# A Framework For Compositional Verification of Multi-Valued Systems Via Abstraction-Refinement *

Yael Meller, Orna Grumberg, and Sharon Shoham

Computer Science Department, Technion, Haifa, Israel,
{ymeller,orna,sharonsh}@cs.technion.ac.il

**Abstract.** We present a framework for fully automated compositional verification of $\mu$-calculus specifications over multi-valued systems, based on multi-valued abstraction and refinement.
Multi-valued models are widely used in many applications of model checking. They enable a more precise modeling of systems by distinguishing several levels of uncertainty and inconsistency. Successful verification tools such as STE (for hardware) and YASM (for software) are based on multi-valued models.
Our compositional approach model checks individual components of a system. Only if all individual checks return *indefinite* values, the *parts of the components* which are responsible for these values, are composed and checked. Thus the construction of the full system is avoided. If the latter check is still indefinite, then a *refinement* is needed.
We formalize our framework based on bilattices, consisting of a truth lattice and an information lattice. Formulas interpreted over a multi-valued model are evaluated w.r.t. to the truth lattice. On the other hand, refinement is now aimed at increasing the information level of model details, thus also increasing the information level of the model checking result. Based on the two lattices, we suggest how multi-valued models should be composed, checked, and refined.

## 1 Introduction

Model checking [8] is a successful technique which is widely used for hardware and software verification. It is limited, however, by its high memory requirement, referred to as the *state explosion problem*. Two of the most successful approaches for fighting this problem are abstraction and compositional verification. In [21] the two approaches are joined in the context of 3-valued abstraction. There, each component $M_i$ of a composed system $M$ is lifted into a 3-valued model $M_i\uparrow$ which forms an abstraction of $M$. Model checking a formula $\varphi$ on $M_i\uparrow$ can result in either a definite value *true* or *false*, or an *indefinite* value. In the former case, it is guaranteed that the result is also the value of $\varphi$ on $M$. In the latter case, however, nothing can be deduced about the composed system. If the checks of all individual components return *indefinite* values, then the *parts of the components* which are responsible for these values are identified, composed, and model checked. Thus, the construction of the full composed system is avoided. Finally, if the latter check is still indefinite then a *refinement* is applied to each component separately.

In this work we present a framework generalizing the compositional approach in [21] to general multi-valued models. Our interest in such a framework stems from the fact

* An extended version including full proofs is published as a technical report in [19]

that multi-valued modeling is widely used in many applications of model checking. It is used to model concrete systems more precisely and to define abstract models.

Multi-valued models enable a more precise modeling of systems by distinguishing several levels of uncertainty and inconsistency. For example, 3-valued models are used to describe models with partial information [3]. 4-valued models can model disagreement and their generalizations are used to handle inconsistent views of a system [10, 16]. Temporal logic query checking [6, 5] can also be reduced to multi-valued model checking. Multi-valued models have been widely used for abstraction as well: 3-valued (abstract) models allow proving truth as well as falsity of formulas for the concrete models they represent [13]. The 6-valued models in [1] are tuned to better achieving proofs of falsification. 4-valued models extend 3-valued abstractions by enabling to capture inconsistencies in software [14] and hardware (in STE) [20]. Tools to provide multi-valued verification such as YASM ([14]) and STE ([20]) were developed and successfully applied to software and hardware verification.

Multi-valued models may still suffer from the state explosion problem. Thus, a compositional approach may enhance the verification of larger systems.

The first step we take in formalizing a compositional multi-valued framework is to introduce bilattices [11]. A bilattice defines two lattices over a given set of elements: the *truth lattice* and the *information lattice*, each accompanied with an order. Formulas interpreted over a multi-valued model are evaluated w.r.t. the truth lattice. On the other hand, the relation of "more abstract" over models is based on the information lattice: Roughly, a model $M_2$ is more abstract than a model $M_1$ if values of atomic propositions and transitions in $M_2$ are smaller or equal by the information order than the corresponding values in $M_1$. Consequently, the valuation of a formula in $M_2$ will be smaller or equal by the information order than its value in $M_1$. In fact, since we consider the full $\mu$-calculus, a bidirectional correspondence between transitions of $M_1$ and $M_2$ is needed. To capture this, we define a mixed-simulation relation, based on the information lattice, which turns out to be nontrivial.

Bilattices provide a natural way to identify lattice elements that are *consistent*, meaning they represent some concrete elements of the bilattice (to be formalized later). We can also identify *definite* elements. Those are elements that need not be refined. For simplicity, in the rest of the paper we restrict the discussion to Consistent Partial Distributive Bilattices (CPDB), which consist of exactly all the consistent elements.

Once we establish our setting by means of bilattices, we can fill in the rest of the framework's ingredients. First, we define the notion of *composition* of multi-valued systems. Next, we use the model checking algorithm for multi-valued systems and the alternation-free $\mu$-calculus, suggested in [22]. We also show, in case the checks on individual components are indefinite, how to identify, compose, and check the parts of the models that are needed for the evaluation of the checked formula. As we exemplify later, the resulting composed system is often much smaller than the full composed system. Finally, we develop a heuristics for finding a *criterion for refinement*, in case the result of model checking the composed system is indefinite.

In the framework above we do not discuss the construction of multi-valued abstract models. This is investigated for instance in [15], which presents a methodology for a systematic construction of an abstract model for a given concrete one.

Other works deal with several aspects of multi-valued model checking, but none investigate a compositional approach. Multi-valued symbolic model checking is described in [7]. An alternative definition of (bi)simulation is suggested in [18]. However, there, the relation returns a value, indicating how "close" the models are. Our mixed simulation, on the other hand, returns either true or false, indicating whether one model is an abstraction of the other. A relation similar to our mixed simulation is defined in [1]. Preservation of formulas via simulation is described there in terms of information order. However, they handle a 6-valued framework, rather then a general multi-valued one. Also, they suggest refinement only if the result is the smallest element in the information order, $\bot$. In contrast, we allow refinement for any indefinite value in the bilattice. Bilattices are used also in [15]. However, they are not exploited there for refinement.

To summarize, the main contributions of this paper are:

- We present a framework for fully automated compositional verification of multi-valued systems w.r.t. $\mu$-calculus specifications. The framework is based on multi-valued abstraction-refinement. To the best of our knowledge, this is the first compositional approach for multi-valued model checking.
- We apply our framework to the alternation-free $\mu$-calculus model checking algorithm. In particular, we develop an algorithm for refinement in this context.
- We formalize our framework based on bilattices. This allows to naturally define the consistent and definite elements in the bilattice. It also provides a clear definition of abstraction and refinement in the multi-valued context. It thus provides a better understanding of the multi-valued framework.
- Based on the information order of a bilattice, we define a mixed simulation relation over multi-valued models, preserving $\mu$-calculus specifications.

## 2 Preliminaries

In this section we introduce the concepts of lattices, multi-valued Kripke models, $\mu$-calculus and multi-valued model checking graphs.

**Definition 2.1.** *A* lattice $\mathcal{L}= (L, \leq)$ *consists of a set $L$ with a partial order $\leq$ over $L$, where every finite subset $B$ of $L$ has a least upper bound,* join*, denoted $\sqcup B$, and a greatest lower bound,* meet*, denoted $\sqcap B$, both in $L$. A lattice is* distributive *if $\sqcup$ and $\sqcap$ distribute over each other.*
Examples of lattices are shown in Fig. 1(a),(b),(c) and (e).

**Definition 2.2.** $\mathcal{D}= (L, \leq, \neg)$ *is a* De Morgan algebra *if $(L, \leq)$ is a finite distributive lattice, $\neg : L \rightarrow L$ is a negation function s.t. $\forall a, b$: $\neg\neg a = a$, $a \leq b \Leftrightarrow \neg b \leq \neg a$, and De Morgan laws are satisfied.*

All De Morgan algebras have a greatest (top) element, denoted $true$, and a least (bottom) element, denoted $false$.

### 2.1 Multi-Valued Models and $\mu$-calculus

**Definition 2.3.** *A* Multi-Valued Kripke model *is a 6-tuple $M = \langle \mathcal{L}, AP, S, s_0, R, \Theta \rangle$, where $\mathcal{L}= (L, \leq, \neg)$ is a De Morgan algebra, AP is a set of atomic propositions, $S$ is a finite set of states, $s_0$ is the initial state, $R : S \times S \rightarrow L$ is a mapping of transitions to values in $L$, and $\Theta : AP \rightarrow (S \rightarrow L)$ associates with each atomic proposition $p$, a mapping from $S$ to $L$, describing the truth value of $p$ in each state.*

**Definition 2.4.** *Let $AP$ be a set of atomic propositions and $Var$ a set of propositional variables, s.t. $p \in AP$ and $Z \in Var$. The $\mu$-calculus in negation normal form is defined by:*

$$\varphi ::= p \mid \neg p \mid Z \mid \psi \vee \psi' \mid \psi \wedge \psi' \mid \Box\psi \mid \Diamond\psi \mid \mu Z.\psi \mid \nu Z.\psi$$

Let $L_\mu$ denote the set of all formulas generated by the above grammar. Fixpoint quantifiers $\mu$ and $\nu$ are variable binders. We write $\eta$ for either $\mu$ or $\nu$. We assume formulas are well-named, i.e. no variable is bound more than once in any formula. Thus for a *closed* formula $\varphi \in L_\mu$, every variable $Z$ identifies a unique subformula $fp(Z) = \eta Z.\psi$ of $\varphi$. The set $Sub(\varphi)$ includes all subformulas of $\varphi$.

An *environment* $\mathcal{V} : Var \to (S \to L)$ defines the meaning of free variables. For a variable $Z \in Var$ and a mapping $l : S \to L$, we write $\mathcal{V}[Z = l]$ for the environment that agrees with $\mathcal{V}$ except that it maps $Z$ to $l$.

The multi-valued semantics $\|\varphi\|^M_{\mathcal{V}}$ of a $\mu$-calculus formula $\varphi$ w.r.t. a multi-valued Kripke model $M$ and an environment $\mathcal{V}$ [4] is given as a mapping $S \to L$, in which each state $s$ of $M$ is mapped to a value in $L$ describing the truth value of $\varphi$ in $s$. In the following, lfp, gfp stand for least and greatest fixpoints, respectively, which exist based on [23]. $\|\varphi\|^M_{\mathcal{V}}$ is defined by:

$$\|p\|^M_{\mathcal{V}} = \lambda s.\Theta(p)(s) \qquad \|\neg p\|^M_{\mathcal{V}} = \lambda s.\neg\Theta(p)(s)$$
$$\|\varphi_1 \vee \varphi_2\|^M_{\mathcal{V}} = \lambda s.\|\varphi_1\|^M_{\mathcal{V}} \vee \|\varphi_2\|^M_{\mathcal{V}} \qquad \|\varphi_1 \wedge \varphi_2\|^M_{\mathcal{V}} = \lambda s.\|\varphi_1\|^M_{\mathcal{V}} \wedge \|\varphi_2\|^M_{\mathcal{V}}$$
$$\|\Diamond\varphi\|^M_{\mathcal{V}} = \lambda s.\bigvee_{s' \in S}(R(s,s') \wedge \|\varphi\|^M_{\mathcal{V}}(s')) \quad \|\Box\varphi\|^M_{\mathcal{V}} = \lambda s.\bigwedge_{s' \in S}(\neg R(s,s') \vee \|\varphi\|^M_{\mathcal{V}}(s'))$$
$$\|Z\|^M_{\mathcal{V}} = \mathcal{V}(Z) \qquad \|\nu Z.\varphi\|^M_{\mathcal{V}} = \text{gfp}(\lambda g.\|\varphi\|^M_{\mathcal{V}[Z=g]}) \qquad \|\mu Z.\varphi\|^M_{\mathcal{V}} = \text{lfp}(\lambda g.\|\varphi\|^M_{\mathcal{V}[Z=g]})$$

For closed formulas we drop the environment, and refer to $\|\varphi\|^M$.

## 2.2 Multi-Valued Model-Checking Algorithm

A multi-valued model checking algorithm for a closed $L_\mu$ formula over a multi-valued Kripke model is suggested in [22]. There, multi-valued games are introduced, and a multi-valued model checking problem is translated into a problem of finding the value of a multi-valued game. In this work, we only use the model checking graph (further referred to as *mc-graph*) defined in [22], with its connections to the model checking algorithm.

Let $M$ be a multi-valued Kripke model over $\mathcal{L} = (L, \leq, \neg)$ and $\varphi_0$ a closed $L_\mu$ formula. The *mc-graph* is defined by $G(M, \varphi_0) = (n^0, N, E)$, where $N$ is a set of nodes, $E \subseteq N \times N$ is a set of edges in the graph and $n^0 \in N$ is the initial node. Nodes in the mc-graph are elements of $S \times Sub(\varphi_0)$, denoted $t \vdash \psi$, and $n^0 = s_0 \vdash \varphi_0$. Nodes of type $s \vdash \varphi_0 \vee \varphi_1$ or $s \vdash \Diamond\varphi$ are considered $\vee$-nodes, whereas nodes of type $s \vdash \varphi_0 \wedge \varphi_1$ or $s \vdash \Box\varphi$ are $\wedge$-nodes. Nodes of type $s \vdash Z$ or $s \vdash \eta Z.\varphi$ can be either $\vee$-nodes or $\wedge$-nodes. The edges of the mc-graph are defined by the following rules.

$$\frac{s \vdash \varphi_0 \vee \varphi_1}{s \vdash \varphi_i}\, i \in \{0,1\} \qquad \frac{s \vdash \eta Z.\varphi}{s \vdash Z} \qquad \frac{s \vdash \Diamond\varphi}{t \vdash \varphi}\, R(s,t) \neq false$$
$$\frac{s \vdash \varphi_0 \wedge \varphi_1}{s \vdash \varphi_i}\, i \in \{0,1\} \qquad \frac{s \vdash Z}{s \vdash \varphi} \text{ if } fp(Z) = \eta Z.\varphi \qquad \frac{s \vdash \Box\varphi}{t \vdash \varphi}\, R(s,t) \neq false$$

Every edge $(n, n') \in E$ corresponds to a rule where $n, n'$ are of the form of the upper, respectively lower, part of the rule. If no rule is defined from some node $n$, then there are no outgoing edges from $n$ in the mc-graph. This happens in terminal nodes of

the form $t \vdash p$ or $t \vdash \neg p$, or in terminal nodes of the form $t \vdash \Diamond\varphi$ or $t \vdash \Box\varphi$ where there are no transitions from the state $t$ in the Kripke model.

Each edge in $E$ is associated with a value from $L$: edges that refer to a transition of the model get the value of that transition. The rest get the value $true$. By abuse of notation we use $R(n, n')$ to refer to the value of an edge $(n, n') \in E$.

**Definition 2.5.** *([22]) Let $n \in G$ be a terminal node, $val(n)$ is defined as follows. $val(t \vdash q) = \Theta(q)(t)$, $val(t \vdash \neg q) = \neg\Theta(q)(t)$, $val(t \vdash \Diamond\varphi) = false$ and $val(t \vdash \Box\varphi) = true$.*

In [22] an algorithm for computing a value of nodes on a mc-graph is presented. The algorithm handles the *alternation-free* fragment of $L_\mu$, where no nesting of fixpoints is allowed. Given a mc-graph $G(M, \varphi_0) = (n^0, N, E)$ and a function $val : N \to L$ which maps terminal nodes in $G$ to values in $L$ (Def. 2.5), the algorithm returns a mc-function $\chi : N \to L$ that maps each node to a value from $L$.

**Algorithm 2.6 (mc-algorithm [22]).** *$G$ is partitioned to* Maximal Strongly Connected Components *(MSCCs) and a (total) order on them is determined, reflected by their numbers: $Q_1$,...,$Q_k$. The order fulfills the rule that if $i < j$ then there are no edges from $Q_i$ to $Q_j$. The components are handled by increasing values of $i$. Consider a single $Q_i$. Each node $n \in Q_i$ is associated with a value $\chi(n)$ as follows. For a terminal node $n$, $\chi(n) = val(n)$. For a $\vee$-node $n$ we set $\chi(n)$ to be $\bigvee\{R(n, n') \wedge \chi(n') | R(n, n') \neq false\}$. Similarly, if $n$ is a $\wedge$-node then $\chi(n) = \bigwedge\{\neg R(n, n') \vee \chi(n') | R(n, n') \neq false\}$. If $Q_i$ is a non-trivial MSCC then it contains exactly one fixpoint variable $Z$. In this case, first label the nodes in $Q_i$ with temporary values, $temp(n)$, that are updated iteratively. For nodes of the form $n = s \vdash Z$, initialize $temp(n)$ to $true$ if $Z$ is of type $\nu$, or to $false$ if $Z$ is of type $\mu$ (the rest remain uninitialized). Then apply the previous rules for $\vee$,$\wedge$-nodes until the temporary values do not change anymore. Finally, set $\chi(n) = temp(n)$ for every node $n$ in $Q_i$. Return $\chi$ as the mc-function.*

In [22], the connection between $\chi$ and the model checking problem is proved, by showing that $\chi(n^0) = \|\varphi_0\|^M(s_0)$. In the context of this work we will be interested also in the internal nodes of $G$. We therefore generalize the correspondence between $\chi$ and the multi-valued semantics to *all* nodes in $G$.

For $\psi \in Sub(\varphi_0)$, $\psi^*$ denotes the result of replacing every free occurrence of $Z \in Var$ in $\psi$ by $fp(Z)$. Note that $\psi^*$ is a closed formula, and if $\psi$ is closed then $\psi^* = \psi$.

**Theorem 2.7.** *Let $G(M, \varphi_0)$ be a mc-graph, s.t. $\varphi_0$ is an alternation-free closed $L_\mu$ formula. Let $\chi$ be the mc-function returned by the* ***mc-algorithm****, then for every $s \vdash \psi \in N$, $\chi(s \vdash \psi) = \|\psi^*\|^M(s)$.*

## 3 Bilattices and Partial Bilattices

In this section we introduce bilattices, consider several of their attributes, and define the notion of partial bilattices.

**Definition 3.1.** *[11] A distributive bilattice is a structure $\mathcal{B}$=$(B, \leq_i, \leq_t, \neg)$ s.t.: (1) $\mathcal{B}_i$=$(B, \leq_i)$ is a lattice, $\mathcal{B}_t$=$(B, \leq_t, \neg)$ is a De Morgan algebra; (2) meet($\otimes$), join($\oplus$) of $\mathcal{B}_i$, and meet($\wedge$), join($\vee$) of $\mathcal{B}_t$ are monotone w.r.t. both $\leq_i$ and $\leq_t$; (3) all meets and joins distribute over each other; and (4) negation ($\neg$) is $\leq_i$ monotone.*

The bilattices considered in this work are distributive, thus the use of the term bilattice refers to distributive bilattice. In our context, the relation $\leq_t$ is an order on the "degree of truth". The bottom in this order is denoted by $false$ and the top by $true$. The relation $\leq_i$ is an order on the "degree of information". Thus, if $x \leq_i y$, $y$ gives us at least as much information as $x$ (and possibly more). The bottom in the information order is denoted by $\bot$ and the top by $\top$.

**Theorem 3.2.** *[11] Let $\mathcal{D}$= $(D, \leq, \neg)$ be a De Morgan algebra, and $\mathcal{B}(\mathcal{D})$ be a structure $(D \times D, \leq_i, \leq_t, \neg)$ s.t. (1) $\langle a, b\rangle \leq_i \langle c, d\rangle \triangleq a \leq c$ and $b \leq d$; (2) $\langle a, b\rangle \leq_t \langle c, d\rangle \triangleq a \leq c$ and $d \leq b$; and (3) $\neg\langle a, b\rangle \triangleq \langle b, a\rangle$. Then, $\mathcal{B}(\mathcal{D})$ is a distributive bilattice. Furthermore, every distributive bilattice is isomorphic to $\mathcal{B}(\mathcal{D})$ for some De Morgan algebra $\mathcal{D}$.*

Intuitively, for a De Morgan algebra $\mathcal{D}$, an element $\langle x, y\rangle$ of $\mathcal{B}(\mathcal{D})$ is interpreted as a value whose "degree of truth" is $x$ and "degree of falsity" is $y$. If we view $\mathcal{D}$ as a concrete truth domain, $\mathcal{B}(\mathcal{D})$ can be viewed as its abstract truth domain. Given an element $c \in D$, $\langle x, y\rangle \in D \times D$ approximates $c$ if $x$ is no more true than $c$, and $y$ is no more false than $c$. Thus, $\langle c, \neg c\rangle$ is the best approximation of $c$, and $\langle x, y\rangle$ approximates $c$ if $\langle x, y\rangle \leq_i \langle c, \neg c\rangle$. We say that $\langle x, y\rangle \in D \times D$ is *consistent* if $\langle x, y\rangle \leq_i \langle c, \neg c\rangle$ for some $c \in D$. Thus $\langle x, y\rangle$ is consistent iff $y \leq \neg x$ (defined similarly in [15]). We say that $\langle x, y\rangle \in D \times D$ is *definite* if $\langle c, \neg c\rangle \leq_i \langle x, y\rangle$ for some $c \in D$. Thus $\langle x, y\rangle$ is definite iff $y \geq \neg x$. If $\langle x, y\rangle \in D \times D$ is definite and consistent, then $\langle x, y\rangle = \langle c, \neg c\rangle$ for some $c \in D$.

*Example 3.3.* Fig. 1(a),(b) present an example of the distributive bilattice for the 4-valued Belnap structure ([2]). This bilattice is isomorphic to the bilattice $\mathcal{B}(\mathcal{D})$ created from the 2-valued De Morgan algebra $\mathcal{D}$= $(\{T, F\}, \leq, \neg)$, where $F \leq T$, $\neg T = F$. Thus, $t \triangleq \langle T, F\rangle$, $f \triangleq \langle F, T\rangle$, $\top \triangleq \langle T, T\rangle$ and $\bot \triangleq \langle F, F\rangle$. $t$, $f$ are best approximations of $T$, resp. $F$. $\top$, representing maximal degree of truth and falsity, is inconsistent. $t$,$f$ and $\top$ are definite elements. $\bot$ is indefinite.

When referring to a bilattice $\mathcal{B}$, we sometimes implicitly refer to the structure $\mathcal{B}(\mathcal{D})$ isomorphic to $\mathcal{B}$ (which exists by Thm. 3.2). In particular, we use '$\leq$' to denote the order on the elements in the De Morgan algebra $\mathcal{D}$ of $\mathcal{B}(\mathcal{D})$.

**Definition 3.4.** *$\mathcal{P}$= $(B, \leq)$ is a* partial lattice *if it is a lattice, except that join is not always defined. A* partial distributive bilattice *is a structure $\mathcal{P}$= $(B, \leq_i, \leq_t, \neg)$ defined similarly to a distributive bilattice (Def. 3.1), except that $\mathcal{P}_i$= $(B, \leq_i)$ is a partial lattice, and requirements (2) and (3) hold for join of $\mathcal{P}_i$ only if it is defined.*

**Definition 3.5.** *Let $\mathcal{B}(\mathcal{D})$= $\langle D \times D, \leq_i, \leq_t, \neg\rangle$ be a bilattice, and let $P \subseteq D \times D$ be the set of all consistent elements in $\mathcal{B}(\mathcal{D})$. Then $\mathcal{P}(\mathcal{B})$= $\langle P, \leq_i, \leq_t, \neg\rangle$ is the* consistent structure *induced by $\mathcal{B}(\mathcal{D})$, where $\leq_t$, $\leq_i$ and $\neg$ in $\mathcal{P}(\mathcal{B})$ are as in $\mathcal{B}(\mathcal{D})$, restricted to $P$.*

**Theorem 3.6.** *Let $\mathcal{B}(\mathcal{D})$= $\langle D \times D, \leq_i, \leq_t, \neg\rangle$ be a bilattice, and let $\mathcal{P}(\mathcal{B})$= $\langle P, \leq_i, \leq_t, \neg\rangle$ be the consistent structure induced by it, then $\mathcal{P}(\mathcal{B})$ is a partial distributive bilattice.*

We refer to consistent structures, which, by Thm. 3.6, are also partial distributive bilattices, as *consistent partial distributive bilattices* (CPDB). Note that in CPDBs we do not have $\top$. $\bot$, $true$ and $false$ always exist. Note further that for CPDBs, the set of maximal elements w.r.t. $\leq_i$ is exactly the set of definite elements, all of the form $\langle c, \neg c\rangle$ for some $c \in D$.

**Theorem 3.7.** *Let $\mathcal{B}$= $\langle B, \leq_i, \leq_t \neg\rangle$ be either a distributive bilattice or a CPDB, and let $a, b \in B$ be definite values. Then $a \wedge b$, $a \vee b$ and $\neg a$ are definite as well.*

*Example 3.8.* Examples of CPDBs appear in Fig. 1. The CPDB induced by the bilattice of the Belnap structure is described in Fig. 1(a) and (b), as the un-boxed elements, which are all the consistent elements. This CPDB is isomorphic to the standard 3-valued structure, where ? $\triangleq \bot$, T $\triangleq t$ and F $\triangleq f$. The structure 3×3 is defined by the CPDB in Fig. 1(e) and (f). This CPDB is isomorphic to the CPDB induced by the bilattice $\mathcal{B}(\mathcal{D})$ created from the 2-views De Morgan algebra $\mathcal{D}$= $(\{T, F\} \times \{T, F\}, \leq, \neg)$, where $\leq$ and $\neg$ are defined bitwise. That is, for $a_1a_2, b_1b_2 \in \{T, F\} \times \{T, F\}$, $a_1a_2 \leq b_1b_2$ iff $a_1 \leq b_1$ and $a_2 \leq b_2$. Also, $\neg a_1a_2 \triangleq \neg a_1\neg a_2$. The 3×3 structure represents two different views, which may be contradictory (e.g. TF). However, such elements should not be confused with inconsistent elements in $\mathcal{B}(\mathcal{D})$ such as $\langle TT, TT\rangle$.
The consistent elements of $\mathcal{B}(\mathcal{D})$ are mapped into pairs over {T,F,?} in the 3×3 structure. E.g., $\langle TF, FF\rangle$ is represented by T? and $\langle TT, FF\rangle$ is represented by TT. The resulting structure contains both representations of the elements of the concrete 2-views domain (e.g. TT), and their approximations (e.g. T?).

Multi-valued Kripke models as well as the semantics of $L_\mu$ formulas and mc-graphs are defined over a De Morgan algebra $\mathcal{L}$. These definitions can easily be extended to a multi-valued structure, which is either a distributive bilattice or a CPDB. Thus, we have both information and truth lattices. In this case, the lattice $\mathcal{L}$ used in the multi-valued semantics is the truth lattice. For simplicity, in the rest of this work we use CPDBs.

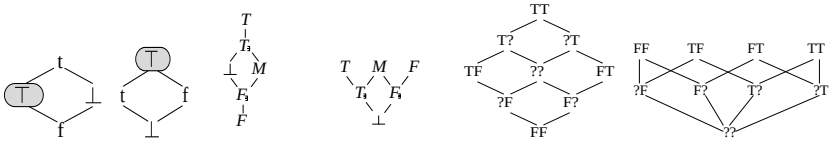


**Fig. 1.** Truth (a) and information (b) orders of 4-valued Belnap structure; Truth (c) and information (d) orders of 6-valued structure; Truth (e) and information (f) orders of 3×3 structure; Boxed nodes are inconsistent

## 4 Mixed Simulation and Refinement of Multi-Valued Models

In this section we define a mixed simulation relation between two multi-valued Kripke models $M_1$ and $M_2$, and present a refinement algorithm based on the multi-valued model checking algorithm. We first define a relation between two multi-valued Kripke models, both defined over the same CPDB $\mathcal{B}$, which guarantees preservation of $L_\mu$ formulas w.r.t the multi-valued semantics. The relation is defined by means of the information order. Intuitively, it identifies the fact that $M_2$ contains less information than $M_1$. Thus, $M_2$ is an abstraction of $M_1$.

**Definition 4.1.** *Let $M_i = \langle \mathcal{B}, AP, S_i, s_0^i, R_i, \Theta_i\rangle$ for $i \in \{1, 2\}$ be multi-valued Kripke models. $H \subseteq S_1 \times S_2$ is a* mixed simulation *from $M_1$ to $M_2$ if $(s_1, s_2) \in H$ implies:*
*1.* $\forall p \in AP : \Theta_2(p)(s_2) \leq_i \Theta_1(p)(s_1)$.
*2.* $\forall t_1 \in S_1$ *s.t.* $R_1(s_1, t_1) \neq false$ $\exists t_2 \in S_2$ *s.t.* $(t_1, t_2) \in H$ *and* $R_2(s_2, t_2) \leq_i R_1(s_1, t_1)$.
*3.* $\forall t_2 \in S_2$ *s.t.* $R_2(s_2, t_2) \not\leq_i false$ $\exists t_1 \in S_1$ *s.t.* $(t_1, t_2) \in H$ *and* $R_2(s_2, t_2) \leq_i R_1(s_1, t_1)$.
*If exists a mixed simulation $H$ s.t. $(s_0^1, s_0^2) \in H$, then $M_2$ abstracts $M_1$, denoted $M_1 \preceq M_2$.*

Note that requirements (2) and (3) are not symmetrical. By requirement (2), every transition in $M_1$ has a representation in $M_2$, whereas by requirement (3), only transitions in $M_2$ s.t. $R_2(s_2, t_2) \not\leq_i false$ have a representation in $M_1$. These requirements are similar to the requirements of mixed simulation in the 3-valued case ([12, 9]). There, every *may* transition in $M_1$ has a representation in $M_2$, and every *must* transition in $M_2$ has a representation in $M_1$. In the multi-valued case transitions which are may and not must are transitions s.t. $R(s,t) \leq_i false$.

**Theorem 4.2.** *Let $H \subseteq S_1 \times S_2$ be a mixed simulation relation from $M_1$ to $M_2$, and let $\varphi$ be a closed $L_\mu$ formula. Then for every $(s_1, s_2) \in H$, $\|\varphi\|^{M_2}(s_2) \leq_i \|\varphi\|^{M_1}(s_1)$.*

The mixed simulation relation can be used to describe the relation between a concrete model, $M_c$, and its abstraction, $M_A$: $M_c \preceq M_A$, where $M_c$ is defined over $\mathcal{D}$ and $M_A$ is defined over $\mathcal{P}(\mathcal{B}(\mathcal{D}))$. This is because $M_c$ can be interpreted as a model over $\mathcal{P}(\mathcal{B}(\mathcal{D}))$, where the values are all definite (by Thm. 3.7 the semantics is maintained).

Given an abstract model, the information order enables us to capture the notion of a model checking result being "not good enough", namely, a result that needs to be refined. This is a result that does not give us the most information possible. That is, it is indefinite.

Let $M_c$ be a concrete model over $\mathcal{D}$, and let $M_A$ be an abstract model for it over $\mathcal{P}(\mathcal{B}(\mathcal{D}))$, i.e. $M_c \preceq M_A$, s.t. $M_A$ should be refined. Our refinement consists of two parts. First, we choose a criterion for model refinement. Then, based on the criterion, the model is refined by either increasing the information level of some transition or of an atomic proposition in some state, or by splitting states. These refinement steps are similar to the refinement steps in [17]. The refinement is applied directly on the mc-graph. In fact, it suffices to refine the indefinite subgraph, where the mc-graph is pruned in definite nodes.

In the rest of the section we study choosing a criterion for model refinement. For a mc-function $\chi : N \to L$, given that $\chi(n^0)$ is indefinite, our goal in the refinement is to find and eliminate at least one of the reasons of the indefinite result. The criterion for the refinement is obtained from a failure node. This is a node $n = s \vdash \varphi \in N$ s.t. (1) $\chi(n)$ is indefinite; (2) $\chi(n)$ affects $\chi(n_0)$; and (3) $\chi(n)$ can be changed by increasing the information level of either an atomic proposition in $s$ or some transition from $s$.
(3) means that $n$ *itself* is responsible for introducing (some) uncertainty. (1) and (2) require that this uncertainty is relevant to $\chi(n^0)$.

We adapt the **mc-algorithm** (Algo. 2.6) to remember for each node whose value is indefinite a failure node and a failure reason. The failure node and reason of $n^0$ will be used for refining the model. For a terminal node $n$, if $\chi(n)$ is indefinite, the failure node and reason attached to it are the node itself. To handle nonterminal nodes, we define an auxiliary function $f : N \to L$ that keeps for each node $n \in N$ its most updated value in the algorithm: If $\chi(n)$ is already defined, then $f(n) = \chi(n)$. Otherwise, if $temp(n)$ is defined, then $f(n) = temp(n)$.

Let $n$ be a node for which $f(n)$ has been updated last. If $f(n)$ is definite, then no failure node and reason are attached to it. If $f(n)$ is indefinite, do the following:

1. If $n$ is a $\vee$-node, find node $n'$ with $R(n,n') \neq false$, for which the following hold:
   (a) $\forall n'' \in N$ where $n' \neq n''$ and $R(n,n'') \neq false$, $(R(n,n'') \wedge f(n'')) \leq_t (R(n,n') \wedge f(n'))$ or $(R(n,n'') \wedge f(n''))$ and $(R(n,n') \wedge f(n'))$ are uncomparable.

(b) $R(n,n') \wedge f(n')$ is indefinite.

Intuitively, for some $n'$, if requirement (a) holds then $R(n,n') \wedge f(n')$ is maximal, and thus affects $f(n)$. Requirement (b) ensures that it is possible to refine $R(n,n')$ or $f(n')$. For the given node $n$ and a chosen node $n'$ satisfying (a),(b) define a failure node and reason for $n$ as follows:

*i* If $f(n')$ is definite or $R(n,n') \lesssim_t f(n')$, then $n$ is the failure node, and the edge $(n,n')$ is the failure reason.

*ii* If $R(n,n')$ is definite or $f(n') \lesssim_t R(n,n')$, then the failure node and reason of $n$ are those of $n'$.

*iii* Otherwise, arbitrarily choose either $n$ as a failure node and the edge as a failure reason, or the failure node and reason of $n'$ as the failure node and reason of $n$.

2. The case where $n$ is a $\wedge$-node is dual, where instead of searching for a maximal $R(n,n') \wedge f(n')$, we now try to find a minimal $\neg R(n,n') \vee f(n')$.

Definite values are closed under $\neg$, $\wedge$ and $\vee$ (Thm. 3.7), thus if a node is given an indefinite value, this indefinite value results from an indefinite value of either a son $n'$ of $n$, or an edge from $n$. For example, consider case 1(*i*). If $f(n')$ is definite, then $R(n,n')$ is necessarily indefinite (Thm. 3.7). Similarly, if $R(n,n') \lesssim_t f(n')$, then $R(n,n') \wedge f(n') = R(n,n')$, which again, means that $R(n,n')$ is indefinite. Either way, $R(n,n')$ can be refined and is therefore a failure reason. The correctness of the failure search is formalized by the following lemma.

**Lemma 4.3.** *For every node $n$, if $f(n)$ is given an indefinite value, then there exists $n'$ s.t. $R(n,n') \neq false$, which satisfies requirements (a),(b). Furthermore, $f(n')$ is already defined at that time. In addition, if the updating of failure node and reason of $n$ is based on $n'$, then $n'$ also has a failure node and reason.*

A failure node and reason for $n$ is updated *every time* $f(n)$ is updated. Thus, when the **mc-algorithm** terminates, for every $n$, if $\chi(n)$ is indefinite, then the failure node and reason for $n$ is based on $\chi$. Altogether there are two cases in which we consider the node itself as a failure node. The first case is when the node is a terminal node whose value is indefinite, for which the failure reason is clear. The second case is when the node has an indefinite edge to $n'$ which is the failure reason. In this case $n$ is the failure node since refining the value of the edge may change the value of $n$. The failure reason translates to an atomic proposition with an indefinite value in the first case, and to an indefinite transition in the second case. Note that the algorithm is heuristic in the sense that it does not guarantee that all possible refinements of the failure node and reason will increase the information level of the result. It greedily searches for a failure node and reason which is most likely to increase the result w.r.t. the information order.

*Example 4.4.* Consider the mc-graph in Fig. 3(b). For the node $n_0$ marked $(s_2,t_2) \vdash \Diamond o$ there are three possible failure nodes and reasons. The first is $n_0$ itself being the failure node and the edge to node $n_3$ marked $(s_3,t_2) \vdash o$ being the reason. The second is node $n_1$ marked $(s_2,t_2) \vdash o$ being the failure node and reason, and the third is node $n_0$ itself and the edge to $n_1$ being the reason.

Recall that refinement is done by either increasing the information level of some transition or atomic proposition, or by splitting states. The information lattice of the

underlying multi-valued structure is finite. Thus, if the underlying concrete model is finite, then there is a finite number of refinement steps possible. We conclude:

**Lemma 4.5.** *If $M_c$ is finite then in a finite number of refinement steps the model checking result will be the same as the one in the underlying concrete model, $M_c$.*

## 5 Partial Model Checking and Subgraphs

In this section we investigate properties of the **mc-algorithm** (Algo. 2.6). In particular, we present sufficient conditions under which a subgraph of a mc-graph can be valuated "correctly" (to be formally defined later) without considering the full mc-graph. In the rest of the section, $G$ denotes a multi-valued mc-graph over $\mathcal{B}= (L, \leq_i, \leq_t, \neg)$.

**Definition 5.1.** *Let $G$ be a mc-graph and let $f : N \to L$ be a function. For a non-terminal node $n \in G$, and two nodes $n', n''$ sons of $n$, $n'$* covers *$n''$ under $f$ w.r.t $n$, if one of the following holds:*

- *$n$ is a $\wedge$-node and: (1) $(\neg R(n, n') \vee f(n')) \leq_t (\neg R(n, n'') \vee f(n''))$; and (2) $\forall v', v'' \in L$ : if $f(n') \leq_i v'$ and $f(n'') \leq_i v''$ then $(\neg R(n, n') \vee v') \leq_t (\neg R(n, n'') \vee v'')$.*
- *$n$ is a $\vee$-node and: (1) $(R(n, n'') \wedge f(n'')) \leq_t (R(n, n') \wedge f(n'))$; and (2) $\forall v', v'' \in L$ : if $f(n') \leq_i v'$ and $f(n'') \leq_i v''$ then $(R(n, n'') \wedge v'') \leq_t (R(n, n') \wedge v')$.*

Intuitively, a son $n'$ covers a son $n''$ in the sense that if $f$ defines the value of the sons, then it suffices to take into account $n'$ (and ignore $n''$) in order to determine the value of the node $n$. In our setting, $f$ will sometimes only provide a lower bound w.r.t. $\leq_i$ on the value of the nodes. However, the second requirement ensures that the covering holds for every $f' \geq_i f$ as well. Note that the notion of covering defines a partial order on the nodes of the mc-graph. As a result, for every covered node $n''$ there exists a covering node $n'$ s.t. $n'$ is non-covered.

*Example 5.2.* Consider the mc-graph in Fig. 3(b). Assume the underlying structure is the $3 \times 3$ structure (Fig. 1(e), (f)). The values of the edges are $R(n_0, n_1) = ?T$ and $R(n_0, n_2) = ??$. Let $f(n_1) = ?T$ and $f(n_2) = F?$. We show that $n_2$ is covered by $n_1$ under $f$. Clearly, requirement (1) holds ($F?\wedge?? \leq_t ?T\wedge?T$). For requirement (2), we need to show that $\forall v_1, v_2 \in L$ : if $f(n_1) \leq_i v_2$ and $f(n_2) \leq_i v_2$ then $(R(n_0, n_2) \wedge v_2) \leq_t (R(n_0, n_1) \wedge v_1)$. Specifically, we need to show that $\forall v_1 \in \{?T, TT, FT\}, \forall v_2 \in \{F?, FF, FT\}$: $v_2 \wedge ?? \leq_t v_1 \wedge ?T$, which obviously holds.

In the example, the value given to $n_0$ based on all its sons is the same as if the son $n_2$ had not been considered. We will next exploit this property.

**Definition 5.3.** *Let $G$ be a mc-graph and $\chi$ its mc-function. A subgraph $G'$ of $G$ is* closed *if every node $n$ in $G'$ is either terminal in $G'$, or $G'$ contains all non-covered sons of $n$ under $\chi$ and corresponding edges.*

Let $G$ be a mc-graph, $\chi$ its mc-function, and $\chi_I : N \to L$ a partial mc-function. $\chi_I$ is *correct* w.r.t. $\chi$ if for every $n \in G$, if $\chi_I(n)$ is defined, then $\chi_I(n) = \chi(n)$.

**Theorem 5.4.** *Let $G$ be a mc-graph and $\chi$ its mc-function. Consider a closed subgraph $G'$ of $G$ with a partial mc-function $\chi_I$ which is correct w.r.t. $\chi$ and defined over (at least) all terminal nodes in $G'$. Then applying the **mc-algorithm** on $G'$ with $\chi_I$ as an initial valuation (replacing $val$) results in a mc-function $\chi'$ of $G'$ s.t. for every $n$ in $G'$, $\chi'(n) = \chi(n)$.*

## 6 Compositional Model Checking

In this section we define the composition of two models, and describe an algorithm for model checking the composed system, without fully constructing it.

In compositional model checking the goal is to verify a formula $\varphi$ on a composed system $M_1||M_2$. In our setting $M_1$ and $M_2$ are multi-valued Kripke models defined over the same CPDB $\mathcal{B}{=}\ (L, \leq_i, \leq_t, \neg)$. The models synchronize on the joint labelling of the states. In the following, for $i \in \{1,2\}$ we assume that every (abstract) model $M_i$ has an underlying concrete model $M_i^c$ s.t. $M_i^c \preceq M_i$. Let $M_i = \langle \mathcal{B}, AP_i, S_i, s_0^i, R_i, \Theta_i \rangle$, we use $\bar{i}$ to denote the remaining index in $\{1,2\} \setminus \{i\}$.

**Definition 6.1.** *Let $s_1 \in S_1$, $s_2 \in S_2$ be states in $M_1$ and $M_2$ resp. Then, $s_1$, $s_2$ are* weakly composable *if for every $p \in AP_1 \cap AP_2 : \Theta_1(p)(s_1) \oplus \Theta_2(p)(s_2)$ is defined.*

Note that $\oplus$ might be undefined since $\mathcal{B}$ is a CPDB (Def. 3.5). Intuitively, if $\oplus$ is defined, then the composition of the states is consistent on all atomic propositions.

**Definition 6.2.** *States $s_1 \in S_1$, $s_2 \in S_2$ are* composable *if they are weakly composable, and for every $p \in AP_1 \cap AP_2 : \Theta_1(p)(s_1)$ and $\Theta_2(p)(s_2)$ are definite.*

In fact, since the definite values in CPDB are highest in the information order, if $s_1$ and $s_2$ are composable, then for every $p \in AP_1 \cap AP_2$, $\Theta_1(p)(s_1) = \Theta_2(p)(s_2)$.

We say that $M_1$ and $M_2$ are *composable* if their initial states are composable.

**Definition 6.3.** *Let $M_1$ and $M_2$ be composable models. Their composition, denoted $M_1||M_2$, is the multi-valued Kripke model $M = \langle \mathcal{B}, AP, S, s_0, R, \Theta \rangle$, where $AP = AP_1 \cup AP_2$, $s_0 = (s_0^1, s_0^2)$, $S = \{(s_1, s_2) \in S_1 \times S_2 | s_1, s_2$ are weakly composable$\}$. For each $(s_1, s_2), (t_1, t_2) \in S$: If $t_1, t_2$ are composable, $R((s_1, s_2), (t_1, t_2)) = R_1(s_1, t_1) \wedge R_2(s_2, t_2)$. Otherwise, if $t_1, t_2$ are weakly composable, $R((s_1, s_2), (t_1, t_2)) = R_1(s_1, t_1) \wedge R_2(s_2, t_2) \wedge \bot$. For each $(s_1, s_2) \in S$ and $p \in AP$: if $p \in AP_1 \cap AP_2$ then $\Theta(p)(s_1, s_2) = \Theta_1(p)(s_1) \oplus \Theta_2(p)(s_2)$, if $p \in AP_i \setminus AP_{\bar{i}}$ then $\Theta(p)(s_1, s_2) = \Theta_i(p)(s_i)$.*

The definition of the states in the composed model enables composition of states that are weakly composable but not composable. Such states do not exist in a composed concrete model (since the values of all atomic propositions in a concrete model are maximal w.r.t. $\leq_i$). However, they might exist when considering a composed *abstract* model. Unlike composable states, the weakly composable states in a composed abstract model might not have any corresponding state in the underlying concrete model. This is because in the concrete model, where the information level of some atomic propositions increases, the states might disagree on some $p$ in their joint labelling.

Even though we are enabling weakly composable states which might not exist in the underlying concrete model, we want the abstract composed model to be an abstraction of the concrete composed model (i.e., we want to maintain a mixed simulation relation between these models). This is done with the definition of $R$. In the case where the target states are composable, the definition of $R$ is immediate. If the target states are weakly composable but not composable, then we want to take into account the possibility that the transition does not exist. Defining its value to be $\bot$ achieves this goal. However, we can in fact guarantee more than that (in terms of the information order) by taking

the meet w.r.t. $\leq_t$ with $\bot$. This ensures that the value of the composed transition is no "more true" than $\bot$, but may be "more false" than $\bot$ and thus more informative. More precisely, consider the CPDB $\mathcal{P}(\mathcal{B}(\mathcal{D}))$ isomorphic to $\mathcal{B}$, where $\mathcal{D}{=}\ (D, \leq, \neg)$ is a De Morgan algebra. Then $\bot$ is defined as $\langle d_\bot, d_\bot \rangle \in D \times D$ where for every $a \in D. d_\bot \leq a$. Thus, for every $\langle a, b \rangle \in D \times D$, $\langle a, b \rangle \wedge \bot{=}\langle d_\bot, b \rangle$, which means that the falsity level of $\langle a, b \rangle$ is preserved, whereas the truth level is minimal.

Allowing weakly composable states gives freedom to the user when abstracting each of the models, as all atomic propositions can be abstracted. In contrast, in [21], where composition of 3-valued models is discussed, joint labelling cannot be abstracted, thus all composable states in the abstract model represent composable states in the concrete model. There is a tradeoff presented with this freedom. On the one hand, the user can define a very coarse abstraction in each of the separate models. On the other hand, the abstract composed model might now include more states that do not represent any state in the concrete model.

From now on we fix $AP$ to be $AP_1 \cup AP_2$.

Next, we define *lifting* of models for the purpose of compositional verification. The idea is to view each model $M_i$ as an abstraction of $M_1||M_2$.

**Definition 6.4.** *The lifted model of* $M_i = \langle \mathcal{B}, AP_i, S_i, s_0^i, R_i, \Theta_i \rangle$ *is* $M_i\uparrow = \langle \mathcal{B}, AP, S_i, s_0^i, R_i\uparrow, \Theta_i\uparrow \rangle$ *where: for every* $s_i, t_i \in S_i$*:* $R_i\uparrow(s_i, t_i) = R_i(s_i, t_i) \wedge \bot$*. For every* $s_i \in S_i$*,* $p \in AP$*: if* $p \in AP_i$ *then* $\Theta_i\uparrow(p)(s_i) = \Theta_i(p)(s_i)$*. If* $p \in AP \setminus AP_i$ *then* $\Theta_i\uparrow(p)(s_i) = \bot$*.*

The value of each literal over $AP \setminus AP_i$ in each state of $M_i\uparrow$ is minimal w.r.t. the $\leq_i$ order ($\bot$). Indeed, its value in $M$ will be determined by $M_{\bar{i}}$. In addition, each transition of $M_i$ is also uncertain, in the sense that it cannot be "more true" than $\bot$. This is because in the composition it might be removed if a matching transition does not exist in $M_{\bar{i}}$.

**Theorem 6.5.** *For every* $i \in \{1, 2\}$*,* $M_1||M_2 \preceq M_i\uparrow$*. The mixed simulation relation* $H \subseteq S \times S_i$ *is given by* $\{((s_1, s_2), s_i) | (s_1, s_2) \in S\}$*.*

Since each $M_i\uparrow$ abstracts $M_1||M_2$, we are able to first consider each component separately: Theorem 4.2 ensures that for every closed $L_\mu$ formula $\varphi$, $\|\varphi\|^{M_i\uparrow} \leq_i \|\varphi\|^{M_1||M_2}$. In particular, $\|\varphi\|^{M_1\uparrow} \oplus \|\varphi\|^{M_2\uparrow}$ is defined, and a definite result on one of the components suffices to determine a definite value on $M_1||M_2$. Note that a definite value on $M_1||M_2$ can be achieved even if both $\|\varphi\|^{M_i\uparrow}$ are indefinite, but $\|\varphi\|^{M_1\uparrow} \oplus \|\varphi\|^{M_2\uparrow}$ is definite.

A more typical case is when the valuation of $\varphi$ on both $M_1\uparrow$ and $M_2\uparrow$ is indefinite. This reflects the fact that $\varphi$ depends on both components and on their synchronization. Typically, such a result requires some refinement of the abstract model. Considering the composition of the two components is a refinement of the lifted models. Still, having considered each component separately can guide us into focusing on the places where we indeed need to consider the composition of the components.

We use the mc-graphs of $M_1\uparrow$ and $M_2\uparrow$ for building a subgraph for $M_1||M_2$, and by that avoid building the full composed model. The mc-graphs of the two components present all the information gained from model checking each component. To resolve the indefinite result, we first try to compose the parts of the mc-graphs which might be needed to determine the value of the formula.

**Definition 6.6.** *For every $i \in \{1,2\}$ let $G_i = (n_i^0, N_i, E_i)$ be the mc-graph of $M_i\uparrow$, with $\chi_i$ its mc-function. $\chi_f : N_1 \times N_2 \to L$ is defined by $\chi_f(n_1, n_2) = \chi_1(n_1) \oplus \chi_2(n_2)$. $E_f : (N_1 \times N_2) \times (N_1 \times N_2) \to L$ is defined as follows. Let $n_i' = (s_i' \vdash \varphi') \in N_i$, then $E_f((n_1, n_2), (n_1', n_2')) = R_1(s_1, s_1') \wedge R_2(s_2, s_2')$ if $s_1'$ and $s_2'$ are composable, and $E_f((n_1, n_2), (n_1', n_2')) = R_1(s_1, s_1') \wedge R_2(s_2, s_2') \wedge \bot$ if $s_1'$ and $s_2'$ are weakly composable but not composable.*

Let $G = (n^0, N, E)$ be a mc-graph and let $f : N \to L$ and $e : N \times N \to L$ be two functions. For a non-terminal node $n$ in $G$, and two nodes $n'$ and $n''$ which are sons of $n$, we abuse the notion of covering (Def. 5.1) and say that $n'$ covers $n''$ under $f$ *and* $e$ w.r.t. $n$, if $n'$ covers $n''$ under $f$ w.r.t. $n$ when $e$ replaces the transition relation $R$.

**Definition 6.7.** ***(Product Graph)*** *For every $i \in \{1,2\}$ let $G_i = (n_i^0, N_i, E_i)$ be the mc-graph of $M_i\uparrow$, with an initial node $n_i^0 = (s_i^0 \vdash \varphi) \in N_i$. Also let $\chi_i$ be the mc-function of $G_i$. The product graph of $G_1$ and $G_2$, denoted $G_\| = (n_\|^0, N_\|, E_\|)$, is defined as the least graph that obeys the following:*

- *$n_\|^0 = (s_1^0, s_2^0) \vdash \varphi$ is the initial node in $G_\|$.*
- *Let $n_1 = s_1 \vdash \psi$, $n_2 = s_2 \vdash \psi$ be s.t. $(n_1, n_2) \in N_\|$, and $\chi_f(n_1, n_2)$ is indefinite. Then for every $n_1' = (s_1' \vdash \psi') \in N_1$ and $n_2' = (s_2' \vdash \psi') \in N_2$, if the following holds:*
  *(1) $s_1'$, $s_2'$ are weakly composable; (2) $E_1(n_1, n_1') \neq false$ and $E_2(n_2, n_2') \neq false$; and (3) $(n_1', n_2')$ is not covered under $\chi_f$ and $E_f$ w.r.t. $(n_1, n_2)$.*
  *Then: (a) $(n_1', n_2') \in N_\|$; and (b) $E_\|((n_1, n_2), (n_1', n_2')) = E_f((n_1, n_2), (n_1', n_2'))$.*

**Lemma 6.8.** *Let $G_\|$ be the product graph defined above. $\forall n \in N_\|$, $\chi_f(n)$ is defined.*

Note that the value of the edges in $G_\|$ is identical to their value in the composed model. This is because the product graph already refers to the complete system $M_1||M_2$. In contrast, the values of the edges in the mc-graphs of each component are all smaller or equal by truth order than $\bot$.

The product graph is constructed by a top-down traversal on the mc-graphs of the two models, where, starting from the initial node, nodes that share the same formulas and whose states are weakly composable, will be considered. Whenever two non-terminal nodes $n_1$, $n_2$ are composed, if $\chi_f(n_1, n_2)$ is indefinite, then the outgoing edges are computed as the product of their outgoing edges, restricted to weakly composable nodes. In particular, this means that if a node in one mc-graph has no matching node in the other, then it will be omitted from the product graph. After computing all legal sons based on the outgoing edges, the nodes which are covered under $\chi_f$ will be removed, leaving as outgoing edges and nodes only nodes which are not covered under $\chi_f$. In addition, when a terminal node of one mc-graph is composed with a non-terminal node of the other, the resulting node is a terminal node in $G_\|$. Note that we compute $\chi_f$ and $E_f$ only by need. In fact, when constructing $G_\|$ it suffices to consider the indefinite subgraphs $G_1^?$ and $G_2^?$ of $G_1$ and $G_2$ resp. (pruned in definite nodes). This is because whenever a definite node is composed with another node (definite or not), $\chi_f$ of the resulting node is definite, which makes it a terminal node.

We accompany $G_\|$ with an initial mc-function, $\chi_I$, for its terminal nodes, based on the mc-functions of the two mc-graphs. We use the following observation:
Let $n = (s_1, s_2) \vdash \psi$ be a terminal node in $G_\|$. Then at least one of the following holds.

Either (a) at least one of $s_1 \vdash \psi$ and $s_2 \vdash \psi$ is a terminal node in its mc-graph; Or (b) $\chi_f(s_1 \vdash \psi, s_2 \vdash \psi)$ is definite; Or (c) both $s_1 \vdash \psi$ and $s_2 \vdash \psi$ are non-terminal but no outgoing edges were left in their composition.

**Definition 6.9.** *The initial mc-function $\chi_I$ of $G_\parallel$ is defined as follows. Let $n = (s_1, s_2) \vdash \psi \in N_\parallel$ be a terminal node. If it fulfills case (a) or (b), then $\chi_I(n) = \chi_1(s_1 \vdash \psi) \oplus \chi_2(s_2 \vdash \psi)$. If it fulfills case (c), then $\chi_I(n) = true$ if $n$ is a $\wedge$-node, and $\chi_I(n) = false$ if $n$ is a $\vee$-node. $\chi_I$ is undefined for the rest of the nodes.*

**Theorem 6.10.** *The resulting product graph $G_\parallel$ is a closed subgraph of $G$, the mc-graph over $M_1||M_2$ with a mc-function $\chi$. In addition, $\chi_I$ is defined over all the terminal nodes of $G_\parallel$, and is correct w.r.t. $\chi$.*

Theorems 5.4 and 6.10 imply that applying the **mc-algorithm** on $G_\parallel$ with $\chi_I$ results in a correct mc-function $\chi$ w.r.t. $G_\parallel$. Thus, $\chi(n^0_\parallel)$ is the value of the model checking of $\varphi$ in $M_1||M_2$. As a result, to model check $\varphi$ on $M_1||M_2$ it remains to model check $G_\parallel$. Note that the full graph for $M_1||M_2$ is not constructed.

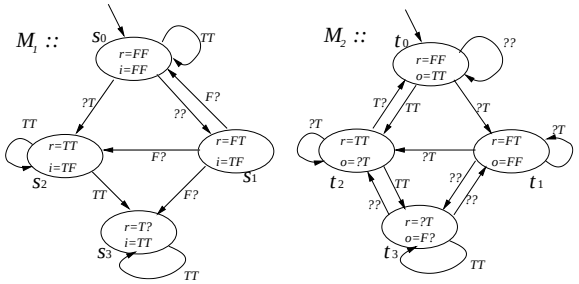


**Fig. 2.** Components $M_1$ and $M_2$

If the model checking result is indefinite (which is only possible if $M_1||M_2$ is abstract), then refinement is performed on each component separately, as described in the following steps, which summarize our compositional algorithm for checking an alternation-free $L_\mu$ formula $\varphi$ on $M_1||M_2$.

**Step 1:** Model check each $M_i\uparrow$ separately (for $i \in \{1, 2\}$):
1. Construct the mc-graph $G_i$ for $\varphi$ and $M_i\uparrow$.
2. Apply multi-valued model checking on $G_i$. Let $\chi_i$ be the resulting mc-function.

If $\chi_1(n^0_1)$ or $\chi_2(n^0_2)$ is definite, return the corresp. model checking result for $M_1||M_2$.

**Step 2:** Consider the composition $M_1||M_2$:
1. Construct the product graph $G_\parallel$ of the mc-graphs $G^?_1$ and $G^?_2$.
2. Apply multi-valued model checking on $G_\parallel$ (with the initial mc-function).

If $\chi_\parallel(n^0_\parallel)$ is definite, return the corresp. model checking result for $M_1||M_2$.

**Step 3:** Refine: Consider the failure node and reason returned by model checking $G_\parallel$ (where $\chi_\parallel(n^0_\parallel)$ is indefinite). If it is $p$ for some $p \in AP_i$, then refine $G^?_i$;
Else let it be a transition $((s_1, s_2), (s'_1, s'_2))$. Then:
1. If $s'_1$, $s'_2$ are weakly composable but not composable, refine both $G^?_1$ and $G^?_2$ according to $AP_1 \cap AP_2$.
2. If $R_i(s_i, s'_i) \leq_t R_{\bar{i}}(s_{\bar{i}}, s'_{\bar{i}})$, refine the transition $R_i(s_i, s'_i)$ in $G^?_i$.
3. If $R_i(s_i, s'_i)$ and $R_{\bar{i}}(s_{\bar{i}}, s'_{\bar{i}})$ are uncomparable, refine the subgraph(s) in which the transition is indefinite.

Go to Step 1(2) with the refined indefinite subgraphs.

**Theorem 6.11.** *For finite components, the compositional algorithm is guaranteed to terminate with a definite answer.*

*Example 6.12.* An example for the algorithm is given in Fig. 2,3. The two (abstract) components are described in Fig. 2. The underlying multi-valued structure is the $3\times3$ structure (described in Fig. 1(e), (f)). The atomic proposition $o$ is local to $M_1$, $i$ is local to $M_2$, and $r$ is a joint atomic proposition that $M_1$ and $M_2$ synchronize on. We wish to verify the property $\Box(\neg i \vee \Diamond o)$. The mc-graph of the lifted model $M_1\uparrow$ is described in Fig. 3(a). The mc-graph of $M_2\uparrow$ can be created similarly. Note that the edges of the mc-graph get a different value than their value in the model, as this is a mc-graph of the lifted model, thus we can no longer guarantee the existence of these edges. The model checking on each of the models does not result in a definite answer, and we need to consider their composition. The parts that are actually composed are marked with solid lines in Fig. 3(a). The product graph and its model checking is shown in Fig. 3(b), where the edges get their actual value. The nodes which are covered are marked with dashed lines. These nodes are created and removed on-the-fly, since they are covered. The actual nodes that compose the product graph are marked with solid lines. The property is still not verified in the composed model, thus a refinement is needed. Note that the product graph considers only a small part of the compound system, as it takes advantage of the information from model checking the separate components.

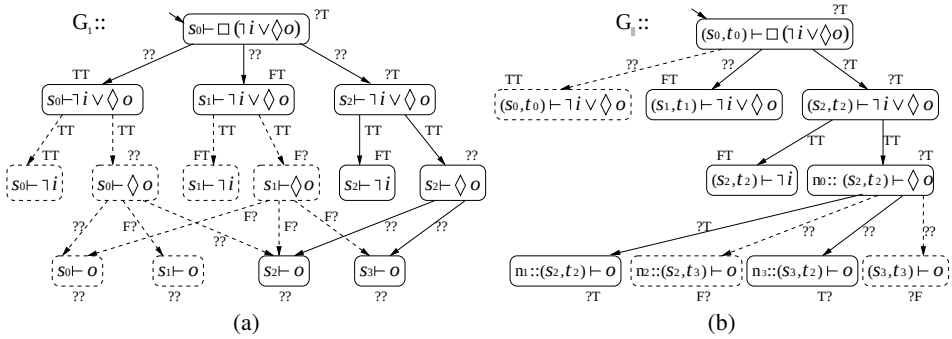


**Fig. 3.** (a) mc-graph of $M_1\uparrow$; (b) the product graph. Dashed nodes are covered. Solid lines mark actual product graph.

## 7 Discussion

This paper describes a *framework* for multi-valued model checking of $L_\mu$ formulas w.r.t. systems composed of several components. The framework is described as follows.

- Lift each individual component $M_i$ into a component $M_i\uparrow$ s.t. $M_1||M_2 \preceq M_i\uparrow$.
- Model check each of the lifted models separately. If the result is definite, then this also holds for the full system.
- Construct the product graph of the individual mc-graphs; model check it correctly.
- If the result on the product graph is definite, then this result holds for the full system. Otherwise, refine the components as needed.

We showed how our framework can be implemented for model checking of CPDBs, and alternation-free $L_\mu$ formulas. We applied a specific model checking and reason finding algorithm (Algo. 2.6, Sec. 4), but these can be replaced by other algorithms.

Our framework is suitable for full $L_\mu$, provided that the model checking and reason finding algorithm can handle the full $L_\mu$. Examples of such algorithms for a 3-valued structure can be found in [13]. Indeed, in [21] a compositional framework such as ours, but with the 3-valued semantics, has been presented for the full $L_\mu$.

Our framework can also be used for logics other than the $\mu$-calculus. For example, the *full-PML* logic, which extends the modal operators with past operators, $AY$ and $EY$, is used in [1], along with a 6-valued structure (described in Fig. 1(c),(d)). The structure is a CPDB, but since they use a logic with significantly different semantics, specific adaptations in some of the framework stages should be done.

## References


1. T. Ball, O. Kupferman, and G. Yorsh. Abstraction for falsification. In *CAV'05*.
2. N.D Belnap. A useful four-valued logic. In *Modern uses of multiple valued logics*. 1977.
3. G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *CAV'99*.
4. G. Bruns and P. Godefroid. Model checking with multi-valued logics. In *ICALP'04*.
5. G. Bruns and P. Godefroid. Temporal logic query checking. In *LICS'01*.
6. W. Chan. Temporal-logic queries. In *CAV*, volume 1855 of *LNCS*, pages 450–463, 2000.
7. M. Chechik, B. Devereux, S.M. Easterbrook, and A. Gurfinkel. Multi-valued symbolic model-checking. *ACM Transactions on Software Engineering and Methodology*, 12, 2003.
8. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT press, 1999.
9. D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst.*, 19(2), 1997.
10. S.M. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *ICSE'01*.
11. M. Fitting. Bilattices are nice things. In *Self-Reference*, 2002.
12. P. Godefroid and R. Jagadeesan. Automatic abstraction using generalized model checking. In *CAV'02*.
13. O. Grumberg, M. Lange, M. Leucker, and S. Shoham. When not losing is better than winning: Abstraction and refinement for the full $\mu$-calculus. *Information and Computation*, 205(8), 2007.
14. A. Gurfinkel and M. Chechik. Why waste a perfectly good abstraction? In *TACAS'06*.
15. A. Gurfinkel, O. Wei, and M. Chechik. Systematic construction of abstractions for model-checking. In *VMCAI'06*.
16. M. Huth and S. Pradhan. Lifting assertion and consistency checkers from single to multiple viewpoints. TR 2002/11, Dept. of Computing, Imperial College, London, 2002.
17. H. Jain, D. Kroening, N. Sharygina, and E.M. Clarke. Word level predicate abstraction and refinement for verifying RTL Verilog. In *DAC'05*.
18. O. Kupferman and Y. Lustig. Latticed simulation relations and games. In *ATVA'07*.
19. Y. Meller, O. Grumberg, and S. Shoham. A framework for compositional verification of multi-valued systems via abstraction-refinement. TR CS-2009-14, Dept. of Computer Science, Technion – Israel Institute of Technology, 2009.
20. C.-J. H. Seger and R. E. Bryant. Formal verification by symbolic evaluation of partially-ordered trajectories. *Formal Methods in System Design*, 6(2), 1995.
21. S. Shoham and O. Grumberg. Compositional verification and 3-valued abstractions join forces. In *SAS'07*.
22. S. Shoham and O. Grumberg. Multi-valued model checking games. In *ATVA'05*.
23. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J.Math*, 5, 1955.