Don't Know for Multi-valued Systems

Alarico Campetelli, Alexander Gruler, Martin Leucker, and Daniel Thoma

Institut für Informatik, Technische Universität München, Germany

Abstract. This paper studies abstraction and refinement techniques in the setting of multi-valued model checking for the μ-calculus. Two dimensions of abstractions are identified and studied: Abstraction by joining states of the underlying multi-valued Kripke structure as well as abstraction of truth values, for each following both an *optimistic* and *pessimistic* account. It is shown that our notion of abstraction is *conservative* in the following sense: The truth value in a concrete system is "between" the optimistic and pessimistic assessment. Moreover, model checking of abstracted systems is shown to be again a multi-valued model checking problem, allowing to reuse multi-valued model checking engines. Finally, whenever the optimistic and pessimistic model checking result differ, the *cause* for such an assessment is identified, allowing the abstraction to be refined to eventually yield a result for which both the optimistic and pessimistic assessment coincide.

1 Introduction

In multi-valued logics, a formula evaluates no longer to just $true$ or $false$ but to one of many *truth values.* This allows to express to which *extent* a property is considered satisfied by a program, or, in the setting of *trust models*, *how much* a person can be trusted [1]. The main motivation of this work is, however, inspired by our study of software product lines (or software product families). Within software product lines [2] a set of similar systems, called *products*, is modelled explicitly expressing their commonalities and differences. We have shown in [3] that a software product family can conveniently be modeled as one *single* multi-valued system, in which each value corresponds to a subset of products. Thus, the question of *which* products of the product line satisfy a certain property corresponds to the truth value of the formula encoding the property with respect to the multi-valued system.

As explained in [4,5], a Kripke structure (KS) can be extended to the multi-valued setting by assigning to each proposition in each state one of many (truth) *values* and likewise to each transition also a value, resulting in the notion of a multi-valued Kripke structure (mv-KS). A value of some proposition might then be interpreted as to which extend a proposition holds, a person may be trusted initially, or in which products of a product line a certain proposition holds. Similarly, the value of a transition might identify to which amount a transition might influence the truth value, might modify the trust value of some person when taking the transition, or, in which products the transition is actually present.

In model checking, a temporal or modal logic is typically used to specify (intended) properties of a given mv-KS. As such a logic typically ranges over atomic proposition,

Z. Liu and A.P. Ravn (Eds.): ATVA 2009, LNCS 5799, pp. 289–305, 2009.

Boolean combinations, and temporal or modal operators, it is helpful to consider values from a *lattice*, where *meet* ($\sqcap$) and *join* ($\sqcup$) naturally yield a semantics for conjunction and disjunction, respectively. The meaning of temporal or modal operators is adjusted appropriately. Then, the semantics of a formula with respect to a mv-KS is one element of the lattice, denoting either the extent to which the formula holds, the trust value of some person performing actions, or the set of all products satisfying the formula. Model Checking multi-valued versions of the classical logics LTL, CTL, CTL*, and the μ-calculus has the been extensively studied already, for example in [6,4,5,7].

The state explosion problem in (two-valued) model checking—complicating its practical application—does, of course, not vanish in the multi-valued setting. Therefore, it is important to study abstraction techniques also for multi-valued model checking. This paper studies abstraction and refinement techniques in the setting of multi-valued model checking for the μ-calculus as introduced in [5].

We identify and study two orthogonal forms of abstractions for mv-KS: Firstly, we consider abstractions of mv-KS induced by joining states to form abstract states, as it is typically considered also in the two-valued setting. In this setting, a meanwhile popular form of abstraction is to consider structures that are simultaneously an *over* as well as *under*-approximation of a system, as introduced in [8] in the setting of the μ-calculus. To this end, Larsen's and Thomsen's Kripke modal transition systems are used to describe abstract systems in which a transition can be *may*, denoting an over-approximation, or *must*, denoting an under-approximation [9]. This, essentially leads to *three* possible values for a transition: It is there for sure, it is not there for sure, or it may be there. This explains that three-valued settings are themselves often used for abstractions [10]. In this paper, however, we study abstraction *of* rather than abstraction *by* mv-KS. Over-approximation can be intuitively explained as an *optimistic* account of the system's transitions, while we consider an under-approximation a *pessimistic* account. We then transfer this understanding to the multi-valued setting to obtain notions of abstractions. A first, interesting result is that an abstract mv-KS can be again considered as a mv-KS, yet over a richer lattice, which we call the op-lattice. This allows to reuse multi-valued model checking engines also for abstraction.

The second source of abstraction that we study is that by abstracting values. Especially in the setting of product lines, in which a family of N products gives rise to the powerset lattice with 2^N elements, it is essential to also abstract lattice elements, practically say by identifying different products, hereby reducing N. To follow both the optimistic and pessimistic view, we need, however, *two* abstractions for the lattice elements, here given as usual in abstract interpretation by (two) Galois connections [11]. We introduce a simulation relation and show that in the presence of such a relation the actual model checking values lies "between" the optimistic and pessimistic one that is based on the abstract system. Note that in [12] the concept of a *latticed simulation* is introduced, which, however, does not allow to combine a pessimistic and optimistic view into one abstraction.

Finally, whenever the optimistic and pessimistic model checking result differ, the *cause* for such an assessment is identified. We explain how to compute the cause based on the structure of the formula to be checked. Knowing causes allows to refine the

abstraction to eventually yield a result for which both the optimistic and pessimistic assessment coincide.

2 Preliminaries

Lattices. An algebraic structure $(\mathcal{L}, \sqcap, \sqcup)$ consisting of a set $\mathcal{L}$, a binary operation $\sqcap : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ called *meet* and a binary operation $\sqcup : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ called *join* is a *lattice* if it satisfies the following equations for all elements $x, y, z \in \mathcal{L}$: (i) $x \sqcap y = y \sqcap x$ and $x \sqcup y = y \sqcup x$, (ii) $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$ and $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$, (iii) $x \sqcap (y \sqcup x) = x$ and $x \sqcup (y \sqcap x) = x$, and (iv) $x \sqcap x = x$ and $x \sqcup x = x$. Equivalently to the definition as an algebraic structure, a lattice can be defined as a partially ordered set $(\mathcal{L}, \sqsubseteq)$ where for each $x, y \in \mathcal{L}$, there exists (i) a unique *greatest lower bound* (glb), which is called the *meet* of x and y, denoted by $x \sqcap y$, (ii) and a unique *least upper bound* (lub), which is called the *join* of x and y, denoted by $x \sqcup y$. Depending on the application, in the following we use one or the other form for dealing with lattices.

The definitions of glb and lub extend to finite sets of elements $A \subseteq \mathcal{L}$ as expected, which are then denoted by $\bigsqcap A$ and $\bigsqcup A$, respectively. A lattice is called *finite* iff $\mathcal{L}$ is finite. Every finite lattice has a least element, called *bottom*, denoted by $\bot$, and a greatest element, called *top*, denoted by $\top$. A lattice is *distributive*, iff $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$, and, dually, $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$. In a *de Morgan* lattice, every element x has a unique *dual* element $\neg x$, such that $\neg\neg x = x$ and $x \sqsubseteq y$ implies $\neg y \sqsubseteq \neg x$. Typically, we denote a de Morgan lattice as a quadruple $(\mathcal{L}, \sqcap, \sqcup, \neg)$.

A complete distributive lattice is called *Boolean* iff for all elements $x \in \mathcal{L}$ we have $x \sqcup \neg x = \top$ and $x \sqcap \neg x = \bot$. A typical Boolean lattice is the one induced by the power of some non-empty finite set $\{1, \ldots, N\}$, for $N \in \mathbb{N}$: $(2^N, \subseteq)$ where meet, join, and dual of elements, are given by set intersection, set union, and complement of sets, respectively. For example, Figure 4(a) shows the powerset lattice for $N = 3$.

Multi-valued Kripke Structures. Let $\mathcal{P}$ be a set of propositional constants. A *multi-valued Kripke structure* (mv-KS) is a tuple $\mathcal{K} = (S, \mathcal{L}, \mathcal{R}, L)$ where S is a set of states, $\mathcal{L}$ is a de Morgan lattice, $\mathcal{R} : S \times S \to \mathcal{L}$ is a transition function associating an element of the lattice to each pair of states, and $L : S \to (\mathcal{P} \to \mathcal{L})$ is yields for a proposition in each state an element of the lattice. With $\mathcal{MK}$ we denote the set of all mv-KS.

A Kripke structure in the usual sense can be seen as a mv-KS with values over the two element lattice consisting of a bottom $\bot$ and a top $\top$ element, ordered as shown in Figure 3(a). Value $\top$ then means that the property holds in the considered state while $\bot$ means that it does not hold. Similarly, $\mathcal{R}(s, s') = \top$ reads as there is a corresponding transition while $\mathcal{R}(s, s') = \bot$ means there is no transition between the states s and s'.

Multi-valued modal μ-calculus. In the following we introduce a *multi-valued modal version* of the μ-calculus along the lines of [7]. Let $\mathcal{V}$ be a set of propositional variables. Formulae of the *multi-valued modal μ-calculus* are given by

$$\varphi ::= \mathit{true} \mid \mathit{false} \mid q \mid \neg\varphi \mid Z \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mu Z.\varphi \mid \nu Z.\varphi$$

where $q \in \mathcal{P}$, and $Z \in \mathcal{V}$. Let $mv\text{-}\mathfrak{L}_\mu$ denote the set of *closed* formulae generated by the above grammar, where the fixpoint quantifiers μ and ν are variable binders. The

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg\varphi]\!]_\rho & := & \lambda s.\neg[\![\varphi]\!]_\rho(s) \\
[\![Z]\!]_\rho & := & \rho(Z)
\end{array}
\qquad
\begin{array}{l}
[\![\varphi \vee \psi]\!]_\rho := [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho := [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho := \lambda s. \bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho := \lambda s. \bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho := \bigsqcap\{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho := \bigsqcup\{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Fig. 1. Semantics of mv-$\mathfrak{L}_\mu$ formulae

semantics of a mv-$\mathfrak{L}_\mu$ formula is an element of $\mathcal{L}^S$, i.e., a function from S to $\mathcal{L}$ yielding for the formula at hand and a given state its *satisfaction value* measured in terms of the lattice. In the setting of software product lines, for example, this is the set of all products for which the formula holds in the given state. More precisely, the *semantics* $[\![\varphi]\!]^{\mathcal{K}}_\rho$ of a mv-$\mathfrak{L}_\mu$ formula φ with respect to a mv-KS $\mathcal{K}$ and an *environment* $\rho : \mathcal{V} \to \mathcal{L}^S$, which explains the meaning of free variables in φ, is defined as shown in Figure 1, where $\rho[Z \mapsto f]$ denotes the environment that maps Z to f and agrees with ρ on all other arguments. When clear from the context, we drop indices $\mathcal{K}$ and ρ.

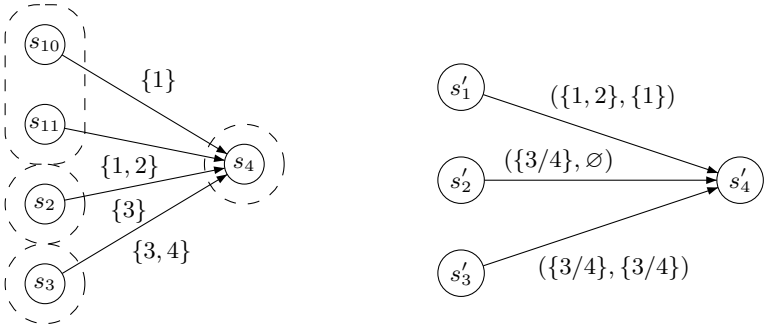
Regarding motivation, consider the $\Diamond$-operator as an example: $\Diamond\varphi$ is classically supposed to hold in states that have a successor satisfying φ. In a multi-valued version, we first take the value of a transition and reduce it (meet it) with the value of φ in that successor. As there might be transitions to different successors, we take the best value.

The functionals $\lambda f.[\![\varphi]\!]_{\rho[Z\mapsto f]} : \mathcal{L}^S \to \mathcal{L}^S$ are monotone wrt. $\sqsubseteq$ for any Z, φ and S. By [13], least and greatest fixpoints of these functionals exist. *Approximants* of mv-$\mathfrak{L}_\mu$ formulae are defined as usual: if $fp(Z) = \mu Z.\varphi$ then $Z^0 := \lambda s.\bot$, $Z^{n+1} := [\![\varphi]\!]_{\rho[Z\mapsto Z^n]}$ for any ordinal n and any environment ρ, and $Z^\lambda := \bigsqcap_{n<\lambda} Z^n$ for a limit ordinal λ. Dually, if $fp(Z) = \nu Z.\varphi$ then $Z^0 := \lambda s.\top$, $Z^{n+1} := [\![\varphi]\!]_{\rho[Z\mapsto Z^n]}$, and $Z^\lambda := \bigsqcup_{n<\lambda} Z^n$.

3 Conservative Abstractions for MV-Kripke Structures

In this section we introduce two different kinds of abstractions for multi-valued Kripke structures, both motivated by practical applications, and made explicit by respective abstraction operators. As we show, the combined application of both abstraction operations (i) yields again a mv-KS, which means that we can technically deal with the abstracted system in the same way as with the concrete system, i. e. in particular, we can apply the same model checking techniques, and (ii) that it is conservative in the sense that the result of evaluating any mv-$\mathfrak{L}_\mu$ formulae on the abstract system is always a conservative abstraction of the evaluation on the concrete system.

Abstraction by joining states. Consider the part of a mv-KS shown in Figure 2(a). A standard idea of abstraction, which we follow as our first approach, is to join states in a concrete system to form combined *abstract states* in the abstract system. For example,



(a) Concrete system. (b) Abstract system (mv-KS).

Fig. 2. An Example of a combined abstraction. The abstraction joined states as indicated by the dashed lines, e. g. s_{10} and s_{11}, and does additionally abstract the original lattice elements 3 and 4 with one abstract element denoted by $3/4$.

the concrete states s_{10} and s_{11} in Figure 2(a) are joined (indicated by the dashed line) to form the abstract state s'_1 in Figure 2(b).[1]

Regarding transitions, we label the abstract transitions with values representing over- and under-approximation of the corresponding concrete (original) transitions. More specifically, the two transitions from the concrete states s_{10} and s_{11} to s_4, respectively, are over-approximated by the label $\{1,2\}$ and under-approximated with $\{1\}$, in the abstraction. We combine this approximation information to label the corresponding transition in the abstract system with a *tuple* consisting of the over- and under-approximation. This results again in a lattice $\mathcal{L}_{op}$ of tuples based on the lattice $\mathcal{L}$ used to label transitions in the concrete system as follows:

Definition 1 (op-lattice). *Let $\mathcal{L}$ be a de Morgan lattice. The lattice*

$$\mathcal{L}_{op} = (\{(m_1, m_2) \in \mathcal{L} \times \mathcal{L} \mid m_1 \sqsupseteq m_2\}, \sqcap_{op}, \sqcup_{op}, \neg_{op})$$

with the operations $\sqcap_{op}$, $\sqcup_{op}$, $\neg_{op}$ given by

$$\begin{aligned}(m_1, m_2) \sqcap_{op} (m'_1, m'_2) &:= (m_1 \sqcap m'_1, m_2 \sqcap m'_2)\\ (m_1, m_2) \sqcup_{op} (m'_1, m'_2) &:= (m_1 \sqcup m'_1, m_2 \sqcup m'_2)\\ \neg_{op}(m_1, m_2) &:= (\neg m_2, \neg m_1)\end{aligned}$$

is called the optimistic-pessimistic lattice (op-lattice) *for $\mathcal{L}$.*

We called $\mathcal{L}_{op}$ an *optimistic-pessimistic* lattice as its elements embody these two kinds of views—regardless whether we interpret its elements as degrees of truth or configurations of a software product line: The first entry of the tuple represents the best case, e. g. the "highest" truth value or the largest set of configurations of a software product family, while the second entry represents the (worst) case which is achieved at the least, e. g.

[1] Please ignore the transitions from s'_2 and s'_3 to s'_4 for the moment.

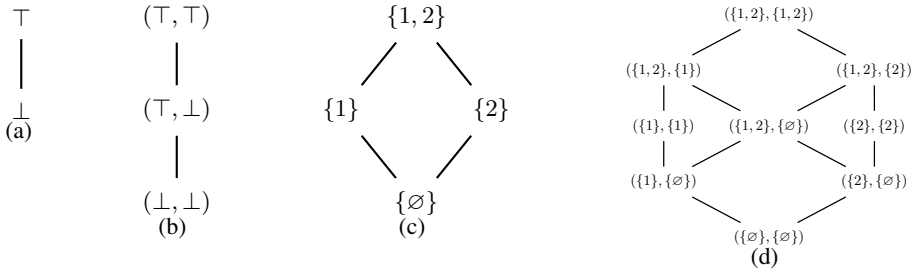


Fig. 3. In (b) true, false and don't know lattice for the two-element lattice (a), in (c) lattice for two product lines and corresponding op-lattice in (d)

the "lowest" guaranteed truth value or the (smallest) set of configurations which still guarantees a certain property. The op-lattice of the two valued and four valued Boolean lattices shown in Figures 3(a) and 3(c) are given in Figures 3(b) and 3(d), respectively. Observe that Figure 3(b) shows the three-valued lattices commonly used for abstraction in the setting of the μ-calculus [14].

Let us formalize the abstraction process motivated before. To this end, we define what it means to combine concrete states and abstract them to an abstract state in an abstract system. Let the function $\gamma : S_A \to \mathcal{P}(S_C)$ be a mapping which identifies (i) which concrete states in S_C are combined, (ii) and to which abstract state in S_A they are abstracted. We require γ to be *abstraction complete* , i.e., for all $s_C \in S_C$ there is $s_A \in S_A$ with $s_C \in \gamma(s_A)$. Thus each concrete state is accounted by at least one abstract state. Now, we are prepared to define an abstraction operator abs_S which represents the act of abstracting by joining states.

Definition 2 (State Abstraction Operator). *We call the function abs_S yielding the abstract mv-KS $abs_S((S_C, \mathcal{L}_C, \mathcal{R}_C, L_C), \gamma) = (S_A, \mathcal{L}_A, \mathcal{R}_A, L_A)$ of the concrete mv-KS $(S_C, \mathcal{L}_C, \mathcal{R}_C, L_C)$ by joining states according to the abstraction complete function γ the* state abstraction operator, *where the set S_A of abstract states is implicitly given by γ, the lattice $\mathcal{L}_A$ is the op-lattice of $\mathcal{L}_C$ and*

$$\mathcal{R}_A(s_A, s'_A) = \left(\bigsqcup_{s_C \in \gamma(s_A)} \bigsqcup_{s'_C \in \gamma(s'_A)} \mathcal{R}_C(s_C, s'_C) \, , \; \bigsqcap_{s_C \in \gamma(s_A)} \bigsqcup_{s'_C \in \gamma(s'_A)} \mathcal{R}_C(s_C, s'_C) \right)$$

$$L_A(s_A, p) = \left(\bigsqcup_{s_C \in \gamma(s_A)} L_C(s_C, p) \, , \; \bigsqcap_{s_C \in \gamma(s_A)} L(s_C, p) \right)$$

The function abs_S maps a mv-KS representing a concrete system to a mv-KS which representing its abstraction by combining states via function γ. For the pessimistic view, it takes the worst of the (best) combinations of states. For example, the lattice shown in Figure 3(d) is the op-lattice which defines the transition labels for an abstract mv-KS which is constructed by abstracting from a concrete mv-KS where the transitions were

labeled with elements of the lattice shown in Figure 3(c). Also note, that following our construction we exactly get the usual three-valued lattice (cf. Figure 3(b)) if we abstract a (two-valued) Kripke structure, where the existence and missing of transitions is equal to labeling the transitions with the two-valued *Boolean lattice* as shown in Figure 3(a) [14].

Abstraction of lattices elements. The abstraction of states as performed by our first abstraction operation abs_S reduces the state space by joining states. Thus, the abstract system usually also has significantly fewer transitions. To further reduce the abstract system, we may want to identify some elements of the lattice. Therefore, we also consider abstractions of the original lattice. In the following we introduce such a kind of abstraction and provide a definition of a second abstraction operator, correspondingly. Note, that the first abstraction operator yields a mv-KS labeled with an op-lattice, on which we can apply the second abstraction operator subsequently.

Consider again Figure 2(b): Assume that we no longer want to differ between the two lattice elements 3 and 4. Thus, we abstract these elements to a single element—denoted by $3/4$—in the abstract lattice. Being the only abstraction of lattice elements we make, the resulting abstract lattice consists of the three elements $\{1, 2, 3/4\}$, which may now be used to label transitions in the abstract system. The effect of this abstraction is shown in Figure 2(b): Since we abstract 3 and 4 together, whenever 3 and 4 occur apart from each other in the concrete system, this asks for an optimistic and a pessimistic view in the abstract system, represented by the transition $s'_2 \xrightarrow{(3/4,\emptyset)} s'_4$: There is no transition in the pessimistic case (entry $\emptyset$ of the tuple), yet a transition for the element $3/4$ in the optimistic view.

In order to formalize such an abstraction from lattices we use the concept of Galois connections which is well know in the area of abstract interpretation.

Definition 3 (Galois Connection [11]). *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be lattices. A pair $(\uparrow, \downarrow)$ of monotone functions $\uparrow : \mathcal{L}_1 \to \mathcal{L}_2$ and $\downarrow : \mathcal{L}_2 \to \mathcal{L}_1$ is a* Galois connection *from $\mathcal{L}_1$ to $\mathcal{L}_2$, if $\forall l \in \mathcal{L}_1 : l \sqsubseteq \downarrow(\uparrow(l))$ and $\forall a \in \mathcal{L}_2 : \uparrow(\downarrow(a)) \sqsubseteq a$.*

For the soundness proof of our approach we depend on some of the common properties of Galois connections. In particular, any Galois connection $(\uparrow, \downarrow)$ from $\mathcal{L}_1$ to $\mathcal{L}_2$ fulfills (i) $\uparrow(l) \sqsubseteq m \Leftrightarrow l \sqsubseteq \downarrow(m)$, (ii) $\uparrow(\bigsqcup L) = \bigsqcup_{l \in L} \uparrow(l)$, and (iii) $\downarrow(\bigsqcap M) = \bigsqcap_{m \in M} \downarrow(m)$. Figure 4 shows an example of a Galois connection between two lattices which illustrates these properties. The abstraction combines the (concrete) elements 1 and 2 of the lattice $\mathcal{L}_1$ (cf. Figure 4(a)) to the single element $1/2$ in the lattice $\mathcal{L}_2$. The solid line shows an example of how the Galois connection $(\uparrow, \downarrow)$ works: We abstract $(\uparrow)$ element $\{2\}$ to $\{1/2\}$ and concretize $(\downarrow)$ then back to the $\{1, 2\}$. This is an optimistic approximation as $\{2\} \sqsubseteq \{1, 2\}$. In particular, properties (i)-(iii) are fulfilled.

Usually Galois connections are applied for abstractions by using $\uparrow$ as the abstraction function and $\downarrow$ as the concretization function. Since by definition $l \sqsubseteq \downarrow(\uparrow(l))$ holds, this abstraction yields an over-approximation, or an optimistic approximation in our terminology. For our approach we additionally need a pessimistic approximation (under-approximation). We define the pessimistic approximation using a second Galois connection and swapping the interpretation of the mapping functions: For the pessimistic case we use $\downarrow$ as an abstraction function and $\uparrow$ as a concretization function.

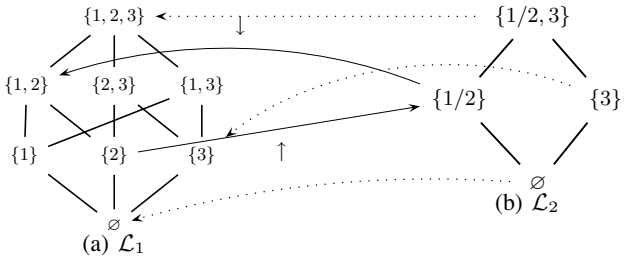


Fig. 4. A galois connection allows to interpret the lattice $\mathcal{L}_2$ as an abstraction of $\mathcal{L}_1$, where the (concrete) lattice elements 1 and 2 are abstracted to a single element in $\mathcal{L}_2$, denoted by $1/2$. Dashed and solid lines illustrate a part of the galois connection $(\uparrow, \downarrow)$.

We now define the lattice to label our abstract systems. We call it the *abstract op-lattice.* It is based on the original lattice from which we abstract elements and two Galois connections which define our optimistic and pessimistic view. Since we now use different lattices for the optimistic and pessimistic approximation, it is no longer possible to define negation for the elements of the abstract lattice directly. Therefore we additionally require two negation functions, that map between the optimistic and pessimistic view appropriately.

Definition 4 (aop-lattice). *Let $\mathcal{L}_C$ be a de Morgan lattice and $\mathcal{L}_\mathsf{o}$ and $\mathcal{L}_\mathsf{p}$ be lattices. Let $(\uparrow_\mathsf{o}, \downarrow_\mathsf{o})$ with $\uparrow_\mathsf{o} : \mathcal{L}_C \to \mathcal{L}_\mathsf{o}$ and $\downarrow_\mathsf{o} : \mathcal{L}_\mathsf{o} \to \mathcal{L}_C$ and $(\uparrow_\mathsf{p}, \downarrow_\mathsf{p})$ with $\uparrow_\mathsf{p} : \mathcal{L}_\mathsf{p} \to \mathcal{L}_C$ and $\downarrow_\mathsf{p} : \mathcal{L}_C \to \mathcal{L}_\mathsf{p}$ be Galois connections. Furthermore, let $\mathcal{L}_\mathsf{o}$ and $\mathcal{L}_\mathsf{p}$ be connected by two anti-monotone negation functions $\neg_\mathsf{o} : \mathcal{L}_\mathsf{o} \to \mathcal{L}_\mathsf{p}$ and $\neg_\mathsf{p} : \mathcal{L}_\mathsf{p} \to \mathcal{L}_\mathsf{o}$ with $\neg_\mathsf{o}\uparrow_\mathsf{o}(x) \sqsubseteq \downarrow_\mathsf{p}(\neg x)$ and $\uparrow_\mathsf{o}(\neg x) \sqsubseteq \neg_\mathsf{p}\downarrow_\mathsf{p}(x)$. We call the lattice*

$$\mathcal{L}_{aop} = \big(\{(m_\mathsf{o}, m_\mathsf{p}) \in \mathcal{L}_\mathsf{o} \times \mathcal{L}_\mathsf{p} \mid \downarrow_\mathsf{o}(m_\mathsf{o}) \sqsupseteq \uparrow_\mathsf{p}(m_\mathsf{p})\},\ \sqcap_{aop},\ \sqcup_{aop},\ \neg_{aop}\big)$$

with the operations given by

$$\begin{aligned}(m_\mathsf{o}, m_\mathsf{p}) \sqcap_{aop} (m'_\mathsf{o}, m'_\mathsf{p}) &:= (m_\mathsf{o} \sqcap m'_\mathsf{o}\ ,\ m_\mathsf{p} \sqcap m'_\mathsf{p})\\ (m_\mathsf{o}, m_\mathsf{p}) \sqcup_{aop} (m'_\mathsf{o}, m'_\mathsf{p}) &:= (m_\mathsf{o} \sqcup m'_\mathsf{o}\ ,\ m_\mathsf{p} \sqcup m'_\mathsf{p})\\ \neg_{aop}(m_\mathsf{o}, m_\mathsf{p}) &:= (\neg_\mathsf{p} m_\mathsf{p}\ ,\ \neg_\mathsf{o} m_\mathsf{o})\end{aligned}$$

the abstract optimistic-pessimistic lattice (aop-lattice) *for the lattice $\mathcal{L}_C$.*

Using the properties of Galois connections and the definition of the negation functions, we easily see:

Proposition 1 (aop-lattice is well-defined). *For all lattice values $x \in \mathcal{L}_C$ it holds that $(\uparrow_\mathsf{o}(x), \downarrow_\mathsf{p}(x)) \in \mathcal{L}_{aop}$ and that $\sqcup_{aop}$, $\sqcap_{aop}$, $\neg_{aop}$ are well-defined and in the sense that they preserve the condition $\downarrow_\mathsf{o}(m_\mathsf{o}) \sqsupseteq \uparrow_\mathsf{p}(m_\mathsf{p})$.*

Negation on an aop-lattice always yields an over- and under-approximation of a corresponding concrete element. But it also allows for the loss of information, since the

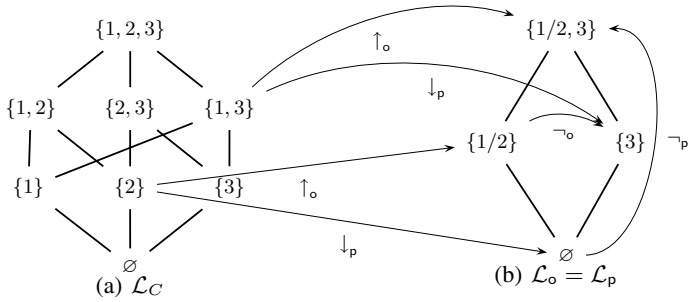


Fig. 5. Illustration of the negation function(s) in a aop-lattice

result of negation in the concrete might not be representable "exactly" in the abstract. Figure 5 illustrates such a situation. It shows the lattice $\mathcal{L}_C$ that we consider for the concrete system, whereas for the abstraction, the same lattices $\mathcal{L}_\mathsf{o} = \mathcal{L}_\mathsf{p}$ for the optimistic and pessimistic case are used. Consider the concrete element $x = \{2\}$ and its negation $\neg x = \{1, 3\}$ (in the concrete system). Following the arrows shows $\neg_\mathsf{o}\uparrow_\mathsf{o}(x) = \{3\} \sqsubseteq \{3\} = \downarrow_\mathsf{p}(\neg x)$ and $\uparrow_\mathsf{o}(\neg x) = \emptyset \sqsubseteq \{1/2, 3\} = \neg_\mathsf{p}\downarrow_\mathsf{p}(x)$. This means that the negation of the optimistic-pessimistic tuple encloses the actually correct negation result.

An aop-lattice is obviously distributive, but the preceding discussion shows that it is not a de Morgan lattice. However, our whole theory and model checking machinery can be extended to non-de Morgan lattices. In particular, Theorem 1 also holds for non de Morgan lattices. Nevertheless, using the same lattice for the over and underapproximation (cf. Theorem 2) yields an aop-lattice which *is* a de Morgan lattice. Thus, to simplify presentation, we silently assume that an aop-lattice is indeed a de Morgan lattice. Note, that if using the state abstraction operator only, we get an op-lattice which also *is* a de Morgan lattice.

Now we make precise the idea of abstraction by abstracting the lattice and provide a suitable abstraction operator abs_L.

Definition 5 (Lattice Abstraction Operator). *Let* $(S_A, \mathcal{L}_A, \mathcal{R}_A, L_A)$ *be a mv-KS, and* $\uparrow_\mathsf{o}, \downarrow_\mathsf{p}$ *be two Galois connections with corresponding negation functions* $\neg_\mathsf{o}, \neg_\mathsf{p}$*. Then, the* lattice abstraction operator abs_L *yields an abstracted mv-KS*

$$abs_L\left((S_A, \mathcal{L}_A, \mathcal{R}_A, L_A), \uparrow_\mathsf{o}, \downarrow_\mathsf{p}, \neg_\mathsf{o}, \neg_\mathsf{p}\right) = (S'_A, \mathcal{L}'_A, \mathcal{R}'_A, L'_A)$$

labeled with an aop-lattice $\mathcal{L}'_A$*, where* $S'_A = S_A$ *and*

$$\mathcal{R}'_A(s, s') = \left(\uparrow_\mathsf{o}((\mathcal{R}_A(s, s'))_1)\ ,\ \downarrow_\mathsf{p}((\mathcal{R}_A(s, s'))_2)\right)$$
$$L'_A(s, p) = \left(\uparrow_\mathsf{o}((L_A(s, p))_1)\ ,\ \downarrow_\mathsf{p}((L_A(s, p))_2)\right)$$

This completes our idea of abstraction: For any concrete mv-KS, the subsequent application of both abstraction operators—abs_L after abs_S—yields a mv-KS labeled with an aop-lattice representing the combination of both kinds of abstractions.

While we have provided two abstraction operators to exemplify our ideas of abstraction, we show that model checking the abstract system yields conservative results for the concrete system by means of a *conservative abstraction*.

Definition 6 (Conservative Abstraction). *Let $\mathcal{L}_C$ be a de Morgan lattice and $\mathcal{L}_A$ an aop-lattice with the Galois connections $(\uparrow_{\mathsf{o}}, \downarrow_{\mathsf{o}})$ from $\mathcal{L}_C$ to $\mathcal{L}_{\mathsf{o}}$ and $(\uparrow_{\mathsf{p}}, \downarrow_{\mathsf{p}})$ from $\mathcal{L}_{\mathsf{p}}$ to $\mathcal{L}_C$ with the negation functions $\neg_{\mathsf{o}}$ and $\neg_{\mathsf{p}}$. Let $\mathcal{K}_C = (S_C, \mathcal{L}_C, \mathcal{R}_C, L_C)$ be the concrete and $\mathcal{K}_A = (S_A, \mathcal{L}_A, \mathcal{R}_A, L_A)$ the abstract multi-valued Kripke-structure. Furthermore, let γ be an abstraction complete function which specifies how to abstract concrete states. Then, we call $\mathcal{K}_A$ a* conservative abstraction *of $\mathcal{K}_C$, if the following conditions hold:*

$$\uparrow_{\mathsf{p}}((L_A(s_A,p))_2) \sqsubseteq \bigsqcap_{s_C \in \gamma(s_A)} L(s_C,p) \qquad \text{(ca-lab (i))}$$

$$\downarrow_{\mathsf{o}}((L_A(s_A,p))_1) \sqsupseteq \bigsqcup_{s_C \in \gamma(s_A)} L(s_C,p) \qquad \text{(ca-lab (ii))}$$

$$\uparrow_{\mathsf{p}}((\mathcal{R}_A(s_A,s'_A))_2) \sqsubseteq \bigsqcap_{s_C \in \gamma(s_A)} \bigsqcup_{s'_C \in \gamma(s'_A)} \mathcal{R}_C(s_C,s'_C) \qquad \text{(ca-trans (i))}$$

$$\downarrow_{\mathsf{o}}((\mathcal{R}_A(s_A,s'_A))_1) \sqsupseteq \bigsqcup_{s_C \in \gamma(s_A)} \bigsqcup_{s'_C \in \gamma(s'_A)} \mathcal{R}_C(s_C,s'_C) \qquad \text{(ca-trans (ii))}$$

Note that both abstraction operators as well as their concatenation induce conservative abstractions. A conservative abstraction is exactly that kind of abstract mv-KS which we require such that the evaluation of a formula $\varphi \in mv\text{-}\mathfrak{L}_\mu$ (cf. Section 2) yields useful results. More precisely, evaluating a $mv\text{-}\mathfrak{L}_\mu$ formula on a conservative abstraction of a concrete system $\mathcal{K}_C$ always yields a tuple representing the optimistic and pessimistic approximation of the result that would be produced when evaluating the formula on $\mathcal{K}_C$ directly. Theorem 1 states this correctness result in a formal manner.

Theorem 1 (Correctness of abstraction). *Let $\mathcal{K}_C = (S_C, \mathcal{L}_C, \mathcal{R}_C, L_C)$ be the concrete multi-valued Kripke-structure and $\mathcal{K}_A = (S_A, \mathcal{L}_A, \mathcal{R}_A, L_A)$ be a conservative abstraction of $\mathcal{K}_A$. Let the corresponding Galois connections $(\uparrow_{\mathsf{o}}, \downarrow_{\mathsf{o}})$, $(\uparrow_{\mathsf{p}}, \downarrow_{\mathsf{p}})$, and the abstraction function γ be defined as in Definition 6. Then for all $s_A \in S_A$, for all $s_C \in \gamma(s_A)$ and for all formulae $\varphi \in mv\text{-}\mathfrak{L}_\mu$ it holds that:*

$$\uparrow_{\mathsf{p}}(m_p) \;\sqsubseteq\; [\![\varphi]\!]^{\mathcal{K}_C}_{\varnothing}(s_C) \;\sqsubseteq\; \downarrow_{\mathsf{o}}(m_o)$$

where $(m_o, m_p) = [\![\varphi]\!]^{\mathcal{K}_A}_{\varnothing}(s_A)$ is the result of the evaluation of φ on $\mathcal{K}_A$.

Proof. The proof is carried out by induction over the structure of a μ-calculus formula. To demonstrate the central ideas, we explain the induction step for the $\Diamond$-operator. We confine ourselves to the correctness of the under-approximation as the over-approximation can be proved in an similar, slightly simpler way.

Let $(\_)_2$ denote the second, pessimistic entry of an aop-lattice tuple. We want to prove, that $\uparrow_{\mathsf{p}}([\![\Diamond\varphi]\!](s_A))_2)$ yields an under-approximation for each concrete state $s_C \in \gamma(s_A)$ of the evaluation of the same formula on the concrete system. By semantics of the $\Diamond$-operator we obtain $\uparrow_{\mathsf{p}}(\bigsqcup_{s'_A}\{(\mathcal{R}(s_A,s'_A))_2 \sqcap ([\![\varphi]\!](s'_A))_2\})$. By induction we know that $([\![\varphi]\!](s'_A))_2 \;\sqsubseteq \downarrow_{\mathsf{p}}(\bigsqcap_{\hat{s}'_C \in \gamma(s'_A)} [\![\varphi]\!](\hat{s}'_C))$. This yields an upper bound for our under-approximation:

$$\uparrow_{\mathsf{p}}(\bigsqcup_{s'_A}\{\downarrow_{\mathsf{p}}(\bigsqcap_{\tilde{s}_C\in\gamma(s_A)}\bigsqcup_{\tilde{s}'_C\in\gamma(s'_A)}\mathcal{R}(\tilde{s}_C,\tilde{s}'_C))\sqcap\downarrow_{\mathsf{p}}(\bigsqcap_{\hat{s}'_C\in\gamma(s'_A)}[\![\varphi]\!](\hat{s}'_C))\})$$

Now we can apply Galois connection properties and choose $\tilde{s}_C = s_C$ and thus obtain a weaker upper bound $\bigsqcup_{s'_A}\{\bigsqcup_{\tilde{s}'_C\in\gamma(s'_A)}\mathcal{R}(s_C,\tilde{s}'_C)\sqcap\bigsqcap_{\hat{s}'_C\in\gamma(s'_A)}[\![\varphi]\!](\hat{s}'_C)\}$. By exploiting distributivity of $\sqcap$ and $\sqcup$ this can be simplified to:

$$\bigsqcup_{s'_A}\bigsqcup_{\tilde{s}'_C\in\gamma(s'_A)}\bigsqcap_{\hat{s}'_C\in\gamma(s'_A)}\{\mathcal{R}(s_C,\tilde{s}'_C)\sqcap[\![\varphi]\!](\hat{s}'_C)\}$$

We can now choose $\hat{s}'_C = \tilde{s}'_C$ and obtain once again a weaker upper bound:

$$\bigsqcup_{s'_C}\{\mathcal{R}(s_C,s'_C)\sqcap[\![\varphi]\!](s'_C)\} = [\![\Diamond\varphi]\!](s_C)$$

This least upper bound is identical to the definition of the $\Diamond$-operator semantics and thus completes the proof. □

So far, we have presented the most general setting, in which we used different lattices for the optimistic and pessimistic approximation, respectively. However, it may usually be more convenient to use the same (de Morgan) lattice for both kinds of approximation. Doing so allows to define one of the two required Galois connections in terms of the other one and to use the negation defined on the lattice instead of the two negation functions mapping between separated lattices:

Theorem 2 (Simplification). *Let $\mathcal{L}_C$ be a lattice and $\mathcal{L}_{\mathsf{op}}$ be a de Morgan lattice. Let $(\uparrow_{\mathsf{o}},\downarrow_{\mathsf{o}})$ be a Galois connection from $\mathcal{L}_C$ to $\mathcal{L}_{\mathsf{op}}$. This Galois connection induces two other functions $\uparrow_{\mathsf{p}} : x \mapsto \neg\downarrow_{\mathsf{o}}(\neg x)$ and $\downarrow_{\mathsf{p}} : x \mapsto \neg\uparrow_{\mathsf{o}}(\neg x)$ which also define a Galois connection $(\uparrow_{\mathsf{p}},\downarrow_{\mathsf{p}})$ from $\mathcal{L}_{\mathsf{op}}$ to $\mathcal{L}_C$. Together with the negation functions $\neg_{\mathsf{o}}(x) = \neg_{\mathsf{p}}(x) = \neg x$ we obtain an aop-lattice for $\mathcal{L}_C$.*

We can prove this theorem by showing that $(\uparrow_{\mathsf{p}},\downarrow_{\mathsf{p}})$ fulfills the Galois connection properties and that the conditions for the negation functions stated in Definition 4 hold. If not stated differently we will work with such a simple form of abstraction.

4 Causes for Indefinite Results and Refinement

Whenever the optimistic and pessimistic assessment of a formula evaluated on the abstract system differ, we might be interested in refining (one of) the abstractions, to eventually obtain a result that coincides with that for the concrete system. While we leave precise ideas of refinement for future work, we elaborate a function $causes$ that returns the *causes* of why the semantics of a formula interpreted over a mv-Kripke structure using the aop-lattice is a tuple in which the left and right components differ. To determine the causes, we analyze the results of the required steps to compute the semantics of a formula by means of a standard fixpoint computation. Revisit Figure 1, in which the semantics of a mv-$\mathfrak{L}_\mu$ formula is given. Since the semantics of μ-calculus fixpoint operators can be computed in an iterative manner [13], they do not have to be treated

explicitly. The semantics of the $\Box$- and $\Diamond$-operators is given by means of transitions and meet and join operators. Thus, a relevant computation step for determining causes is basically one of (i) the evaluation of the labeling function L for some atomic proposition p and state s (ii) the evaluation of the transition relation function $\mathcal{R}$ for two states s and s', (iii) the computation of negation, or (iv) the computation of meet and join. In the latter case, the meet and join operators may be indexed by state variables iterating over the states of the Kripke structure needed to compute the semantics of the $\Box$- and $\Diamond$-operator. Therefore, to describe a computation, we define a formula language $\mathfrak{L}_\Phi$with state variables s_i by the following grammar:

$$\Phi := \neg\Phi \mid \Phi \sqcap \Phi \mid \Phi \sqcup \Phi \mid \bigsqcap_{s_i} \Phi \mid \bigsqcup_{s_i} \Phi \mid L(s_i, p) \mid \mathcal{R}(s_i, s_j)$$

Before we give a precise definition of causes, we present the intuition behind this notion. At first, let us discuss in which way the abstraction from lattices (the shift from op-lattice to aop-lattice) gives rise to imprecision. Consider the optimistic and pessimistic abstraction of the powerset lattice over the elements $1, \ldots, 5$ shown in Figures 6(a) and 6(b), respectively, in which abstract elements are named according to their concrete counter parts. Now, consider a meet of the elements $\{1, 2, 3, 4\}$ and $\{2, 3, 4, 5\}$ within the pessimistic lattice. As there is no element $\{2, 3, 4\}$, the result is $\varnothing$. Thus, the computation of a pessimistic value might be more pessimistic just because of the lattice abstraction. To identify the situations that pessimistic as well as optimistic values are diluted by such meet and join operations, we consider causes as elements in the *concrete* rather than the abstract lattice.

The other source for different optimistic and pessimistic assessment is due to abstraction by joining states. More precisely, propositions and transitions may be assessed with differing optimistic and pessimistic values. However, combining such imprecise verdicts by meet and join may actually eliminate certain imprecision: Take for example the three-valued lattice in which $(\top, \bot) \sqcap (\bot, \bot)$ yields the precise verdict $(\bot, \bot)$, thus eliminating any cause for refinement.

Hence, we should study in which way meet, join, and negation operators actually modify causes for why the respective subformulae have a differing verdict, both due to the fact that imprecision determined for subformulae may be eliminated by meet and join but also introduced due to the lattice abstraction.

We are now set to introduce our ideas formally. Let $\mathcal{L}_\mathsf{o}$ be the lattice for the optimistic and $\mathcal{L}_\mathsf{p}$ for the pessimistic view as introduced in Definition 4. The function $causes$ that returns the set of causes for one computation step from a causes domain $\mathbb{c}$ made precise below uses the result of the present computation step, the results of the directly preceding computation steps, and the causes computed for the preceding steps (its subformulae). Thus, the function $causes$ has the type:

$$causes : (\mathfrak{L}_\Phi \times \mathcal{L}_\mathsf{o} \times \mathcal{L}_\mathsf{p}) \times ((\mathfrak{L}_\Phi \to \mathcal{L}_\mathsf{o}) \times (\mathfrak{L}_\Phi \to \mathcal{L}_\mathsf{p})) \times (\mathfrak{L}_\Phi \to \mathbb{c}) \to \mathbb{c}$$

Let us now elaborate on the causes domain: We consider a cause to be a pair $(m_\mathsf{o}, m_\mathsf{p})$ of elements of the concrete lattice $\mathcal{L}_C$, which limit the possible range of values for the semantics, together with a *context*, describing the qualitative origin of the different assessment. More specifically, a context could denote one of (i) a proposition in some

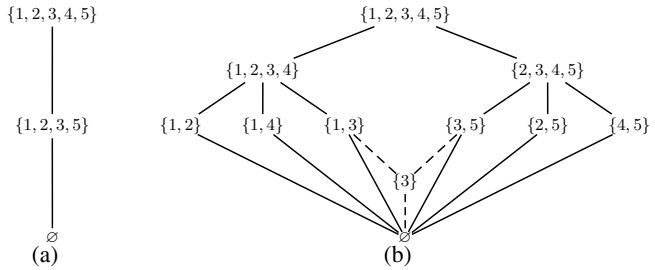


Fig. 6. An optimistic lattice $\mathcal{L}_{\mathsf{o}}$ and a pessimistic lattice $\mathcal{L}_{\mathsf{p}}$, where $\{3\}$ is added during refinement

state, (ii) some transition, (iii) or o/p identifying the lattices $\mathcal{L}_{\mathsf{p}}$ or $\mathcal{L}_{\mathsf{o}}$. The latter symbols are used to indicate the situation that the results are diluted due to abstraction of lattices. Consequently, we define the domain of causes as $\mathbb{c} := \mathcal{L}_C \times \mathbb{k}$ and the set of contexts as $\mathbb{k} := (S \times P) \cup (S \times S) \cup \{\mathsf{o}, \mathsf{p}\}$, where (i) $S \times P$ refers to a label of a state, (ii) $S \times S$ to a transition, and (iii) $\{\mathsf{o}, \mathsf{p}\}$ to the optimistic or pessimistic lattice $\mathcal{L}_{\mathsf{o}}$ or $\mathcal{L}_{\mathsf{p}}$.

In the following we will use as names for the function parameters m_{o} for the optimistic and m_{p} for the pessimistic result, ξ_{o} (ξ_{p}) for the function mapping preceding computations to their optimistic (pessimistic, resp.) results and $\zeta \subseteq \mathbb{c}$ to their causes. We give the definition of $causes$ in an inductive fashion.

Atomic propositions. For an atomic proposition p a cause is just the pair of concrete lattice elements, for which the proposition is undetermined for some state s:

$$causes(p(s), m_{\mathsf{o}}, m_{\mathsf{p}}, \xi_{\mathsf{o}}, \xi_{\mathsf{p}}, \zeta) = \{(s, p, (\downarrow_{\mathsf{o}}(m_{\mathsf{o}}), \uparrow_{\mathsf{p}}(m_{\mathsf{p}})))\}$$

For example, if an atomic proposition p evaluates to $\{1, 2, 3, 4, 5\}$ in the optimistic and to $\{2, 3, 4, 5\}$ in the pessimistic account, the cause is $(s, p, (\{1, 2, 3, 4, 5\}, \{2, 3, 4, 5\}))$.

Transitions. Causes for transitions are similarly defined as for atomic propositions:

$$causes(\mathcal{R}(s, s'), m_{\mathsf{o}}, m_{\mathsf{p}}, \xi_{\mathsf{o}}, \xi_{\mathsf{p}}, \zeta) = \{(s, s', (\downarrow_{\mathsf{o}}(m_{\mathsf{o}}), \uparrow_{\mathsf{p}}(m_{\mathsf{p}})))\}$$

Let us illustrate the causes for transitions, which may be raised by two different reasons: Several states could have been joined and therefore differently labeled transitions (or propositions respectively) could have been merged as shown in Figure 7(a) or information could have been lost due to the abstraction from the concrete lattice as shown in Figure 7(b). A combination of both cases is likewise possible as shown in Figure 7(c), where the example is driven over the powerset lattice of three elements.

Negation. When using different lattices for the optimistic and pessimistic view, negation results in a loss of precision if the complement of an element of one lattice is not exactly representable in the other lattice. A cause in this case expresses, that in one of the abstract lattices a given element is missing. Since negation never increases precision, causes for preceding computations can just be passed on. Figure 8 shows how information can be lost due to negation.

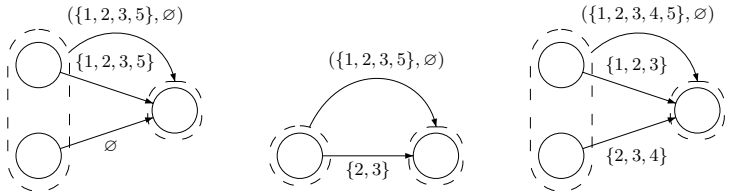


Fig. 7. Imprecision due to joining states 7(a), to lattice abstraction 7(b), and to both 7(c)

To define the cause for negation formally, let us consider one case: Recall that $m_{\mathsf{p}} = \neg\xi_{\mathsf{o}}(\varphi(s))$. If this computation would have been carried out in the concrete lattice, we would have obtained $\downarrow_{\mathsf{o}}(\xi_{\mathsf{o}}(\varphi(s)))$. If the negation of the latter value is different from $\uparrow_{\mathsf{p}}(m_{\mathsf{p}})$, we have a further imprecision due to the lattice abstraction. Otherwise, the only imprecision is due to the so far accounted ones denoted by ζ. Thus, we define *causes* as follows:

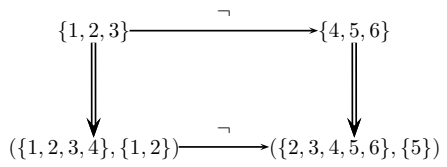


Fig. 8. Information loss due to negation. Negating $\{1,2,3\}$ results in $\{4,5,6\}$. Negating the abstraction $\{1,2,3,4\},\{1,2\}$ could yield $(\{3,4,5,6\},\{5,6\})$ but the result in this example is even less precise.

$$causes((\neg\varphi)(s), m_{\mathsf{o}}, m_{\mathsf{p}}, \xi_{\mathsf{o}}, \xi_{\mathsf{p}}, \zeta) = \zeta(\varphi(s)) \cup \begin{cases} \{(\neg\downarrow_{\mathsf{o}}(\xi_{\mathsf{o}}(\varphi(s))), \uparrow_{\mathsf{p}}(m_{\mathsf{p}}))\} & \text{if components differ} \\ \{(\uparrow_{\mathsf{p}}(\xi_{\mathsf{p}}(\varphi(s))), \neg\downarrow_{\mathsf{o}}(m_{\mathsf{o}}))\} & \text{if components differ} \end{cases}$$

Meet and join. Let us consider the case of a meet operation. The treatment of the join operator is dual and omitted here. Let $(m_{\mathsf{o}}, m_{\mathsf{p}})$ be obtained by $((\xi^1_{\mathsf{o}} \sqcap \xi^2_{\mathsf{o}}), (\xi^1_{\mathsf{p}} \sqcap \xi^2_{\mathsf{p}}))$, where $\xi^j_\sigma = \xi_\sigma(\varphi_j(s))$, for $j \in \{1,2\}$, $\sigma \in \{\mathsf{o}, \mathsf{p}\}$.

As with negation calculating the meet can, on one hand, result in further imprecision (because of the lattice abstraction). However, due to the properties of Galois connections both operations are exact on either the optimistic or the pessimistic view. Therefore we can only lose informations because of missing elements in one of the abstract lattices. So, as seen below, we add, similarly as in the case of negation, causes.

On the other hand, a meet may result in a gain of information due to the meet in the pessimistic view. The additional information can be used to remove or reduce the imprecision listed in causes of preceding computation steps. To illustrate the idea, consider the meet of $\xi^1 = (\{2,3\},\{2,3\})$ and $\xi^2 = (\{1,2,3\},\{3\})$, which results into $(m_{\mathsf{o}}, m_{\mathsf{p}}) = (\{2,3\},\{3\})$. Note that ξ^2, in the product line interpretation, denotes that no precise information about products 1 and 2 is available, while ξ^1 represents a precise result. Now, considering the result $(\{2,3\},\{3\})$, we observe that the imprecision about product 1 is no longer a concern, while that for product 2 is still of interest. Practically,

the result may be expressed by taking the cause based on $\xi^2 = (\xi^2_{\mathsf{o}}, \xi^2_{\mathsf{p}})$ and adjusting its components by the result m_{o} and m_{p} as follows: $(\xi^2_{\mathsf{o}} \sqcap m_{\mathsf{o}}, (\xi^2_{\mathsf{p}} \sqcup m_{\mathsf{p}}) \sqcap (\xi^2_{\mathsf{o}} \sqcap m_{\mathsf{o}}))$, where the meet in the second component with the first component is only to make sure that the second component is smaller than the first component. This modification is expressed by the following operator *fil*, that takes an optimistic result m_{o}, a pessimistic result m_{p} and a cause:

$$\mathit{fil}(m_{\mathsf{o}}, m_{\mathsf{p}}, (k, (l_{\mathsf{o}}, l_{\mathsf{p}}))) = (k, l_{\mathsf{o}} \sqcap \downarrow_{\mathsf{o}}(m_{\mathsf{o}}), (l_{\mathsf{p}} \sqcup \uparrow_{\mathsf{p}}(m_{\mathsf{p}})) \sqcap (l_{\mathsf{o}} \sqcap \downarrow_{\mathsf{o}}(m_{\mathsf{o}})))$$

Then we are ready to define the function causes for a meet operation as:

$$\begin{aligned} &\mathit{causes}((\varphi_1 \sqcap \varphi_2)(s), m_{\mathsf{o}}, m_{\mathsf{p}}, \xi_{\mathsf{o}}, \xi_{\mathsf{p}}, \zeta) = \\ &\qquad \{(\downarrow_{\mathsf{o}}(\xi_{\mathsf{o}}(\varphi_1(s))) \sqcap \downarrow_{\mathsf{o}}(\xi_{\mathsf{o}}(\varphi_2(s))), \uparrow_{\mathsf{p}}(m_{\mathsf{p}}))\} \text{ if components differ} \\ &\cup \bigcup_{c \in \zeta(\varphi_1(s)) \cup \zeta(\varphi_(s))} \mathit{fil}(m_{\mathsf{o}}, m_{\mathsf{p}}, c) \end{aligned}$$

Example 1. For a better understanding of the propagation of causes we consider as example the meet of $(\{1,2,3,4,5\}, \{1,2,3,4\})$ and $(\{1,2,3,4,5\}, \{2,3,4,5\})$. Using the lattices in Figures 6(a) and 6(b) for the optimistic respectively pessimistic view, this results in $(\{1,2,3,4,5\}, \varnothing)$. Calculating the meet in an exact manner would have resulted in $(\{1,2,3,4,5\}, \{2,3,4\})$. We therefore *obtain* the cause $(\mathsf{p}, (\{2,3,4\}, \varnothing))$.

Now, let us assume that the fixpoint computation (expressed by a formula of the grammar introduced above) requires to join the previous result $(\{1,2,3,4,5\}, \varnothing)$ with the pair $(\{1,2,3,4,5\}, \{1,2\})$ which results in $(\{1,2,3,4,5\}, \{1,2\})$. Since we now have information about 2, we can *modify* the cause to $(\mathsf{p}, (\{2,3,4\} \sqcap \{1,2,3,4,5\}, (\varnothing \sqcup \{1,2\}) \sqcap \{2,3,4\}))$, resulting in $(\mathsf{p}, (\{2,3,4\}, \{2\}))$.

As next step we take the meet of $(\{1,2,3,4,5\}, \{1,2\})$ and $(\{1,2,3,5\}, \{1,2\})$ which results in $(\{1,2,3,5\}, \{1,2\})$. Since we obtained information about 4, the cause is further simplified to $(\mathsf{p}, (\{2,3\}, \{2\}))$. We can now modify the lattice for the pessimistic view $\mathcal{L}_{\mathsf{o}}$ by adding an element for 3 as shown in Figure 6(b). Repeating the computation results in $(\{1,2,3,5\}, \{1,2,3\})$ and thus yield the same imprecision as we started the computation with.

Note that the treatment of causes as pairs $(m_{\mathsf{o}}, m_{\mathsf{p}})$ can be simplified in Boolean lattices to a single value $m_{\mathsf{o}} \sqcap \neg m_{\mathsf{p}}$ actually representing set difference.

The computation of causes can be interweaved with the computation of the semantics. In case of a *don't know result*, meaning that the optimistic and pessimistic assessment differ, a cause can be picked according to a meaningful heuristics, whose discussion is beyond the scope of this paper, and a suitable refinement may be accomplished. Reassessment of the semantics eventually yields a precise model checking result.

5 Conclusion

In this paper we have introduced abstraction techniques for multi-valued Kripke structures. More precisely, we have given two different kinds of abstraction: The first kind abstracts by joining states, in the usual way. The second kind of abstraction combines

lattice elements realizing the idea of not differing between certain concrete products (of a software product line) or truth values in the abstract anymore. The combination of both abstractions yields again a multi-valued Kripke structure which represents the abstract system. Here, the transitions are labeled with a tuple representing the optimistic and pessimistic assessment of this transition.

On such an abstract multi-valued Kripke structure we can evaluate properties using the existing multi-valued model checking machinery. More importantly, we have shown that the result of evaluating a mv-$\mathfrak{L}_\mu$ formula on the abstract system always yields the optimistic and pessimistic limits, in which the concrete value is located for sure.

To eventually obtain a model checking result for which the optimistic and pessimistic assessment coincide, refinement of abstractions in needed. Therefore, we have introduced the notion of causes: Whenever the optimistic and pessimistic assessment differ, a cause guides us during the process of concretization and gives us those states in the abstract, which we have to concretize first.

References

1. Sassone, V., Krukow, K., Nielsen, M.: Towards a formal framework for computational trust. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2006. LNCS, vol. 4709, pp. 175–184. Springer, Heidelberg (2007)
2. Clements, P.C., Northrop, L.: Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley, Reading (2001)
3. Gruler, A., Leucker, M., Scheidemann, K.D.: Modeling and model checking software product lines. In: Barthe, G., de Boer, F.S. (eds.) FMOODS 2008. LNCS, vol. 5051, pp. 113–131. Springer, Heidelberg (2008)
4. Chechik, M., Easterbrook, S.M., Petrovykh, V.: Model-checking over multi-valued logics. In: Oliveira, J.N., Zave, P. (eds.) FME 2001. LNCS, vol. 2021, pp. 72–98. Springer, Heidelberg (2001)
5. Bruns, G., Godefroid, P.: Model checking with multi-valued logics. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 281–293. Springer, Heidelberg (2004)
6. Chechik, M., Devereux, B., Gurfinkel, A.: Model-checking infinite state-space systems with fine-grained abstractions using spin. In: Dwyer, M.B. (ed.) SPIN 2001. LNCS, vol. 2057, pp. 16–36. Springer, Heidelberg (2001)
7. Shoham, S., Grumberg, O.: Multi-valued model checking games. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 354–369. Springer, Heidelberg (2005)
8. Bruns, G., Godefroid, P.: Model checking partial state spaces with 3-valued temporal logics. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 274–287. Springer, Heidelberg (1999)
9. Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS, pp. 203–210. IEEE Computer Society, Los Alamitos (1988)
10. Godefroid, P., Huth, M., Jagadeesan, R.: Abstraction-based model checking using modal transition systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 426–440. Springer, Heidelberg (2001)
11. Cousot, P., Cousot, R.: Abstract interpretation: A unified model for static analysis of programs by construction or approximation of fixpoints. In: Proc.4th ACM Symp. on Principles of Programming Languages, pp. 238–252 (1977)

12. Kupferman, O., Lustig, Y.: Latticed simulation relations and games. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 316–330. Springer, Heidelberg (2007)
13. Tarski, A.: A lattice-theoretical fixpoint theorem and its application. Pacific J. Math. 5, 285–309 (1955)
14. Grumberg, O., Lange, M., Leucker, M., Shoham, S.: Don't know in the μ-calculus. In: Cousot, R. (ed.) VMCAI 2005. LNCS, vol. 3385, pp. 233–249. Springer, Heidelberg (2005)