is false, too. Since $(M, s_0) \prec (N, t_0)$ is false we get that $(N, t_0) \models^p \psi_{s_0}$ is false as well by (13). By assumption, this means that $\mathsf{Val}\,(N, t_0, \psi_{s_0})$ is false, too. So there has to exist some implementation (J, j) of (N, t_0) that does not satisfy ψ_{s_0}. By (13) again, this implies that (J, j) is not an implementation of (M, s_0).

Second, assume that for all partial Kripke structures N and states t_0 we have that the thorough refinement $I(N, t_0) \subseteq I(M, s_0)$ equals the modal refinement $(M, s_0) \prec (N, t_0)$. Let $\mathsf{Val}\,(N, t_0, \psi_{s_0})$ be true. It suffices to show that $(N, t_0) \models^p \psi_{s_0}$ is true as well. By (13) again, we get this from showing that $(M, s_0) \prec (N, t_0)$. By assumption, it therefore suffices to show that $I(N, t_0) \subseteq I(M, s_0)$. But all implementations of (N, t_0) satisfy ψ_{s_0} since $\mathsf{Val}\,(N, t_0, \psi_{s_0})$ is true, and so they must be implementations of (M, s_0), too, by (13).

The example of Figure 5, witnessing that thorough and modal refinement are different for modal transition systems can be translated into the world of partial Krikpe structures (along the lines of [28]). As a consequence, not all characteristic formulae are pessimistically self-minimizing.

We have already said that $\models^p$ can be decided at the same cost as its 2-valued counterpart $\models$. In [26] it is shown that modal transition systems can be incrementally abstracted by modal transition systems such that the cost of computing abstract models is the same as in the 2-valued world, where transition systems are abstracted by simulating transition systems. Furthermore these abstractions can be synthesized symbolically using BDDs, and they are sound for temporal logics that may mix universal and existential path quantifiers, such as CTL.

This program of computing abstractions automatically is continued in [27], but now with a focus on GMC. That paper also studies fragments of temporal logics for which GMC has linear complexity in the size of the model.

8 Open Problems

Our presentation of modal and mixed specifications identified some open research issues or desired tool support, which we would like to list here in the hope that the community may help with resolving them.

The results in Table 1 can be improved. Recent, not yet published, work of the authors of this paper describes a reduction of acceptance of input for a linearly bounded alternating-time Turing machine (a known EXPTIME-complete problem) to CI for modal transition systems. The reductions presented in this paper therefore provide a knock-on effect for that reduction, meaning that all PSPACE-hard/EXPTIME results in Table 1 can be strengthened to being EXPTIME-complete. There is one notable exception though. Our reductions do not have TR for *modal* specifications as a target, since the reduction from C to TR involves a non-modal specification.

O1 What is the exact complexity of the TR problem for *modal* specifications?

We currently believe that this is more likely to be in PSPACE (and so PSPACE-complete) than EXPTIME-hard.

The open problem O1 has a more general instance. The systems of equations (6) derived from specifications present characteristic formulas in vectorized form. It is folklore knowledge that satisfiability of such vectorized formulas is in EXPTIME. But what is their exact complexity for Satisfiability, Validity, and Implication? Note that the complexity of the latter cannot automatically be derived from the complexities of the former since formulas in vectorized form do not have explicit support for negation.

O2 What is the exact complexity of Satisfiability, Validity, and Implication for formulas given as greatest fixed-point equations in vectorized form?

To see the connection of O2 with O1, let M and N be modal specifications with states s_0 and t_0 (respectively). Then the thorough refinement $I(N, t_0) \subseteq I(M, s_0)$ holds iff the implication $\psi_{t_0} \to \psi_{s_0}$ is valid over implementations, i.e. if all implementations that satisfy ψ_{t_0} also satisfy ψ_{s_0}. Our aforementioned unpublished result of the EXPTIME-completeness of CI for modal specifications appears to be extendable to settle O2 above such that all of its three decision problems are EXPTIME-complete; this argument, which does not resolve O1 above, will be made in the upcoming PhD Thesis of Antonik.

Another question is whether there is a way in which one can avoid either that thorough and modal refinement are different, or that thorough refinement has such high computational complexity. One could approach this in various ways.

First, observe that thorough refinement is a notion derived from, and so determined by, modal refinement: from the definition of modal refinement one derives the definition of what implementations are for specifications, which then determines what constitutes a thorough refinement. Therefore, one could imagine alternative co-inductive definitions of (modal) refinement such that their derived notion of total refinement turns out to capture that modal refinement exactly, and so also has low computational complexity. A natural design constraint for this is that any modified definition of modal refinement should still be intuitive and have nice properties, e.g. be a precongruence for parallel composition operators.

Second, one could consider a sub-class of specifications (and so a corresponding sub-class of implementations) and restrict the existing modal and thorough refinement to it. This may filter out counter-examples of the kind seen in Figure 5. A natural design constraint for this is not to rule out specifications that may actually be desired in modeling and validation activities.

O3 Is there a way of either modifying modal refinement or of restricting the class of specifications such that modal and thorough refinement are equal or, failing that, both have low computational complexity?

For propositional modal logic and the modal mu-calculus, we saw considerable gaps in the lower and upper bounds for the computational complexities of deciding whether a formula is optimistically (or pessimistically) self-minimizing.

O4 For modal logics such as propositional modal logic, CTL, and the modal mu-calculus: what is the exact complexity of deciding whether a formula is pessimistically self-minimizing?

We also studied a composition operator for synchronization of modal specifications. We saw that the composition of implementations is again an implementation. But not all implementations of such a parallel composition are representable as parallel compositions of implementations.

O5 For which composition operators of modal specifications is it decidable whether an implementation of a composition of modal specifications is representable as the composition of implementations?

Finally, we point out the need for tool support for the decision problems discussed in this paper. This raises the question of a suitable back-end formalism into which these problems would be translated for subsequent analysis. Greatest fixed-point formulae in vectorized form, and their decision problems of Satisfiability, Validity, and Implication appear to be a rather obvious choice for CI, C, and TR since front-ends could then compile models into such formulae with relative ease. Alternating tree automata or their variants may be suitable for GMC.

9 Concluding Remarks

As we have not aimed at writing a comprehensive survey of work on modal specifications and their variants, the resulting presentation certainly suffers from incompleteness. Let us conclude with mentioning a few papers as further recommended reading in particular areas of interest.

Godefroid and Jagadeesan show in [28] how various variants of modal transition systems such as partial Kripke structures and what we called mixed specifications are equivalent in the following sense: one can translate one kind of model efficiently into another such that the respective refinement notions are being preserved and reflected. Similarly, temporal logic expressions have efficient translations that preserve meaning. So the complexity results for CI, C, TR, and GMC extend to these variants directly via the reductions given in [28]. Similarly, the complexity results about deciding optimistic and pessimistic self-minimization for partial Kripke structures carry over to such variants. Unfortunately, the compositional proof rules for showing self-minimization [25, 5] are brittle under these reductions, and so new proof rules have to be devised for these variant models, e.g. for modal transition systems.

The paper [30] also makes connections between modal transitions and modal shape graphs, as used in 3-valued shape analysis [45].

In [32] the authors show PSPACE-hardness of solving process algebra equations involving contexts. The problem consists of deciding whether some process, in the form of a labeled transition system, exists that, embedded in the given context, satisfies the given inequality or equality. The equations involve bisimulation, weak bisimulation and modal refinement. That paper also establishes that given that a context is deterministic, in the sense that one action from the same state cannot lead to different states, then some equations can be solved in polynomial time.

In [46] a logical-relation calculus is being developed for the construction of Galois connections, which are an effective and elegant device for construction abstract interpretations [14]. This calculus can combine under-approximating and over-approximating Galois connections, can synthesize the optimal mixed transition systems of [15], and can prove the soundness and optimality of these models. This work can be seen as bringing together the techniques of abstract model checking with those of abstract interpretation.

Last, but definitely not least, the paper [18] convincingly argues that automata provide a simple and elegant framework in which one can represent concrete models, abstractions, and properties so that important analysis questions reduce to familiar problems in automata theory such as non-emptiness checks. Such automata often have a finite maximal model $M(\varphi)$ for a property φ expressible in the modal mu-calculus: the model is finite state, satisfies φ, and abstracts any other model that satisfies φ – where "abstraction" may be based either on simulation relations for automata or on the inclusion of the languages they accept. This is in contrast to the use of modal and mixed specifications: there are formulas φ of the modal mu-calculus for which there are no mixed or modal specifications that are finite and maximal for φ for either modal or thorough refinement [17].

Acknowledgments

We thank Nir Piterman for commenting on drafts of this manuscript. Discussions with Sebastian Uchitel inspired part of the material in Sections 8 and Section 6. Part of the research reported here was carried out under the grants *Efficient Specification Pattern Library for Model Validation (EP/D50595X/1)* and *Complete and Efficient Checks for Branching-Time Abstractions (EP/E028985/1)*, funded by the UK EPSRC. Wąsowski was partly funded by the Center for Embedded Software Systems (CISS), Aalborg University.

References

[1] Luca Aceto and Anna Ingolfsdottir. Characteristic formulae: from automata to logic. *Bulletin of the European Association for Theoretical Computer Science*, February 2007. Concurrency Column.

[2] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering (FSE)*, pages 109–120, Vienna, Austria, September 2001. ACM Press.

[3] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Vardi. Alternating refinement relations. In Davide Sangiorgi and Robert de Simone, editors, *Proceedings of the Ninth International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *LNCS*, pages 163–178. Springer, 1998.

[4] Adam Antonik and Michael Huth. Efficient patterns for model checking partial state spaces in CTL $\cap$ LTL. *Electr. Notes Theor. Comput. Sci.*, 158:41–57, 2006.

[5] Adam Antonik and Michael Huth. On the complexity of semantic self-minimization. *Electr. Notes Theor. Comput. Sci.*, 2008. To appear as Proc. of AVoCS 2007.

[6] Adam Antonik, Michael Huth, Kim G. Larsen, Ulrik Nyman, and Andrzej Wąsowski. Complexity of decision problems for mixed and modal specifications. In *FoSSaCS*, volume 4962 of *LNCS*. Springer, 2008.

[7] Albert Benveniste. Multiple viewpoint contracts and residuation. In *2nd International Workshop on Foundations of Interface Technologies (FIT)*, April 2008.

[8] Anders Børjesson, Kim Guldstrand Larsen, and Arne Skou. Generality in design and compositional verification using tav. *Formal Methods in System Design*, 6(3):239–258, 1995.

[9] Greg Brunet, Marsha Chechik, and Sebastián Uchitel. Properties of behavioural model merging. In Jayadev Misra, Tobias Nipkow, and Emil Sekerinski, editors, *FM*, volume 4085 of *Lecture Notes in Computer Science*, pages 98–114. Springer, 2006.

[10] Glenn Bruns and Patrice Godefroid. Model checking partial state spaces with 3-valued temporal logics. In Nicolas Halbwachs and Doron Peled, editors, *CAV*, volume 1633 of *Lecture Notes in Computer Science*, pages 274–287. Springer, 1999.

[11] Glenn Bruns and Patrice Godefroid. Generalized model checking: Reasoning about partial state spaces. In Catuscia Palamidessi, editor, *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 168–182. Springer, 2000.

[12] Anders Børjesson, Kim Guldstrand Larsen, and Arne Skou. Generality in design and compositional verification using tAV. In *FORTE '92 Proceedings*, pages 449–464, Amsterdam, The Netherlands, The Netherlands, 1993. North-Holland Publishing Co.

[13] Kārlis Čerāns, Jens Chr. Godskesen, and Kim Guldstrand Larsen. Timed modal specification - theory and tools. In *CAV '93: Proceedings of the 5th International Conference on Computer Aided Verification*, pages 253–267, London, UK, 1993. Springer-Verlag.

[14] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[15] Dennis Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. PhD thesis, Eindhoven University of Technology, July 1996.

[16] Dennis Dams, Rob Gerth, and Orna Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst.*, 19(2):253–291, 1997.

[17] Dennis Dams and Kedar S. Namjoshi. The existence of finite abstractions for branching time model checking. In *LICS*, pages 335–344. IEEE Computer Society, 2004.

[18] Dennis Dams and Kedar S. Namjoshi. Automata as abstractions. In Radhia Cousot, editor, *VMCAI*, volume 3385 of *Lecture Notes in Computer Science*, pages 216–232. Springer, 2005.

[19] Nicolás D'Ippolito, Dario Fishbein, Howard Foster, and Sebastian Uchitel. MTSA: Eclipse support for modal transition systems construction, analysis and elaboration. In *eclipse '07: Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange*, pages 6–10, New York, NY, USA, 2007. ACM.

[20] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *ICSE*, pages 411–420, 1999.

[21] Jr. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[22] Alessandro Fantechi and Stefania Gnesi. A behavioural model for product families. In *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 521–524, New York, NY, USA, 2007. ACM.

[23] Dario Fischbein, Sebastian Uchitel, and Victor Braberman. A foundation for behavioural conformance in software product line architectures. In *ROSATEA '06 Proceedings*, pages 39–48, New York, NY, USA, 2006. ACM Press.

[24] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[25] Patrice Godefroid and Michael Huth. Model checking vs. generalized model checking: Semantic minimizations for temporal logics. In *LICS*, pages 158–167. IEEE Computer Society, 2005.

[26] Patrice Godefroid, Michael Huth, and Radha Jagadeesan. Abstraction-based model checking using modal transition systems. *Lecture Notes in Computer Science*, 2154:426–440, 2001.

[27] Patrice Godefroid and Radha Jagadeesan. Automatic abstraction using generalized model checking. In Ed Brinksma and Kim Guldstrand Larsen, editors, *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 137–150. Springer, 2002.

[28] Patrice Godefroid and Radha Jagadeesan. On the expressiveness of 3-valued models. In Lenore D. Zuck, Paul C. Attie, Agostino Cortesi, and Supratik Mukhopadhyay, editors, *VMCAI*, volume 2575 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2003.

[29] Altaf Hussain and Michael Huth. Automata games for multiple-model checking. *Electr. Notes Theor. Comput. Sci.*, 155:401–421, 2006.

[30] Michael Huth, Radha Jagadeesan, and David Schmidt. Modal transition systems: A foundation for three-valued program analysis. *Lecture Notes in Computer Science*, 2028, 2001.

[31] David Janin and Igor Walukiewicz. Automata for the modal mu-calculus and related results. In Jirí Wiedermann and Petr Hájek, editors, *MFCS*, volume 969 of *Lecture Notes in Computer Science*, pages 552–562. Springer, 1995.

[32] Bengt Jonsson and Kim Guldstrand Larsen. On the complexity of equation solving in process algebra. In Samson Abramsky and T. S. E. Maibaum, editors, *TAPSOFT, Vol.1*, volume 493 of *Lecture Notes in Computer Science*, pages 381–396. Springer, 1991.

[33] Kim Guldstrand Larsen. Modal specifications. In Joseph Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 1989.

[34] Kim Guldstrand Larsen. Efficient local correctness checking. In Gregor von Bochmann and David K. Probst, editors, *CAV*, volume 663 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 1992.

[35] Kim Guldstrand Larsen, Ulrik Nyman, and Andrzej Wąsowski. Modal I/O automata for interface and product line theories. In Rocco De Nicola, editor, *ESOP*, volume 4421 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2007.

[36] Kim Guldstrand Larsen, Ulrik Nyman, and Andrzej Wąsowski. On modal refinement and consistency. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2007.

[37] Kim Guldstrand Larsen and Bent Thomsen. A modal process logic. In *LICS*, pages 203–210. IEEE Computer Society, IEEE Computer Society, 1988.

[38] Kim Guldstrand Larsen and Liu 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*, pages 108–117, 1990.

[39] Robin Milner. A modal characterisation of observable machine-behaviour. In Egidio Astesiano and Corrado Böhm, editors, *CAAP*, volume 112 of *Lecture Notes in Computer Science*, pages 25–34. Springer, 1981.

[40] Robin Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25, 1983.

[41] Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[42] Shiva Nejati, Mihaela Gheorghiu, and Marsha Chechik. Thorough checking revisited. In *FMCAD*, pages 106–116. IEEE Computer Society, 2006.

[43] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th GI Conference*, volume 104 of *LNCS*, 1981.

[44] Jean-Baptiste Raclet. Residual for component specifications. In *4th International Workshop on Formal Aspects of Component Software (FACS)*, September 2007.

[45] Shmuel Sagiv, Thomas W. Reps, and Reinhard Wilhelm. Parametric shape analysis via 3-valued logic. In *POPL*, pages 105–118, 1999.

[46] David A. Schmidt. A calculus of logical relations for over- and underapproximating static analyses. *Sci. Comput. Program.*, 64(1):29–53, 2007.

[47] Heiko Schmidt. Comparing disjunctive modal transition systems with an one-selecting variant. In *NWPT'06 – The 18th Nordic Workshop on Programming Theory (NWPT'06) Reykjavík, Iceland, 18-20 October, 2006*, Reykjavík, Iceland, October 2006. Reykjavík University.

[48] Heiko Schmidt. Comparing disjunctive modal transition systems with their one-selecting variant. Master's thesis, Christian-Albrechts-Universität zu Kiel, 2006.

[49] Heiko Schmidt and Harald Fecher. Comparing disjunctive modal transition systems with a one-selecting variant. *To appear in the Journal of Logic and Algebraic Programming*, 2007.

[50] Sebastián Uchitel, Greg Brunet, and Marsha Chechik. Behaviour model synthesis from properties and scenarios. In *ICSE*, pages 34–43. IEEE Computer Society, 2007.

[51] Sebastián Uchitel and Marsha Chechik. Merging partial behavioural models. In Richard N. Taylor and Matthew B. Dwyer, editors, *SIGSOFT FSE*, pages 43–52. ACM, 2004.

[52] D. J. Walker. Bisimulation and divergence. *Inf. Comput.*, 85(2):202–241, 1990.

[53] Wikipedia. Generalized geography — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 8-May-2008].

[54] Liu Xinxin. *Specification and Decomposition in Concurrency*. PhD thesis, Department of Mathematics and Comnputer Science, Aalborg University, April 1992.