On model checking multiple hybrid views

Altaf Hussain and Michael Huth

Department of Computing, South Kensington campus, Imperial College London, London, SW7 2AZ, United Kingdom, {ah701,mrh}@doc.imperial.ac.uk

Abstract. We study consistency, satisfiability, and validity problems for collectively model checking a set of views endowed with labelled transitions, hybrid constraints on states, and atomic propositions. A PTIME algorithm for deciding whether a set of views has a common refinement (consistency) is given. We prove that deciding whether a common refinement satisfies a formula of the hybrid mu-calculus (satisfiability), and its dual (validity), are EXPTIME-complete. We determine two generically generated "summary" views that constitute informative and consistent common refinements and abstractions of a set of views (respectively).

1 Introduction

In model checking [7, 31] one builds a model M of an artefact's state and behavior, expresses desired properties of such state and behavior in a temporal logic or some variant thereof, and verifies (refutes) such a property by verifying (refuting) the instance $M \models \phi$ of a satisfaction relation $\models$.

Often not all aspects of state or behavior are known at model time. For example, a concrete program may be abstracted by a Boolean program [1], or requirements for a design may leave details intentionally under-specified [22]. In such situations, models benefit from being 3-valued so that state and behavior can take on values *true* (guaranteed), *false* (impossible) or $\perp$ (possible). Key benefits are: an explicit under-specification through $\perp$, soundness of abstraction-based model checks for properties that mix path quantifiers [23], and reliable feasibility checks for counter-examples and simulations (see e.g. [28]). By now it is well understood that many such 3-valued notions of models are equally expressive as model checking frameworks [11], demonstrating the robustness of this notion. The additional value $\perp$ in models also blurs the boundaries between abstraction and parameterized model checking since a single model may express infinitely many non-equivalent 2-valued systems as its refinements.

For 3-valued models M, the judgment $M \models \phi$ becomes 3-valued as well and may be written as two predicates $V(M, \phi)$ and $S(M, \phi)$ where the intention of $V(M, \phi)$ is to assert ϕ for M: "ϕ holds in all 2-valued refinements of M;" whereas $S(M, \phi)$ states that ϕ is consistent for M: "some 2-valued refinement of M satisfies ϕ." Deciding these judgments, the generalized model checking of Bruns & Godefroid [3], is EXPTIME-complete for formulas ϕ of the propositional mu-calculus and partial Kripke structures as 3-valued models M [3]. The compositional approximation of these judgments, the 3-valued compositional model

checking algorithm given in [2], can be reduced to two 2-valued checks and therefore done in linear time [3]. Deciding whether a 3-valued model has a 2-valued refinement, the answer to $S(M, \mathit{true})$, is trivial as $S(M, \mathit{true})$ is always true due to the consistency packed into the definition of partial Kripke structures or any other variant of Larsen & Thomsen's modal transition systems [25, 11].

In this paper we re-develop this programme of 3-valued compositional model checking, generalized model checking, and deciding the existence of refinements in a setting where not just one M but finitely many 3-valued views M_i $(1 \leq i \leq k)$ are given and where we wish to reason about these views collectively. Assuming a notion of refinement between 3-valued views, Bruns & Godefroid's generalized model checking then reads:

> For a property ϕ, is there a 2-valued refinement of all M_i satisfying ϕ?
> In particular, is there a 2-valued refinement of all M_i?

We show that the latter decision problem is in PTIME and that the former is EXPTIME-complete for the hybrid mu-calculus of Sattler & Vardi [32]. Our notion of view is a variant of Kripke modal transition systems [17] where some atomic state propositions are nominals, true at exactly one state. Nominals allow for the modelling of agents and their movement, XML documents etc. The semantics of ϕ reflects the hybrid constraints on nominals without increasing the EXPTIME upper bound on satisfiability, due to a result in [32].

The PTIME algorithm for consistency checking computes those tuples of the product state space of all M_i that have a common refinement. This subset is the state space of "summary" views that serve as informative consistent common refinements and abstractions of all M_i (respectively).

Outline of this paper: In Section 2 we define 3-valued views and their 3-valued compositional property semantics. Section 3 defines refinement between views and proves that the 3-valued compositional property semantics is sound for, and logically characterizes, refinement. In Section 4 we define generalized model checking and consistency checking for a finite set of views and show that these problems are reducible to satisfiability checking in the hybrid mu-calculus and, in fact, EXPTIME-complete and so no more complex than generalized model checking for the propositional mu-calculus and a single model. Section 5 develops an efficient algorithm for consistency checking which computes all tuples of states that have a common refinement. The algorithm for consistency checking is used in Section 6 to define "summary" views that are informative common refinements (respectively, abstractions) of a given set of views. Section 7 states related work and Section 8 concludes.

2 Views and their property semantics

We define the 2-valued models that express hybrid views of an artefact, essentially the ones given by Sattler & Vardi in [32]. For the remainder of this paper

let AP, Nom, and Act be mutually disjoint finite sets of state propositions, nominals, and events (respectively) with a designated universal event $u \in Act$ which has transitions between any pair of states. The idea is that $AP \cup Nom$ and Act are observables that annotate states and transitions (respectively). The hybrid nature of nominals $n \in Nom$ comes from restricting the class of 2-valued views to those at which each $n \in Nom$ is true (i.e. annotated) at exactly one state. The universal event u enables one to express familiar hybrid logic operators, such as "at nominal n, ϕ holds," in the hybrid mu-calculus (see e.g. [32]).

Definition 1. *A* 2-valued view M *is a tuple* (S, R, L) *where* S *is a set of states,* $R \subseteq S \times Act \times S$ *is a transition relation with* $\{(s, s') \in S \times S \mid (s, u, s') \in R\} = S \times S$*, and* $L\colon AP \cup Nom \to \mathbb{P}(S)$ *is a labelling function such that* $L(n)$ *is a singleton for all* $n \in Nom$*.*

Example 1. Figure 1 depicts a 2-valued view. Throughout this paper, figures omit transitions for the universal event u.

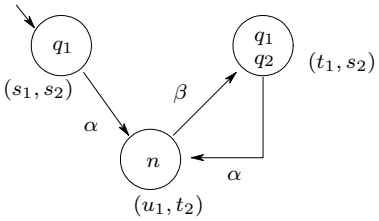

Fig. 1. A 2-valued view with $AP = \{q_1, q_2\}$, $Nom = \{n\}$, $Act = \{\alpha, \beta, u\}$, and three states (s_1, s_2), (u_1, t_2), and (t_1, s_2). It is a common completion of the views in Figure 5.

The hybrid mu-calculus of [32] is defined in negation normal form and its expressiveness relates it closely to description logics [26] which are used in knowledge representation. Here we define its grammar with an unrestricted clause for negation. Hybrid extensions of branching-time temporal logics, e.g. hybrid CTL [10, 14], are all expressible in the hybrid mu-calculus (hMU) whose grammar is

$$\phi ::= q \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle\alpha\rangle\phi \mid \mu Z.\phi \tag{1}$$

where $q \in AP \cup Nom$, $\alpha \in Act \cup \overline{Act}$ with $\overline{Act} = \{\bar{\alpha} \mid \alpha \in Act\}$, and Z ranges over a countable set Var of recursion variables. Our results remain valid without past tense modalities $\langle\bar{\alpha}\rangle$ (by setting $\overline{Act} = \{\}$) or hybrid constraints n (by setting $Nom = \{\}$ throughout). For $\alpha \in Act$, $\bar{\alpha}$ denotes the "inverse" event; its semantics is given by extending the definition of R for a 2-valued view $M = (S, R, L)$ through $((s, \bar{\alpha}, s') \in R$ iff $(s', \alpha, s) \in R)$. For $\alpha \in Act$, we let $\bar{\bar{\alpha}}$ be α again. We write $\phi_1 \vee \phi_2$ for $\neg(\neg\phi \wedge \neg\phi_2)$, $\phi_1 \to \phi_2$ for $\neg(\phi_1 \wedge \neg\phi_2)$, and, for all $\alpha \in Act \cup \overline{Act}$, $[\alpha]\phi$ for $\neg\langle\alpha\rangle\neg\phi$. In the least fixed point formula

$\mu Z.\phi$, μZ binds all occurrences of Z in ϕ with static scoping and we require that all free occurrences of Z in ϕ are under an even scope of negations. If $\phi[Z/\psi]$ denotes the formula obtained from ϕ by replacing all free occurrences of Z in ϕ with ψ, the greatest fixed point formula $\nu Z.\phi$ is derived as $\neg\mu Z.\neg\phi[Z/\neg Z]$. A formula ϕ is closed if it contains no free recursion variables. Its meaning is then independent from an environment. The semantics of formulas over 2-valued views is a special case of that for 3-valued views so we first define those models and their semantics.

Definition 2. *A* 3-valued view M *is a pair* (M^a, M^c) *with* $M^a = (S, R^a, L^a)$ *and* $M^c = (S, R^c, L^c)$ *where* S *is a set of states,* $R^a, R^c \subseteq S \times (Act \cup \overline{Act}) \times S$ *are transition relations with* $R^a \subseteq R^c$, $\{(s, s') \in S \mid (s, u, s') \in R^a\} = S \times S$, *and* $((s, \bar{\alpha}, s') \in R^m$ *iff* $(s', \alpha, s) \in R^m)$ *for* $m \in \{a, c\}$, *and* $L^a, L^c \colon AP \cup Nom \to \mathbb{P}(S)$ *are labelling functions with* $L^a(q) \subseteq L^c(q)$ *for all* $q \in AP \cup Nom$ *subject to the following constraints, for all* $n \in Nom$*:*

1. $L^a(n)$ *contains at most one state and* $L^c(n)$ *is non-empty,*
2. *if* $L^a(n)$ *is non-empty,* $L^a(n) = L^c(n)$.

For any $s \in S$, (M, s) *is a* pointed 3-valued view M *with initial state* s.

The intuition behind 3-valued views is that all a-structure specifies asserted (guaranteed, valid etc) information, whereas all c-structure declares consistent (possible, satisfiable etc) information [25]. The complement of c-information specifies an impossibility, e.g. $(s, \alpha, s') \notin R^c$ expresses that event α cannot lead from state s to state s'. Note that the universal event u is asserted to be so. The inclusions for transitions and labels ensure logical consistency of the semantics given for hMU below. In particular, the universal event u is universally possible. The two constraints on labelling nominals are based on [14] where $s \in L^a(n)$ states "n is at s," and that $s \in L^c(n)$ expresses "n may be at s."

Example 2. Two 3-valued views (M_1, s_2) and (M_2, s_2) are depicted in Figure 4. In figures of this paper, transitions for all $\bar{\alpha} \in \overline{Act}$ are implied, solid lines denote R^a-transitions, and dashed lines denote $R^c \setminus R^a$-transitions. For $q \in AP \cup Nom$, a label q, $q?$, or its absence at state t denote $t \in L^a(q)$, $t \in L^c(q) \setminus L^a(q)$, and $t \notin L^c(q)$ (respectively).

The denotational semantics $[\![\cdot]\!]^m$ of hMU over 3-valued views maps formulas ϕ and environments ρ, functions $Z \mapsto (\rho^a(Z), \rho^c(Z))$ of type $Var \to \{(L, U) \subseteq S \times S \mid L \subseteq U\}$, into sets of states for a mode of analysis $m \in \{a, c\}$ in Figure 2. Note that $\neg a = c$, $\neg c = a$, and $\mathrm{pre}^m_\alpha(A) = \{s \in \Sigma \mid \exists s' \in A \colon (s, \alpha, s') \in R^m\}$ for all $\alpha \in Act \cup \overline{Act}$ and $m \in \{a, c\}$.

Definition 3. *We write* $(M, s) \models^a_\rho \phi$ *for* $s \in [\![\phi]\!]^a_\rho$, *saying that* ϕ *is a (*ρ*-)*valid *assertion at* s; *similarly* $(M, s) \models^c_\rho \phi$ *means* $s \in [\![\phi]\!]^c_\rho$ *and denotes that* ϕ *is (*ρ*-)*consistent *at* s. *(If* ϕ *is closed, we elide* ρ*.)*

$$\begin{array}{ll} [\![q]\!]^m_\rho = L^m(q) & [\![Z]\!]^m_\rho = \rho^m(Z) \\ [\![\neg\phi]\!]^m_\rho = \Sigma \setminus [\![\phi]\!]^{\neg m}_\rho & [\![\phi_1 \wedge \phi_2]\!]^m_\rho = [\![\phi_1]\!]^m_\rho \cap [\![\phi_2]\!]^m_\rho \\ [\![\langle\alpha\rangle \phi]\!]^m_\rho = \mathrm{pre}^m_\alpha([\![\phi]\!]^m_\rho) & [\![\mu Z.\phi]\!]^m_\rho = \mathrm{lfp}\lambda A.[\![\phi]\!]^m_{\rho[Z\mapsto A]}\,. \end{array}$$

Fig. 2. Semantics of hMU over 3-valued views for mode $m \in \{a, c\}$.

Least fixed points $\mathrm{lfp}\lambda A.[\![\phi]\!]^m_{\rho[Z\mapsto A]}$ are formed in the complete lattice $(\mathbb{P}(\Sigma), \subseteq)$. Note that for $m \in \{a, c\}$ we have $s \models^m [\alpha]\phi$ iff (for all $(s, \alpha, s') \in R^{\neg m}$, $s' \models^m \phi$). If K is a 2-valued view, we may cast it into a 3-valued view M with $M^a = M^c = K$. Then $[\![\phi]\!]^a_\rho = [\![\phi]\!]^c_\rho$ holds in M for all ρ and ϕ of hMU so this defines the 2-valued semantics $k \models_\rho \phi$ to be $k \in [\![\phi]\!]^a_\rho$ for all states k of K.

Example 3. For (M_2, s_2) in Figure 4 we have $s_2 \models^c \neg\langle\alpha\rangle[\beta]\langle\alpha\rangle \mathit{true}$ since we don't have $s_2 \models^a \langle\alpha\rangle[\beta]\langle\alpha\rangle \mathit{true}$, for $(s_2, \alpha, x) \in R^a$ implies $x = t_2$, $(t_2, \beta, t_2) \in R^c$, but there is no R^a-transition labelled with α out of t_2. We have $s_2 \models^a \mu Z.[\beta]\neg \mathit{true} \wedge \langle\bar{\beta}\rangle\langle\bar{\alpha}\rangle Z$ since there is no $(s_2, \beta, x) \in R^c$ and so $s_2 \models^a [\beta]\neg \mathit{true}$, and there is a R^a-cycle $(s_2, \bar{\beta}, t_2)(t_2, \bar{\alpha}, s_2)$ from s_2 to s_2 that generates the word $\bar{\beta}\bar{\alpha}$.

3 Refinement between views

In specifying a 3-valued view we implicitly describe a possibly infinite set of 2-valued views. Such intuitions can be formalized in Cousot & Cousot's framework of abstract interpretation [8] or through a co-inductive definition of refinement, as done by Larsen & Thomsen in [25].

Remark 1. One can reconcile both formalizations. First, a co-inductive refinement defines a concretization function $(M, s) \mapsto \mathcal{C}(M, s)$ for pointed 3-valued views, where $\mathcal{C}(M, s)$ is the class of pointed 3-valued views that refine (M, s) and are 2-valued up to casting. Second, it is less obvious how one should abstract a class C of 2-valued views by a single pointed 3-valued view as any (M, s) with $C \subseteq \mathcal{C}(M, s)$ is a possible abstraction. But from results in [15] one can derive a Galois adjunction between compact sets C of 2-valued models and bounded Scott-closed sets of 3-valued models in the domain model for refinement of [18].

We won't elaborate this point further and turn to defining refinement.

Definition 4. *For $i = 1, 2$ let $(M_i, s_i) = (((S_i, R^a_i, L^a_i), (S_i, R^c_i, L^c_i)), s_i)$ be pointed 3-valued views. Then (M_1, s_1) is* refined *by (M_2, s_2) iff there is a relation $Q \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in Q$ and, for all $(s, t) \in Q$, we have*

1. *for all $q \in AP \cup Nom$, $s \in L^a_1(q)$ implies $t \in L^a_2(q)$,*
2. *for all $q \in AP \cup Nom$, $t \in L^c_2(q)$ implies $s \in L^c_1(q)$,*
3. *for all $\alpha \in Act \cup \overline{Act}$, if $(s, \alpha, s') \in R^a_1$, there is $(t, \alpha, t') \in R^a_2$ with $(s', t') \in Q$,*
4. *for all $\alpha \in Act \cup \overline{Act}$, if $(t, \alpha, t') \in R^c_2$, there is $(s, \alpha, s') \in R^c_1$ with $(s', t') \in Q$.*

We write $(M_1, s) \prec (M_2, t)$ *whenever there is such a* Q *with* $(s, t) \in Q$ *and denote by* $\mathcal{C}(M, s)$ *the* completions *of* (M, s)*, the set of those refinements* (N, t) *of* (M, s) *that are 2-valued up to casting.*

Example 4. The pointed 3-valued view $(\mathcal{V}_+, (s_1, s_2))$ of Figure 6 refines the views (M_1, s_2) and (M_2, s_2) of Figure 5 where refinement relates states from hybrid views to their tuple state: s_1 to (s_1, s_2), t_2 to (u_1, t_2) etc.

Since refinement is transitive, $(M_1, s) \prec (M_2, t)$ implies $\mathcal{C}(M_2, t) \subseteq \mathcal{C}(M_1, s)$ and the converse has been proved for views without nominals in [16]. Proofs from the non-hybrid setting, showing that the fixed-point free fragment of the modal mu-calculus logically characterizes refinement and that refinement is sound with respect to the compositional 3-valued property semantics [23], also apply to our setting and so we state them without proof.

Theorem 1. *For pointed 3-valued views* (M, s) *and* (N, t) *we have* $(M, s) \prec (N, t)$ *iff (for all closed, fixed-point free formulas* ϕ *of hMU,* $s \in [\![\phi]\!]^a$ *implies* $t \in [\![\phi]\!]^a$*). If* $(M, s) \prec (N, t)$*, then* $s \in [\![\phi]\!]^a$ *implies* $t \in [\![\phi]\!]^a$*, and* $t \in [\![\psi]\!]^c$ *implies* $s \in [\![\psi]\!]^c$*, for all closed* ϕ, ψ *of hMU.*

This logical characterization and soundness secure soundness of $[\![\phi]\!]^m$ relative to the thorough semantics of Bruns & Godefroid in [3] adapted to our hybrid setting, where for any closed ϕ of hMU the predicate $GMC(M, s, \phi)$ means "$k \models \phi$ for some $(K, k) \in \mathcal{C}(M, s)$."

Corollary 1. *For any closed* ϕ *in hMU and state* s *of any 3-valued view* M*:*

1. *If* $s \in [\![\phi]\!]^a$*, then* $GMC(M, s, \neg\phi)$ *is false.*
2. *If* $GMC(M, s, \phi)$ *is true, then* $s \in [\![\phi]\!]^c$*.*

Proof. This is proved as in [3] where completions may violate hybrid constraints, which soundly over- and under-approximates the statements in items 1 and 2 (respectively) where $GMC(M, s, \phi)$ refers to hybrid models only. ∎

Example 5. For (M_2, s_2) from Figure 5 we have $s_2 \not\models^a \langle\beta\rangle true \vee \neg\langle\beta\rangle true$ as there is no $(s_2, \beta, x) \in R^a$ but $(s_2, \beta, t_2) \in R^c$. The formula is a tautology and so true for all completions of (M_2, s_2): the converse of item 1 is not true.

4 Multiple views and their decision problems

We can now define the decision problems studied in this paper. Let $\mathcal{V} = \{(M_i, s_i) \mid 1 \leq i \leq k\}$ be any finite set of pointed 3-valued views (M_i, s_i) where all $(M_i, s_i) \in \mathcal{V}$ have only finitely many states. Each (M_i, s_i) may be an abstraction of a concrete and consistent artefact, e.g. a piece of software. Alternatively, each (M_i, s_i) may be the description of a hybrid knowledge-representation or database view (where hybrid constraints are needed in XML documents) or the description of a particular stake holder in the sense of requirements engineering.

In the case of software, consistency of all $(M_i, s_i) \in \mathcal{V}$ is usually assured as the artefact is a consistent common refinement by construction, up to casting and representational changes. In the latter cases, consistency is a primary concern in crafting a model that honors all views $(M_i, s_i) \in \mathcal{V}$. For example, if each (M_i, s_i) is an answer to a query from a local database, $\mathcal{V}$ is often not consistent. We identify decision problems.

Definition 5. *Let* $\mathcal{C}(\mathcal{V}) = \{(N,t) \mid \forall (M,s) \in \mathcal{V}\colon (N,t) \in \mathcal{C}(M,s)\}$ *be the set of* common completions *of* $\mathcal{V}$*. For closed* ϕ *of hMU, we define parameterized boolean expressions* $C(\mathcal{V})$*,* $S(\mathcal{V},\phi)$*, and* $V(\mathcal{V},\phi)$*:*

1. Consistency: $C(\mathcal{V})$ *holds iff all views of* $\mathcal{V}$ *have a common completion, i.e. iff* $\mathcal{C}(\mathcal{V}) \neq \{\}$.
2. Satisfiability: $S(\mathcal{V},\phi)$ *is true iff there is a common completion of* $\mathcal{V}$ *that satisfies* ϕ*, i.e. iff* $\{(N,t) \in \mathcal{C}(\mathcal{V}) \mid t \models \phi\} \neq \{\}$.
3. Validity: $V(\mathcal{V},\phi)$ *holds iff all common completions of* $\mathcal{V}$ *satisfy* ϕ.

Since all pointed 3-valued (M,s) have (M^a,s) as a completion, $\mathcal{C}(\mathcal{V})$ holds iff all views of $\mathcal{V}$ have a common *refinement*. Note that $V(\mathcal{V},\phi)$ holds for all ϕ if $\mathcal{V}$ has no common refinement. Thus it is wise to first decide $C(\mathcal{V})$ so as to avoid unintended certifications through $V(\mathcal{V},\phi)$. We show that all three decision problems above are reducible to satisfiability checks of hMU over 2-valued views. Inspired by [23] we construct a closed formula $[M_i, s_i]$ of hMU for each pointed 3-valued view (M_i, s_i) such that for all pointed 3-valued views (N,t) we have

$$(N,t) \models^a [M_i, s_i] \qquad \text{iff} \qquad (M_i, s_i) \prec (N,t)\,. \tag{2}$$

The existence of such formulas secures the desired reductions.

Theorem 2. *1. Each pointed 3-valued view* (M_i, s_i) *has a formula* $[M_i, s_i]$ *of the hybrid mu-calculus satisfying (2) for all pointed 3-valued views* (N,t)*.*
2. The decision problems $C(\mathcal{V})$*,* $S(\mathcal{V},\phi)$*, and* $V(\mathcal{V},\phi)$ *are reducible to satisfiability checks of hMU over 2-valued views and in EXPTIME.*

Proof. 1. For each state t_i in M_i we set (similar to (3) in [23])

$$[M_i, t_i] = \Big(\bigwedge_{(t_i,\alpha,t_i') \in R^a} \langle\alpha\rangle [M_i, t_i']\Big) \wedge \Big(\bigwedge_{\alpha \in Act} [\alpha]\Big(\bigvee_{(t_i,\alpha,t_i') \in R^c} [M_i, t_i']\Big)\Big) \tag{3}$$
$$\wedge \bigwedge \{q \mid t_i \in L^a(q)\} \wedge \bigwedge \{\neg q \mid t_i \notin L^c(q)\}$$

as a system of greatest fixed point equations. As M_i has only finitely many states, each $[M_i, t_i]$ is expressible in hMU. The proof that $[M_i, s_i]$ satisfies (2) is basically the one given in [25].

2. We can reduce $C(\mathcal{V})$ by proving that $\mathcal{V} = \{(M_i, s_i) \mid 1 \leq i \leq k\}$ has a common completion iff the closed formula

$$\sigma_{\mathcal{V}} = \bigwedge_{i=1}^{k} [M_i, s_i] \tag{4}$$

of hMU is satisfiable over 2-valued views. If $\sigma_{\mathcal{V}}$ is satisfiable, $k \models \sigma_{\mathcal{V}}$ for some pointed 2-valued view (K,k). Since (K,k) can be cast into a pointed 3-valued view, (2) and $k \models \sigma_{\mathcal{V}}$ render $(M_i,s_i) \prec (K,k)$ for all $i = 1,2,\ldots,k$ and so $(K,k) \in \mathcal{C}(\mathcal{V})$. Conversely, if $\mathcal{V}$ has a common completion (K,k) we have $(M_i,s_i) \prec (K,k)$ for all $i = 1,2,\ldots,k$. Using (2) this implies $(K,k) \models \sigma_{\mathcal{V}}$ and so $\sigma_{\mathcal{V}}$ is satisfiable over 2-valued views. The reductions for $S(\mathcal{V},\phi)$ and $V(\mathcal{V},\phi)$ are variations of the reduction for $C(\mathcal{V})$. The check $S(\mathcal{V},\phi)$ holds iff $\phi \wedge \sigma_{\mathcal{V}}$ is satisfiable over 2-valued views. The check $V(\mathcal{V},\phi)$ holds iff $\neg\phi \wedge \sigma_{\mathcal{V}}$ is unsatisfiable over 2-valued views. The EXPTIME result follows directly from these reductions and Theorem 2 of [32]. ∎

Example 6. For (M_2,s_2) from Figure 5 we express $[M_2,s_2]$ in hMU with $Z_{s_2}, Z_{t_2} \in Var$: $[M_2,s_2] = \nu Z_{s_2}.\langle\alpha\rangle\psi \wedge [\alpha]\psi \wedge [\beta]\psi \wedge [\bar{\beta}]\psi \wedge q_1 \wedge \neg n$ where $\psi = \nu Z_{t_2}.\langle\bar{\alpha}\rangle Z_{s_2} \wedge [\alpha]Z_{t_2} \wedge [\beta]Z_{s_2} \wedge [\bar{\alpha}]Z_{s_2} \wedge [\bar{\beta}]Z_{s_2} \wedge n \wedge \neg q_1 \wedge \neg q_2$.

The semantics of Figure 2 is in NP and in co-NP via a reduction to 2-valued checks similar to the one in [3]. Such a reduction is not possibly in general for $S(\mathcal{V},\phi)$ and $V(\mathcal{V},\phi)$ as they are EXPTIME-complete.

Theorem 3. *$S(\mathcal{V},\phi)$ and $V(\mathcal{V},\phi)$ are EXPTIME-complete in the size of ϕ.*

Proof. For $\mathcal{V} = \{(M,s)\}$, $S(\mathcal{V},\phi)$ and $V(\mathcal{V},\phi)$ ask whether some (respectively all) completions of (M,s) satisfy ϕ. So $S(\mathcal{V},\phi)$ is the generalized model checking problem $GMC(M,s,\phi)$ of Bruns & Godefroid in [3] and $V(\mathcal{V},\phi)$ its dual. Since $GMC(M,s,\phi)$ is EXPTIME-complete for formulas of the modal mu-calculus [3] (essentially, hMU with $\overline{Act} = \{\}$ and $Nom = \{\}$ by [11]), $S(\mathcal{V},\phi)$ and $V(\mathcal{V},\phi)$ are EXPTIME-hard for general $\mathcal{V}$ and ϕ of hMU. By Theorem 2 the decision problems $S(\mathcal{V},\phi)$ and $V(\mathcal{V},\phi)$ are in EXPTIME and so EXPTIME-complete. ∎

5 Complexity of common refinement checks

Practical considerations suggest to investigate whether the upper bound of Theorem 2 can be lowered for $C(\mathcal{V})$, which we now do.

Definition 6. *Let $\mathcal{V} = \{(M_i,s_i) \mid 1 \leq i \leq k\}$, each M_i having state space S_i.*

1. *We denote $\prod_{i=1}^{k} S_i$ by $S_{\mathcal{V}}$, write $\boldsymbol{t}$ for tuples $(t_1,t_2,\ldots,t_k) \in S_{\mathcal{V}}$, and use $\mathcal{V}_{\boldsymbol{s}}$ to stress that s_i is the initial state in each (M_i,s_i) of $\mathcal{V}$.*
2. *A* common refinement witness *is a relation $W \subseteq S_{\mathcal{V}}$ such that $\boldsymbol{t} \in W$ implies*
   (a) *for all i and $q \in AP \cup Nom$, if $t_i \in L^a(q)$ then $t_j \in L^c(q)$ for all $j \neq i$,*
   (b) *for all i and $\alpha \in Act \cup \overline{Act}$, if $(t_i,\alpha,t_i') \in R^a$, there is $\boldsymbol{t'} \in W$ such that $(t_j,\alpha,t_j') \in R^c$ for all $j \neq i$, and*
   (c) *for all $n \in Nom$, there is $\boldsymbol{t} \in W$ such that for all i, $t_i \in L^c(n)$.*

Note that in clause (b) above the ith coordinate of $\boldsymbol{t'}$ is bound to the given t_i' and that clause (c) is required as each n holds in a state of K for any $(K,k) \in \mathcal{C}(\mathcal{V})$. As the arbitrary union of common refinement witnesses is a common refinement witness, there is a greatest common refinement witness for each $\mathcal{V}_{\boldsymbol{s}}$, denoted by $W_{\mathcal{V}_{\boldsymbol{s}}}$. This relation captures the existence of common refinements.

Theorem 4. *For any $\mathcal{V}_{\boldsymbol{s}}$, the predicate $C(\mathcal{V}_{\boldsymbol{s}})$ is equivalent to "$\boldsymbol{s} \in W_{\mathcal{V}_{\boldsymbol{s}}}$."*

Proof. We show $W = \{\boldsymbol{t} \in S_{\mathcal{V}_{\boldsymbol{s}}} \mid \mathcal{C}(\mathcal{V}_{\boldsymbol{t}}) \neq \{\}\} \subseteq W_{\mathcal{V}_{\boldsymbol{s}}}$. Given $\boldsymbol{t} \in W$, there is $(K,k) = ((S_K, R_K, L_K), k) \in \mathcal{C}(\mathcal{V}_{\boldsymbol{t}})$. Clause (b): For any i if $(t_i, \alpha, t_i') \in R^a$, there is $(k, \alpha, k') \in R_K$ with $(M_i, t_i') \prec (K, k')$ as $(M_i, t_i) \prec (K, k)$. Since $(M_j, t_j) \prec (K, k)$ for all $j \neq i$ and $(k, \alpha, k') \in R_K$, there is $(t_j, \alpha, t_j') \in R^c$ with $(M_j, t_j') \prec (K, k')$ for each $j \neq i$. In particular, $\mathcal{V}_{\boldsymbol{t'}}$ has (K, k') as a common refinement and so $\boldsymbol{t'} \in W$. A similar reasoning applies to clauses (b) and (c) and so $W \subseteq W_{\mathcal{V}_{\boldsymbol{s}}}$.

1. Let $C(\mathcal{V}_{\boldsymbol{s}})$. Then $\boldsymbol{s} \in W \subseteq W_{\mathcal{V}_{\boldsymbol{s}}}$.
2. Let $\boldsymbol{s} \in W_{\mathcal{V}_{\boldsymbol{s}}}$. We define $K = (W_{\mathcal{V}_{\boldsymbol{s}}}, R, L)$ as the product of c-structure: $(\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R$ iff (for all i, $(t_i, \alpha, t_i') \in R^c$), and $\boldsymbol{t} \in L(q)$ iff (for all i, $t_i \in L^c(q)$) for $q \in AP$. For $n \in Nom$, we set $L(n) = \{t\}$ for some $t \in W_{\mathcal{V}_{\boldsymbol{s}}}$ satisfying clause (c) of Definition 6.2. We claim that $(K, \boldsymbol{s}) \in \mathcal{C}(\mathcal{V}_{\boldsymbol{s}})$ with refinement $\{(t_i, \boldsymbol{t}) \mid \boldsymbol{t} \in W_{\mathcal{V}_{\boldsymbol{s}}}\}$ showing $(M_i, t_i) \prec (K, \boldsymbol{t})$. By definition, any transition from $\boldsymbol{t} \in W_{\mathcal{V}_{\boldsymbol{s}}}$ in K or propositional/nominal label at $\boldsymbol{t}$ in K is c-matched for t_i in each M_i. Conversely, any a-transition (t_i, α, t_i') in M_i with $\boldsymbol{t} \in W_{\mathcal{V}_{\boldsymbol{s}}}$ ensures matching c-transitions (t_j, α, t_j') for all $j \neq i$ such that $\boldsymbol{t'} \in W_{\mathcal{V}_{\boldsymbol{s}}}$ as $\boldsymbol{t} \in W_{\mathcal{V}_{\boldsymbol{s}}}$. So $(\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R$ as $R^a \subseteq R^c$ in M_i. Since $\boldsymbol{t'} \in W_{\mathcal{V}_{\boldsymbol{s}}}$ this works co-inductively. A similar argument applies to $t_i \in L^a(q)$ and $t_i \in L^a(n)$. ∎

Figure 3 shows an algorithm for computing $W_{\mathcal{V}_{\boldsymbol{s}}}$ where we omitted any optimizations for sake of clarity. This algorithm is related to the partition refinement algorithms for computing the greatest bisimulation relation (see e.g. [29]), except that $W_{\mathcal{V}_{\boldsymbol{s}}}$ is not an equivalence relation and so no partition or splitting occurs. However, if $\mathcal{V}$ consists of two pointed 2-valued views the algorithm is a non-optimal version of the familiar splitting algorithm for bisimulation.

```
No = {};
let (bad (t, No)) = // fails clause (a) of Definition 6.2:
  ((some i, j, q | t_i in L^a(q) && not t_j in L^c(q))
  ||                // fails clause (b) of Definition 6.2:
   (some (t_i,a,x) in R^a | all t' in S_V minus No |
      x = t'_i ==> some j | not (t_j,a,t_j') in R^c
  )) in
  { while (some t in S_V minus No | (bad (t, No))) {
      No = No union {t};
  } // else-branch: fails clause (c) of Definition 6.2
if (all n in Nom | some t' in S_V minus No | all i | t_i in L^c(n))
  { Yes = S_V minus No; } else { Yes = {}; }
```

Fig. 3. Computing $W_{\mathcal{V}_{\boldsymbol{s}}}$ for a given set of views $\mathcal{V}_{\boldsymbol{s}}$, where `union` and `minus` denote set-theoretic union and complement, respectively.

Theorem 5. *The algorithm of Figure 3 terminates after at most $\mid S_{\mathcal{V}} \mid$ iterations and assigns to* `Yes` *the set $W_{\mathcal{V}_{\boldsymbol{s}}}$.*

Proof. For termination, `S_V minus No` equals `S_V` initially and `No` is a subset of `S_V` that increases by one at each iteration so there cannot be more iterations than elements in `S_V`. It remains to show correctness:

- For $W_{\mathcal{V}_s} \subseteq$ `Yes` it suffices to show $W \subseteq$ `Yes` for a non-empty common refinement witness $W \subseteq S_{\mathcal{V}}$. To that end, it suffices to show that $W \subseteq$ `S_V minus No` is an invariant of the while-statement as $\{\} \neq W$ forces execution of the `if`-branch. The inclusion $W \subseteq$ `S_V minus No` holds initially as then `No` is empty and $W \subseteq$ `S_V`. Assume that $W \subseteq$ `S_V minus No` holds right before an iteration of the while-statement. Given $t \in W$, the expression `(bad (t, No))` is false since t is in the common refinement witness W and the range of the quantifier `all t'` is the set `S_V minus No` and subsumes W by assumption. Thus, no $t \in W$ can be added to `No`.
- For `Yes` $\subseteq W_{\mathcal{V}_s}$ it suffices to show that a non-empty `Yes` is a common refinement witness. After the assignment to the non-empty `Yes`, the expression `(bad (t, No))` is false for all t in `Yes` and the Boolean guard of the `if`-statement is true, so this states that `Yes` is a common refinement witness. ∎

Example 7. Let $Act = \{\alpha, \beta, u\}$ and $\overline{Act} = \{\bar{\alpha}, \bar{\beta}, \bar{u}\}$.

1. Let $\mathcal{V} = \{(M_1, s_1), (M_2, s_2)\}$ with $S_1 = \{s_1, t_1, u_1\}$, $S_2 = \{s_2, t_2\}$, $AP \cup Nom = \{\}$, and transitions and labelling as in Figure 4. The algorithm nondeterministically computes `Yes` to be empty: add (t_1, s_2) to `No` as $(t_1, \beta, s_1) \in R_1^a$ and there is no $(s_2, \beta, x) \in R_2^c$; add (u_1, s_2) to `No` as $(s_2, \bar{\beta}, t_2) \in R_2^a$ and there is no $(u_1, \bar{\beta}, x) \in R_1^c$; add (u_1, t_2) since $(t_2, \beta, s_2) \in R_2^a$ and the only match $(u_1, \beta, t_1) \in R_1^c$ is such that (t_1, s_2) is already in the set `No`; add (s_1, s_2) as $(s_1, \alpha, u_1) \in R_1^a$ but the only match $(s_2, \alpha, t_2) \in R_2^c$ is such that (u_1, t_2) is in `No`; add (s_1, t_2) since $(t_2, \beta, t_2) \in R_2^a$ and there is no $(s_1, \beta, x) \in R_1^c$; and finally add (t_1, t_2) as $(t_1, \beta, s_1) \in R_1^a$ but (s_1, x) is in `No` for all choices of x.
2. Figure 5 shows $\mathcal{V} = \{(M_1, s_1), (M_2, s_2)\}$ with $Nom = \{n\}$ and $W_{\mathcal{V}} = \{(s_1, s_2), (u_1, t_2), (t_1, s_2)\}$; all other pairs are ruled out as they violate clause (a) of Definition 6.2 for n.

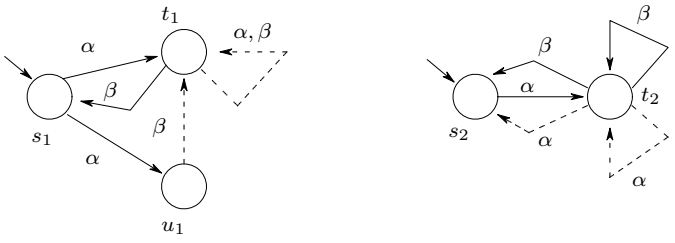

Fig. 4. A $\mathcal{V} = \{(M_1, s_1), (M_2, s_2)\}$ for which no $\mathcal{V}_t$ has a common refinement.

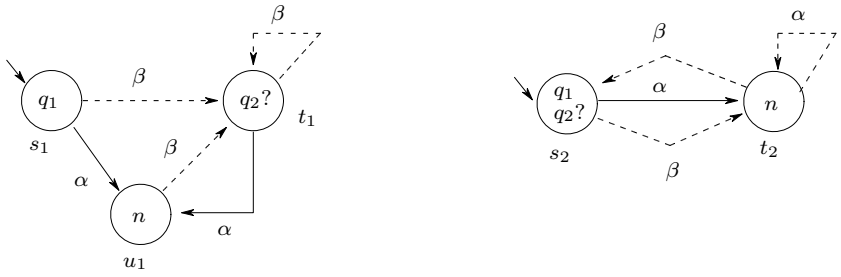

Fig. 5. A $\mathcal{V} = \{(M_1, s_1), (M_2, s_2)\}$ for which there are common refinements.

6 Summary views

We construct 3-valued "summary" views $\mathcal{V}_-$ and $\mathcal{V}_+$ that serve as consistent and informative common abstractions and refinements of $\mathcal{V}$ (respectively).

Definition 7. *Let $\mathcal{V}$ be given. For $\star \in \{-, +\}$ we define a 3-valued view $\mathcal{V}_\star = (\mathcal{V}^a_\star, \mathcal{V}^c_\star)$ with $\mathcal{V}^m_\star = (W_\mathcal{V}, R^m_{\mathcal{V}_\star}, L^m_{\mathcal{V}_\star})$ for $m \in \{a, c\}$ such that for all $q \in AP \cup Nom$, $\alpha \in Act \cup \overline{Act}$*

$$
\begin{array}{ll}
\boldsymbol{t} \in L^a_{\mathcal{V}_+}(q) \text{ iff } \exists i\colon t_i \in L^a(q) & \boldsymbol{t} \in L^c_{\mathcal{V}_+}(q) \text{ iff } \forall i\colon t_i \in L^c(q) \\
(\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R^a_{\mathcal{V}_+} \text{ iff } \exists i\colon (t_i, \alpha, t'_i) \in R^a & (\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R^c_{\mathcal{V}_+} \text{ iff } \forall i\colon (t_i, \alpha, t'_i) \in R^c \\
\boldsymbol{t} \in L^a_{\mathcal{V}_-}(q) \text{ iff } \forall i\colon t_i \in L^a(q) & \boldsymbol{t} \in L^c_{\mathcal{V}_-}(q) \text{ iff } \exists i\colon t_i \in L^c(q) \\
(\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R^a_{\mathcal{V}_-} \text{ iff } \forall i\colon (t_i, \alpha, t'_i) \in R^a & (\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R^c_{\mathcal{V}_-} \text{ iff } \exists i\colon (t_i, \alpha, t'_i) \in R^c
\end{array}
$$

except for $n \in Nom$: $L^a_{\mathcal{V}_+}(n) = \{\}$ if its definition above does not render a singleton; and $L^c_{\mathcal{V}_-}(n) = L^a_{\mathcal{V}_-}(n)$ if the latter's definition above renders a singleton.

These summary views render common refinements and abstractions.

Proposition 1. *Let $\mathcal{V} = \{M_i \mid 1 \leq i \leq k\}$ be given. Then $\mathcal{V}_-$ and $\mathcal{V}_+$ are 3-valued views and, for all $\boldsymbol{t} \in W_\mathcal{V}$ and all i, we have $(\mathcal{V}_-, \boldsymbol{t}) \prec (M_i, t_i) \prec (\mathcal{V}_+, \boldsymbol{t})$.*

Proof. All constraints on transitions and the labelling for 3-valued views are met in $\mathcal{V}_-$ and $\mathcal{V}_+$ by construction: the provisos on n take care of the constraints on 3-valued labels for nominals (items 1 and 2 in Definition 2), and the restriction of the state space $S_\mathcal{V}$ to $W_\mathcal{V}$ and consistency of each M_i take care of the rest. We claim that $Q = \{(t_i, \boldsymbol{t}) \mid \boldsymbol{t} \in W_{\mathcal{V}_s}\}$ shows $(M_i, t_i) \prec (\mathcal{V}_+, \boldsymbol{t})$: (a) Let $t_i \in L^a(q)$. Then $\boldsymbol{t} \in L^a_{\mathcal{V}_+}(q)$ by definition. (b) Let $\boldsymbol{t} \in L^c_{\mathcal{V}_+}(q)$. Then $t_i \in L^c(q)$ by definition. (c) Let $(t_i, \alpha, t'_i) \in R^a$. Then $(\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R^a_{\mathcal{V}_+}$ by definition and $(t'_i, \boldsymbol{t'}) \in Q$. (d) Let $(\boldsymbol{t}, \alpha, \boldsymbol{t'}) \in R^c_{\mathcal{V}_+}$. Then $(t_i, \alpha, t'_i) \in R^c$ by definition and $(t'_i, \boldsymbol{t'}) \in Q$. — The proof of $(\mathcal{V}_-, \boldsymbol{t}) \prec (M_i, t_i)$ is dual to the one just given. ∎

The abstraction $\mathcal{V}_-$ is concrete enough to meaningfully relate to the common views. The common refinement $\mathcal{V}_+$ aids comprehension. Users may want to explore $W_\mathcal{V}$ to generate alternative summaries, as done for $S_\mathcal{V}$ in [33].

Example 8. Figure 6 shows $\mathcal{V}_+$ and $\mathcal{V}_-$ for the $\mathcal{V}$ of Figure 5.

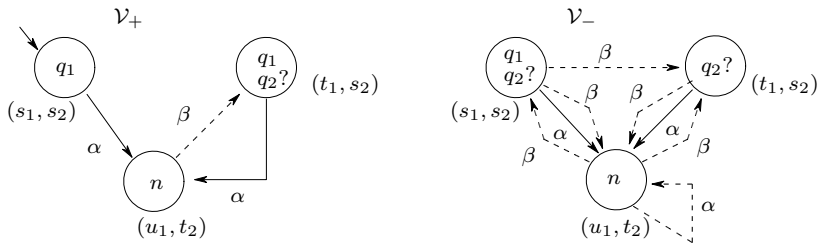

Fig. 6. The summary views $\mathcal{V}_+$ and $\mathcal{V}_-$ for the $\mathcal{V}$ of Figure 5.

7 Related work

Uchitel & Chechik [33] merge modal transition systems with overlapping but different sets of events to obtain a minimal common refinement and suggest user participation to explore common behavior if no minimal common refinement exists. Their algorithms check the consistency of two models and construct a least common refinement if it exists. Their models are more general in that events may differ in views, but less general in that we handle hybrid constraints and compute the space of all consistent tuples. They stress engineering activities in model elaboration, we use static analysis and identify the complexities of the relevant decision problems. Larsen et al. use projective views for a constrained-based proof methodology on modal transition systems [24]. Fitting uses a partial order of experts to constrain the consistency of experts' assertions about the truth and falsity of transitions and state observables in multiple-valued Kripke structures [9]. Chechik et al. endow Fitting's models with a semantics for negation drawn from a De Morgan lattice negotiated among experts. For these models they devise a multiple-valued version of computation tree logic and its symbolic model checking algorithm [6]. Multiple-valued model checking is reducible to 2-valued model checking [6, 5]. Bruns & Godefroid [4] build a query checker for temporal logic which, for a Kripke structure and a query with a hole as input, returns a formula of propositional logic that, when placed into the query's hole, makes the query true for that Kripke structure. Our models for hybrid views assume that views have the same representation, here algebraic signature and type, of a specification. As Jackson points out, this may not always be appropriate [21]. Nentwich et al. developed the tool `xlinkit` that analyzes distributed

XML documents for possible inconsistencies, based on rules written in first-order logic [27]. Guerra [13] proposes a specification framework for software artifacts, where specifications have defaults and allow for exceptions stemming from the reuse or evolution of system demands. In loc. cit. specifications are written in linear-time temporal logic [30] and a non-monotonic semantics for this logic is defined based on default institutions [12]. Foundations for view-based model checking, where models are those of first-order logic with transitive closure, are developed in [19]. In [20] assertion-consistency lattices are defined and argued to be the proper generalization of De Morgan lattices for sound abstraction of multiple-valued models and their checks. For modal transition systems and the modal mu-calculus, the decision problems of this paper have already been defined in [16] and the reduction to satisfiability in the modal mu-calculus for common refinement checks has been stated in [15].

8 Conclusions

We studied finite sets of views, where each view is a 3-valued hybrid model. Such views are suitable models for answers to database queries, knowledge representation, functional requirements etc. We showed that the decision problems "Is there is common refinement satisfying ϕ?" (satisfiability) and "Does ϕ hold for all common refinements?" (validity) are EXPTIME-complete for the hybrid mu-calculus. We gave a PTIME decision procedure for checking whether such a set is consistent in that it has a common refinement (satisfiability for ϕ being *true*). This procedure was used to compute the state space of "summary" views that are informative and consistent common refinements (respectively, abstractions).

Acknowledgments

Sebastian Uchitel was always ready to give his feedback on this line of work. We acknowledge discussions with Glenn Bruns on the static analysis of sets of answers drawn from a set of queries executed on a database view.

References

1. T. Ball, A. Podelski, and S. K. Rajamani. Boolean and Cartesian Abstraction for Model Checking C Programs. In T. Margaria and W. Yi, editors, *Proceedings of TACAS'2001*, volume 2031 of *LNCS*, pages 268–283, Genova, Italy, April 2001. Springer Verlag.
2. G. Bruns and P. Godefroid. Model Checking Partial State Spaces with 3-Valued Temporal Logics. In *Proceedings of the 11th Conference on Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 274–287. Springer Verlag, July 1999.
3. G. Bruns and P. Godefroid. Generalized Model Checking: Reasoning about Partial State Spaces. In *Proceedings of the 11th International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 168–182. Springer Verlag, August 2000.

4. G. Bruns and P. Godefroid. Temporal Logic Query Checking. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, pages 409–417, Boston, Massachusetts, 16-19 June 2001. IEEE Computer Society Press.
5. G. Bruns and P. Godefroid. Model Checking with Multiple-Valued Models. In *Proc. 31st International Colloquium on Automata, Languages and Programming*, Turku, Finland, 12-16 July 2004. To appear.
6. M. Chechik, B. Devereux, A. Gurfinkel, and S. Easterbrook. Multi-Valued Symbolic Model-Checking. *ACM Transactions on Software Engineering and Methodology*, 12(4):1–38, October 2003.
7. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In D. Kozen, editor, *Logic of Programs Workshop*, number 131 in LNCS. Springer Verlag, 1981.
8. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs. In *Proc. 4th ACM Symp. on Principles of Programming Languages*, pages 238–252. ACM Press, 1977.
9. M. Fitting. Many-valued modal logics II. *Fundamenta Informaticae*, 17:55–73, 1992.
10. M. Franceschet and M. de Rijke. Model Checking for Hybrid Logics. In *Proc. of the Workshop on Methods for Modalities*, INRIA Lorraine, Nancy, France, September 2003.
11. P. Godefroid and R. Jagadeesan. On The Expressiveness of 3-Valued Models. In L. D. Zuck, P. C. Attie, A. Cortesi, and S. Mukhopadhyay, editors, *Proc. of 4th Conference on Verification, Model Checking and Abstract Interpretation*, volume 2575 of *LNCS*, pages 206–222, New York, January 2003. Springer Verlag.
12. J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, January 1992.
13. S. Guerra. Distance Functions for Defaults in Reactive Systems. In T. Rus, editor, *Proc. of the 8th International Conference on Algebraic Methodology and Software Technology*, volume 1816 of *Lecture Notes in Computer Science*, pages 26–40, Iowa City, Iowa, May 2000. Springer Verlag.
14. M. Huth. Abstraction and probabilities for hybrid logics. In A. Cerone and A. Di Pierro, editors, *Proc. of the Second Workshop in Quantitative Aspects of Programming Languages*, Electronic Notes in Theoretical Computer Science, page 16, Barcelona, Spain, March 2004. Elsevier & Science Direct. To appear, preliminary version appeared in pre-proceedings.
15. M. Huth. Beyond image-finiteness: labelled transition systems as a Stone space. In *Proc. of the 19th Annual IEEE Symposium on Logic in Computer Science*, Turku, Finland, 13-17 July 2004. IEEE Computer Society. To appear.
16. M. Huth. Refinement is complete for implementations. Invited submission to the special issue for the Third International Workshop on Automated Verification of Critical Systems. Under review, January 2004.
17. M. Huth, R. Jagadeesan, and D. A. Schmidt. Modal transition systems: a foundation for three-valued program analysis. In D. Sands, editor, *Proc. of the European Symposium on Programming*, pages 155–169. Springer Verlag, April 2001.
18. M. Huth, R. Jagadeesan, and D. A. Schmidt. A domain equation for refinement of partial systems. Accepted for publication in the journal Mathematical Structures in Computer Science. In press; 1 February, 2003.
19. M. Huth and S. Pradhan. Model-Checking View-Based Partial Specifications. In S. Brookes and M. Mislove, editors, *Electronic Notes in Theoretical Computer Science*, volume 45. Elsevier Science Publishers, 2001.

20. M. Huth and S. Pradhan. Consistent Partial Model Checking. In M. Mislove, editor, *Electronic Notes in Theoretical Computer Science*, volume 73, page 39. Elsevier Science Publishers, 2003.
21. D. Jackson. Structuring Z Specifications With Views. *ACM Transactions on Software Engineering and Methodology*, 4(4):365–389, October 1995.
22. D. Jackson, I. Shlyakhter, and M. Sridharan. A Micromodularity Mechanism. In *Proc. of the ACM SIGSOFT Conference on the Foundations of Software Engineering/European Software Engineering Conference*, September 2001.
23. K. G. Larsen. Modal Specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, number 407 in Lecture Notes in Computer Science, pages 232–246. Springer Verlag, June 12–14 1989. International Workshop, Grenoble, France.
24. K. G. Larsen, B. Steffen, and C. Weise. A Constraint Oriented Proof Methodology Based on Modal Transition Systems. In E. Brinksma, R. Cleaveland, K. G. Larsen, T. Margaria, and B. Steffen, editors, *Tools and Algorithms for Construction and Analysis of Systems, First International Workshop*, volume 1019 of *Lecture Notes in Computer Science*, pages 17–40, Aarhus, Denmark, 19-20 May 1995. Springer Verlag.
25. K. G. Larsen and B. Thomsen. A Modal Process Logic. In *Third Annual Symposium on Logic in Computer Science*, pages 203–210. IEEE Computer Society Press, 1988.
26. B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1990.
27. C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein. xlinkit: a consistency checking and smart link generation service. *ACM Transactions on Internet Technology*, 2(2):151–185, 2002.
28. C. S. Pasareanu, M. B. Dwyer, and W. Visser. Finding Feasible Counter-examples when Model Checking Abstracted Java Programs. In T. Margaria and W. Yi, editors, *Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *LNCS*, pages 284–298, Genova, Italy, April 2-4 2001. Springer Verlag.
29. D. Peled. *Software Reliability Methods*. Springer Verlag, New York City, New York, 2001.
30. A. Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium on the Foundations of Computer Science*, pages 46–57, 1977.
31. J. P. Quielle and J. Sifakis. Specification and verification of concurrent systems in cesar. In *Proc. of the fifth International Symposium on Programming*, 1981.
32. U. Sattler and M. Vardi. The Hybrid μ-calculus. In R. Goré, A. Leitsch, and T. Nipkov, editors, *Proc. of the First International Joint Conference on Automated Reasoning*, volume 2083 of *Lecture Notes in Computer Science*, pages 76–91, Siena, Italy, 18-23 June 2001. Springer Verlag.
33. S. Uchitel and M. Chechik. Merging Partial Behavioural Models. In *Proc. of the 12th ACM Int'l Symposium on Foundations of Software Engineering*, Newport Beach, California, 31 October - 5 November 2004. To appear.