International Journal of Foundations of Computer Science


# Latticed Simulation Relations and Games

Orna Kupferman
*School of Engineering and Computer Science, Hebrew University, Jerusalem 91904, Israel*
*Email: orna@cs.huji.ac.il*

and

Yoad Lustig
*School of Engineering and Computer Science, Hebrew University, Jerusalem 91904, Israel*
*Email: yoad@cs.huji.ac.il*

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

Multi-valued Kripke structures are Kripke structures in which the atomic propositions and the transitions are not Boolean and can take values from some set. In particular, latticed Kripke structures, in which the elements in the set are partially ordered, are useful in abstraction, query checking, and reasoning about multiple view-points. The challenges that formal methods involve in the Boolean setting are carried over, and in fact increase, in the presence of multi-valued systems and logics. We lift to the latticed setting two basic notions that have been proven useful in the Boolean setting. We first define *latticed simulation* between latticed Kripke structures. The relation maps two structures $M_1$ and $M_2$ to a lattice element that essentially denotes the truth value of the statement "every behavior of $M_1$ is also a behavior of $M_2$". We show that latticed-simulation is logically characterized by the universal fragment of latticed $\mu$-calculus, and can be calculated in polynomial time. We then proceed to defining latticed two-player games. Such games are played along graphs in which each transition have a value in the lattice. The value of the game essentially denotes the truth value of the statement "the $\vee$-player can force the game to computations that satisfy the winning condition". An earlier definition of such games involved a zig-zagged traversal of paths generated during the game. Our definition involves a forward traversal of the paths, and it leads to better understanding of multi-valued games. In particular, we prove a min-max property for such games, and relate latticed simulation with latticed games.

## 1. Introduction

Several recent verification methods involve reasoning about *multi-valued Kripke structures* in which an atomic proposition is interpreted at a state as a *lattice* element, rather than a Boolean value.[a] The multi-valued setting arises directly in

[a] A lattice $\langle \mathcal{L}, \leq \rangle$ is a partially ordered set in which every two elements $a, b \in \mathcal{L}$ have a least upper bound ($a$ join $b$) and a greatest lower bound ($a$ meet $b$). We will define lattices in detail in

systems in which the designer can give to the atomic propositions rich values like "unknown" or "don't care" (c.f., the IEEE Standard Multivalue Logic System for VHDL Model Interoperability [21]), or in contexts in which one reasons about qualitative properties of systems [7]. The multi-valued setting arises indirectly in applications like abstraction methods, in which it is useful to allow the abstract system to have unknown assignments to atomic propositions and transitions [28, 4], query checking [6], which can be reduced to model checking over multi-valued Kripke structures, and verification of systems from inconsistent viewpoints [20], in which the value of the atomic propositions is the composition of their values in the different viewpoints. Recently, multi-valued logics were used to reason about systems that have a number of prioritized requirements rather than a single specification [1].

The various applications use various types of lattices (see Figure 1). For example, in the abstraction application, researchers have used three values ordered as in $\mathcal{L}_3$ [4], as well as its generalization to linear orders [9, 7, 1]. In query checking, the lattice elements are sets of formulas, ordered by the inclusion order [5]. When reasoning about inconsistent viewpoints, each viewpoint is Boolean, and their composition gives rise to products of the Boolean lattice, as in $\mathcal{L}_{2,2}$ [13]. Finally, in systems with rich values of the atomic propositions, several orders may be used, allowing the modeling of uncertainty, disagreement, and relative importance [10]. In the most general setting, both the atomic propositions and the transitions in the Kripke structure are elements in a lattice [14]. We refer to such structures as *latticed Kripke structures*.

Properties of latticed Kripke structures can be specified using multi-valued logics. In particular, [3] introduced a latticed version of the $\mu$-calculus. The value of a *latticed $\mu$-calculus* formula $\psi$ in a latticed Kripke structure $M$, denoted $[\![M, \psi]\!]$, is an element in $\mathcal{L}$ — the lattice with respect to which $M$ and $\psi$ are defined. Several model-checking algorithms for latticed $\mu$-calculus are studied in the literature [3, 29]. Automated tools for reasoning about multi-valued logics include theorem provers for first-order multi-valued logics (cf. [17, 30]) and symbolic multi-valued model checkers (cf. [8]). Naturally, the challenges that formal methods involve in the Boolean setting are carried over, and in fact increase, in the presence of multi-valued systems and logics.

In 1971, Milner defined the *simulation* pre-order between systems [26]. Simulation enjoys many appealing properties, making it a key notion in reasoning about systems and their specifications. First, simulation has a fully abstract semantics: a Kripke structure $M_2$ simulates a Kripke structure $M_1$ iff every computation tree embedded in the unrolling of $M_1$ can be embedded also in the unrolling of $M_2$. Second, simulation has a logical characterization: $M_2$ simulates $M_1$ iff every universal branching-time formula satisfied by $M_2$ is satisfied also by $M_1$ [27, 2, 16]. It follows that simulation is a suitable notion of implementation, and it is the coarsest abstraction of a system that preserves universal branching-time properties. Third, simulation can be defined locally by means of a game that relates states with their

Section 2.

immediate successor states. Based on its local definition, simulation between finite-state systems can be checked in polynomial time [11] and symbolically [18]. Finally, simulation implies trace-containment, which cannot be defined locally and requires polynomial space for verification [31]. Hence simulation is widely used both in manual and in automatic verification.

An adoption of the advantages of simulation to the multi-valued setting was partially suggested in the context of abstraction. There, *modal transition systems* (MTS) are used in order to model systems at different levels of abstraction [24]. Accordingly, atomic propositions have three possible values (false, unknown, and true), and transitions have three possible values (false, may, and must). Researchers have defined *mixed simulation* between MTSs [12, 15] and use it as a precision order: $M_1$ is simulated by $M_2$ iff $M_2$ is more precise (less abstract) than $M_1$. The adoption is partial in the sense that it fits only for the special case of MTS, and it returns a Boolean value: either $M_2$ simulates $M_1$ or it does not.

In this work we define and study *latticed simulation* in general. Given two Kripke structures $M_1$ and $M_2$ over a lattice $\mathcal{L}$, the simulation value of $M_1$ by $M_2$, denoted $sim\_val(M_1, M_2)$, is an element in $\mathcal{L}$ that essentially denotes the truth value of the statement "every behavior of $M_1$ is also a behavior of $M_2$". Technically, for two states $q_1$ of $M_1$ and $q_2$ of $M_2$, the simulation value of $q_1$ by $q_2$ refers to both the agreement between the states on the values of the atomic propositions, and to the value in which successors of $q_1$ can be matched with successors of $q_2$. The logical characterization of simulation is extended to the latticed setting: for every sentence $\psi$ in the universal fragment of latticed $\mu$-calculus, we have that $sim\_val(M_1, M_2) \leq [\![M_2, \psi]\!] \rightarrow [\![M_1, \psi]\!]$. Thus, the greater the simulation value is, the more likely it is that the value of $\psi$ in $M_1$ is not smaller than its value in $M_2$. The characterization is tight, in the sense that if $sim\_val(M_1, M_2) \not\geq l$, for a lattice value $l \in \mathcal{L}$, then there is a sentence $\psi$ in the universal latticed $\mu$-calculus such that $[\![M_2, \psi]\!] \rightarrow [\![M_1, \psi]\!] \not\geq l$. In [23], we defined the *implication value* between two latticed Kripke structures $M_1$ and $M_2$, denoted $imp\_val(M_1, M_2)$. Essentially, $imp\_val(M_1, M_2)$ denotes the truth value of the statement "every computation of $M_1$ is also a computation of $M_2$"; thus it is the latticed counterpart of trace containment. The computational advange of simulation with respect to trace containment is carried over to the latticed setting: $sim\_val(M_1, M_2) \leq imp\_val(M_1, M_2)$, and while the calculation of $imp\_val(M_1, M_2)$ is PSPACE-complete [23], $sim\_val(M_1, M_2)$ can be calculated in PTIME. We also define *latticed bisimulation* and study its properties.

It is easy to see that Boolean simulation and its properties are a special case of latticed simulation and its properties. This may create an impression as if the extension to the latticed setting is straightforward. To see that this is not the case, note that there are quite many extensions of the Boolean setting to a latticed setting that coincide with the Boolean setting for the Boolean lattice. For example, we could have defined latticed simulation so that if the simulation value of $M_1$ by $M_2$ is a lattice element $l$, then for all universal formulas $\psi$, if $[\![M_2, \psi]\!] \geq l$, then $[\![M_1, \psi]\!] \geq l$. To see that this definition is different, consider the three-valued linearly-ordered lattice $\{0, \frac{1}{2}, 1\}$ and assume there is a formula $\psi$ such that $[\![M_2, \psi]\!] =$

$\frac{1}{2}$ and $[\![M_1, \psi]\!] = 0$. A simulation value of $\frac{1}{2}$ is possible according to our definition (indeed, $\frac{1}{2} \geq (\frac{1}{2} \rightarrow 0)$) but is not possible according to the alternative definition (indeed, $[\![M_2, \psi]\!] \geq \frac{1}{2}$ yet $[\![M_1, \psi]\!] \not\geq \frac{1}{2}$). Our search for a good definition have eventually converged to the suggested one — the only definition that enjoys all the helpful properties of Boolean simulation.

Recall that in the Boolean case, simulation can be defined by means of a game between two players. We define and study *latticed games* and show that latticed simulation can be defined by means of such games. An earlier definition of latticed games is presented in [29]. As in our setting, the game graph is a latticed Kripke structure whose states are partitioned into $\vee$-states and $\wedge$-states. Also, the game is defined so that its value is a lattice element that essentially denotes the truth value of the statement "the $\vee$-player can force the games into computations that satisfy the winning condition". The definition of the value of a game in [29], however, is conceptually different from the definition of the winner in a Boolean two-players game. Indeed, the value of a play in the game in [29] is defined as the limit of a sequence $\{val_i\}_{i=0}^{\infty}$, where each value $val_i$ is computed by backward traversal of the prefix $v_0, v_1, \ldots, v_i$ of the path generated during the play. Thus, while in Boolean games the generated path is traversed in a forward manner, here the need to calculate a lattice value has forced a zig-zagged traversal.

Our definition of a value of a game avoids the zig-zagged traversal and is based on a mutual definition of two values: one for the $\vee$-player and one for the $\wedge$ player. The values are updated during the play, and the values after the $i$-th transition depends on the values before the $i$-th transition and the value of the edge traversed during the $i$-th transition. The value of a game according to our definition coincides with the value in [29]. The fact our definition resembles the forward traversal in Boolean games leads to a better understanding of latticed games. In particular, we prove a min-max theorem for latticed games: the value of a game for the $\vee$-player complements the value of the game for the $\wedge$-player.[b] We note that this result is technically very challenging. In particular, unlike Boolean games, the value of a game need not be achieved with a single strategy. Beyond the relation between latticed games and latticed simulation, they are of independent interest. In particular, as discussed in [29], model checking of latticed $\mu$-calculus can be reduced to latticed-game solving.

## 2. Preliminaries

Let $\langle A, \leq \rangle$ be a partially ordered set, and let $P$ be a subset of $A$. An element $a \in A$ is an *upper bound* on $P$ if $a \geq b$ for all $b \in P$. Dually, $a$ is a *lower bound* on $P$ if $a \leq b$ for all $b \in P$. An element $a \in A$ is the *least element of* $P$ if $a \in P$ and $a$ is a lower bound on $P$. Dually, $a \in A$ is the *greatest element of* $P$ if $a \in P$ and $a$ is an upper bound on $P$. A partially ordered set $\langle A, \leq \rangle$ is a *lattice* if for every two elements $a, b \in A$ both the least upper bound and the greatest lower bound of $\{a, b\}$ exist, in which case they are denoted $a \vee b$ ($a$ *join* $b$) and $a \wedge b$ ($a$ *meet* $b$), respectively. A lattice is *complete* if for every subset $P \subseteq A$ both the least upper

[b] In multi-valued games, *determinancy* is generalized to having a min-max property.

bound and the greatest lower bound of $P$ exist, in which case they are denoted $\bigvee P$ and $\bigwedge P$, respectively. In particular, $\bigvee A$ and $\bigwedge A$ are denoted $\top$ (*top*) and $\bot$ (*bottom*), respectively. A lattice $\langle A, \leq \rangle$ is finite if $A$ is finite. Note that every finite lattice is complete. A lattice is *distributive* if for every $a, b, c \in A$, we have $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

The traditional disjunction and conjunction logic operators correspond to the join and meet lattice operators. In a general lattice, however, there is no natural counterpart to negation. A *De Morgan* (or *quasi-Boolean*) lattice is a lattice in which every element $a$ has a unique complement element $\neg a$ such that $\neg\neg a = a$, De Morgan rules hold, and $a \leq b$ implies $\neg b \leq \neg a$. In the rest of the paper we consider only finite [c] distributive De Morgan lattices.

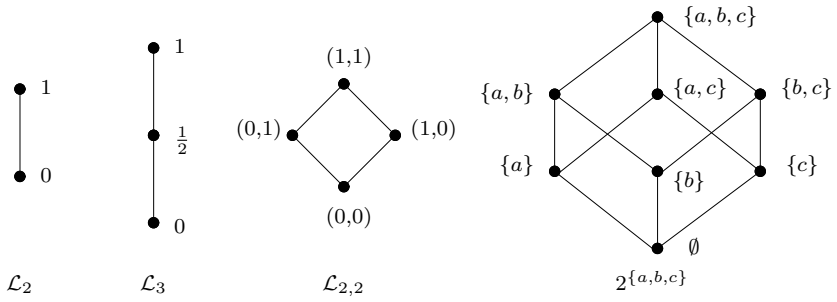


Figure 1: Some lattices.

In Figure 1 we describe some (finite distributive De Morgan) lattices. The elements of the lattice $\mathcal{L}_2$ are the usual truth values 1 (**true**) and 0 (**false**) with the order $0 \leq 1$. The lattice $\mathcal{L}_3$ contains in addition the value $\frac{1}{2}$, with the order $0 \leq \frac{1}{2} \leq 1$, and with negation defined by $\neg 0 = 1$ and $\neg\frac{1}{2} = \frac{1}{2}$.

The lattice $\mathcal{L}_{2,2}$ is the Cartesian product of two $\mathcal{L}_2$ lattices, thus $(a, b) \leq (a', b')$ if both $a \leq a'$ and $b \leq b'$. Also, $\neg(a, b) = (\neg a, \neg b)$. Finally, the lattice $2^{\{a,b,c\}}$ is the power set of $\{a, b, c\}$ with the set-inclusion order (that is, the transitive closure of the edges in the figure). Complementation is interpreted as set complementation relative to $\{a, b, c\}$. In this lattice, for example, $\{a\} \vee \{b\} = \{a, b\}$, $\{a\} \wedge \{b\} = \bot$, $\{a, c\} \vee \{b\} = \top$, and $\{a, c\} \wedge \{b\} = \bot$.

A *join irreducible* element is a value $l \in \mathcal{L}$ such that $l \neq \bot$ and for all $l_1, l_2 \in \mathcal{L}$, if $l_1 \vee l_2 \geq l$, then $l_1 \geq l$ or $l_2 \geq l$. For example, in $\mathcal{L}_3$ (and in every linear order), all elements are join irreducible. On the other hand, in the lattice $2^{\{a,b,c\}}$, the elements $\{a\}, \{b\}, \{c\}$, and are join irreducible, but $\{a, b\}, \{b, c\}$, and $\{a, c\}$ are not join irreducible. To see the latter, note that $\{a\} \vee \{b, c\} \geq \{a, c\}$ but $\{a\} \not\geq \{a, c\}$ and $\{b, c\} \not\geq \{a, c\}$. Birkhoff's representation theorem for finite distributive lattices implies that in order to prove that $l_1 = l_2$ it is sufficient if to prove that for every

[c] Note that focusing on finite lattices is not as restrictive as may first seem. Indeed, even when the lattice is infinite, the problems we consider involve only finite Kripke structures. Therefore, only a finite number of lattice elements appear in a problem, and since the lattice is distributive, the closure of logical operations on these values is still finite.

join irreducible element $l$ it holds that $l_1 \geq l$ iff $l_2 \geq l$. We denote the set of join irreducible elements of $\mathcal{L}$ by $JI(\mathcal{L})$. A *meet irreducible* element $l \in \mathcal{L}$ is a value for which if $l_1 \wedge l_2 \leq l$ then either $l_1 \leq l$ or $l_2 \leq l$. Note that in a De Morgan lattice, an element is meet irreducible iff its complement is join irreducible.

Consider a lattice $\mathcal{L}$ (we abuse notation and refer to $\mathcal{L}$ also as a set of elements, rather than a pair of a set with an order on it). For a set $X$ of elements, an *$\mathcal{L}$-set over $X$* is a function $S : X \to \mathcal{L}$ assigning to each element of $X$ a value in $\mathcal{L}$. It is convenient to think about $S(x)$ as the truth value of the statement "$x$ is in $S$". We say that an $\mathcal{L}$-set $S$ is *Boolean* if $S(x) \in \{\top, \bot\}$ for all $x \in X$. The usual set operators can be lifted to $\mathcal{L}$-sets as expected. Given two $\mathcal{L}$-sets $S_1$ and $S_2$ over $X$, we define join, meet, and complementation so that for every element $x \in X$, we have $S_1 \vee S_2(x) = S_1(x) \vee S_2(x)$, $S_1 \wedge S_2(x) = S_1(x) \wedge S_2(x)$, and $comp(S_1)(x) = \neg S_1(x)$.[d]

A *latticed Kripke structure* is a 6-tuple $M = \langle \mathcal{L}, AP, Q, Q_0, R, \Theta \rangle$, where $\mathcal{L}$ is a lattice, $AP$ is a set of atomic propositions, $Q$ is set of states, $Q_0$ is an $\mathcal{L}$-set of initial states, $R : Q \times Q \to \mathcal{L}$ is an $\mathcal{L}$-set of transitions, and $\Theta : AP \to \mathcal{L}^Q$ maps each atomic proposition $p$ to an $\mathcal{L}$-set of states, describing the truth value of $p$ in each state.

The $\mu$-calculus [22] is an expressive temporal logic that subsumes logics like LTL and CTL$^\star$. A multi-valued variant of the $\mu$-calculus, was suggested and studied in [3]. We call this logic *latticed $\mu$-calculus* (LMC, for short)[e]. For technical convenience, we consider LMC formulas in a positive form in which negation applies only to atomic propositions. Given a set $AP$ of atomic propositions and a set $Var$ of variables, LMC formulas have the following syntax, with $p \in AP$ and $y \in Var$.

$$\varphi = p | \neg p | \varphi_1 \vee \varphi_2 | \varphi_1 \wedge \varphi_2 | AX\varphi | EX\varphi | \mu y.\varphi(y) | \nu y.\varphi(y)$$

Note that the operators $\vee$ and $\wedge$ stand for "join" and "meet", rather than "or" and "and". In fixed-point formulas $\mu y.\varphi(y)$ and $\nu y.\varphi(y)$, the operators $\mu$ and $\nu$ bind free occurrences of $y$ in $\varphi$. We use $\varphi_1 \to \varphi_2$ to abbreviate (the positive normal form of) $\neg\varphi_1 \vee \varphi_2$.

A *valuation* $\mathcal{V} : Var \to \mathcal{L}^Q$ over a lattice $\mathcal{L}$ maps the variables in $Var$ into $\mathcal{L}$-sets of states. We write $\mathcal{V}_\bot$ for the valuation that maps every variable to the $\mathcal{L}$-set that maps every state to $\bot$ (that it, for every $y \in Var$ and $q \in Q$, it holds that $\mathcal{V}_\bot(y)(q) = \bot$). For a variable $y \in Var$ and an $\mathcal{L}$-set $l$, we write $\mathcal{V}[y = l]$ for the valuation that agrees with $\mathcal{V}$ except that it maps $y$ to $l$.

The semantics of an LMC formula $\varphi$ is defined with respect to a latticed Kripke structure $M$ and a valuation $\mathcal{V}$ to the free variables in $\varphi$. Given such a valuation, the formula induces an $\mathcal{L}$-set of states, denoted $[\![M, \varphi]\!]_\mathcal{V}$, in which each state $s$ of $M$ is mapped to a value in $\mathcal{L}$ describing the truth value of the formula in $s$. Accordingly, given $\mathcal{V}$, a formula $\varphi$ with a free variable $y$ can be viewed as a transformer $f_\varphi^{\mathcal{V},\dagger} : \mathcal{L}^Q \to \mathcal{L}^Q$, defined by $f_\varphi^{\mathcal{V},y}(g) = [\![M, \varphi]\!]_{\mathcal{V}[y=g]}$. We use $\mu f_\varphi^{\mathcal{V},y}$ and $\nu f_\varphi^{\mathcal{V},y}$ to denote

[d] If $S_1$ and $S_2$ are over different domains $X_1$ and $X_2$, we can view them as having the same domain $X_1 \cup X_2$ and let $S_1(x) = \bot$ for $x \in X_2 \setminus X_1$ and $S_2(x) = \bot$ for $x \in X_1 \setminus X_2$.

[e] The logic is termed $\mu L$ in [3], and is termed $\mathcal{L}_\mu$ in [29]. We prefer a terminology that does not involve mathematical symbols.

the the least and greatest fixed-points of $f_\varphi^{\mathcal{V},y}$ with respect to $\mathcal{V}$. By the Tarksi-Knaster Theorem, these fixed-points exist.

Formally, the *interpretation* $[\![M,\varphi]\!]_\mathcal{V}$ of an LMC formula $\varphi$ in a latticed Kripke structure $M = \langle \mathcal{L}, AP, Q, Q_0, R, \Theta\rangle$ and valuation $\mathcal{V}$ over a complete lattice $\mathcal{L}$ is defined as follows:[f]

$$
\begin{array}{rcl@{\qquad}rcl}
[\![M,p]\!]_\mathcal{V} & = & \Theta(p) & [\![M,\varphi_1\vee\varphi_2]\!]_\mathcal{V} & = & [\![M,\varphi_1]\!]_\mathcal{V}\vee[\![M,\varphi_2]\!]_\mathcal{V}\\
[\![M,\neg p]\!]_\mathcal{V} & = & comp(\Theta(p)) & [\![M,\varphi_1\wedge\varphi_2]\!]_\mathcal{V} & = & [\![M,\varphi_1]\!]_\mathcal{V}\wedge[\![M,\varphi_2]\!]_\mathcal{V}\\
[\![M,\mu y.\varphi]\!]_\mathcal{V} & = & \mu f_\varphi^{\mathcal{V},y} & [\![M,EX\varphi]\!]_\mathcal{V} & = & \lambda s.\ \bigvee_{s'\in Q}(R(s,s')\wedge[\![M,\varphi]\!]_\mathcal{V}(s'))\\
[\![M,\nu y.\varphi]\!]_\mathcal{V} & = & \nu f_\varphi^{\mathcal{V},y} & [\![M,AX\varphi]\!]_\mathcal{V} & = & \lambda s.\ \bigwedge_{s'\in Q}(R(s,s')\rightarrow[\![M,\varphi]\!]_\mathcal{V}(s'))\\
[\![M,y]\!]_\mathcal{V} & = & \mathcal{V}(y) & & &
\end{array}
$$

A formula in which every variable is in the scope of a fixed-point operator is a *sentence*. If $\varphi$ is a sentence, we write $[\![(M,s),\varphi]\!]$ for the value $[\![M,\varphi]\!]_{\mathcal{V}_\perp}(s)$. Since we almost exclusively deal with sentences, we abuse notation and omit the valuation $\mathcal{V}_\perp$.

The *value* of an LMC sentence $\varphi$ in a latticed Kripke structure $M$, denoted $[\![M,\varphi]\!]$, is $\bigwedge_{q\in Q}(Q_0(q)\rightarrow[\![M,\varphi]\!](q))$. Note that we get the standard Boolean semantics of $\mu$-calculus as a special case.

The universal fragment of LMC, termed ALMC, consists of the formulas that do not contain the $EX$ operator. Note that since we assume that formulas are in a positive normal form, the above syntactic restriction implies that indeed formulas of ALMC can only impose universal requirements. Finally, the fixed-point free fragment consists of formulas that do not contain a fixed-point operator. Thus, it corresponds to a latticed version of Modal Logic, we term it LML, and term its universal fragment ALML.

**Example 1** Consider the latticed Kripke structure $M = \langle 2^{\{a,b,c\}}, \{p\}, \{q_0,q_1,q_2\}, Q_0, R, \Theta\rangle$, where $Q_0(q_0) = \top$ and $Q_0(q_1) = Q_0(q_2) = \perp$, appearing in Figure 2. The values of $R$ and $\Theta$ are described in the figure (we describe only transitions with value greater than $\emptyset$; in the description of the value of the transitions and the value of $p$ inside the states, we omit the $\{\}$ notation). For example, $\Theta(p)(q_0) = \{a,b\}$ and $R(q_0,q_1) = \{b,c\}$. Also, $Q_0(q_0) = \{a,b,c\}$ and the initial value of $q_1$ and $q_2$ is $\emptyset$.

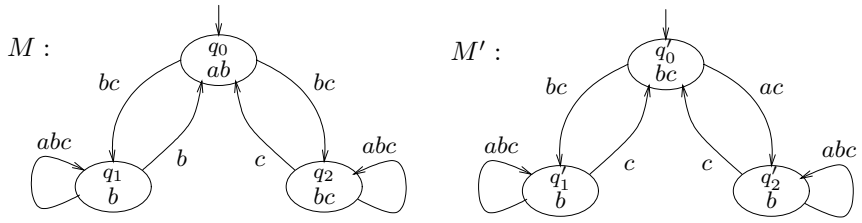


Figure 2: Latticed Kripke structures over $\mathcal{L} = 2^{\{a,b,c\}}$.

Below we describe the truth value of some LMC formulas in $M$. Since $q_0$ is the only state with $Q_0(q_0) \neq \perp$ and $Q_0(q_0) = \{a,b,c\}$, we have $[\![M,\varphi]\!] = [\![(M,q_0)]\!](\varphi)$.

[f] We use $\lambda s.\theta(s)$ to denote the $\mathcal{L}$-set in which each state $s$ is mapped to $\theta(s)$.

- $[\![M, p]\!] = \{a, b\}$.
- $[\![M, EXp]\!] = (\{b, c\} \wedge \{b\}) \vee (\{b, c\} \wedge \{b, c\}) = \{b, c\}$.
- $[\![M, AXp]\!] = (\{b, c\} \rightarrow \{b\}) \wedge (\{b, c\} \rightarrow \{b, c\}) = (\{a, b\} \wedge \{a, b, c\}) = \{a, b\}$.

Note that in the Boolean case, a state may satisfy $AX\theta$ without satisfying $EX\theta$ only if it does not have successors. In the latticed setting, the transitions to the successors have values. This is why the value of $EX\theta$ may not be greater than the value of $AX\theta$, as we see here.

Let us now calculate $[\![M, \nu z.p \wedge AXz]\!]$; that is, the truth value of "$p$ holds at all reachable states". Let $\theta(z) = p \wedge AXz$. We start with $z_0$ that maps all states to $\{a, b, c\}$. We then iterate $\theta$ as described in Table 1. A fixed-point is reached when $z_2 = z_3$, and $[\![M, \nu z.p \wedge AXz]\!] = \{a, b\}$. Intuitively, $p$ holds at all reachable states from both viewpoints $a$ and $b$: From viewpoint $b$, the proposition $p$ indeed holds at all states. From viewpoint $a$, the proposition $p$ does not hold in states $q_1$ and $q_2$, but these states are not reachable.

| | $q_0$ | $q_1$ | $q_2$ |
|---|---|---|---|
| $z_0$ | {a,b,c} | {a,b,c} | {a,b,c} |
| $z_1$ | {a,b} | {b} | {b,c} |
| $z_2$ | {a,b} | {b} | {b} |
| $z_3$ | {a,b} | {b} | {b} |

Table 1: $[\![M, \nu z.p \wedge AXz]\!]$ calculation.

□

## 3. Latticed Simulation

In this section we define latticed simulation — an extention of the definition of the simulation pre-order of [26] to the latticed context. Let $M_1 = \langle \mathcal{L}, AP, Q_1, Q_0^1, R_1, \Theta_1 \rangle$ and $M_2 = \langle \mathcal{L}, AP, Q_2, Q_0^2, R_2, \Theta_2 \rangle$ be two latticed Kripke structures. In the Boolean case, a relation $S \subseteq Q_1 \times Q_2$ is a simulation relation if two conditions are satisfied: First, simulating states satisfy the same propositions. Second, if $q_2 \in Q_2$ simulates $q_1 \in Q_1$ then for every successor $q_1'$ of $q_1$ there exists a successor $q_2'$ of $q_2$ such that $q_2'$ simulates $q_1'$. In the latticed setting, the simulation relation is an $\mathcal{L}$-set $S \in \mathcal{L}^{Q \times Q}$. Intuitively, $S(q_1, q_2)$ describes the truth value of the statement "every behavior of $M_1$ is also a behavior of $M_2$". As in the Boolean case, the simulation value of a pair of states $q_1$ and $q_2$ depends both in agreement on the values of the atomic propositions in $q_1$ and $q_2$ and in the ability to match successors of $q_1$ with successors of $q_2$.

We capture the first condition by the value

$$S_{AP}(q_1, q_2) = \bigwedge_{p \in AP} [\![M_1, p]\!](q_1) \leftrightarrow [\![M_2, p]\!](q_2).$$

We capture the second condition by the value

$$S_R(q_1, q_2) = \bigwedge_{q_1' \in Q_1} \left( R_1(q_1, q_1') \rightarrow \bigvee_{q_2' \in Q_2} (R_2(q_2, q_2') \wedge S(q_1', q_2')) \right).$$

An $\mathcal{L}$-relation $S : Q_1 \times Q_2 \rightarrow \mathcal{L}$ is an *$\mathcal{L}$-simulation* from $M_1$ to $M_2$ if for all $q_1 \in Q_1$ and $q_2 \in Q_2$, we have $S(q_1, q_2) = S_{AP}(q_1, q_2) \wedge S_R(q_1, q_2)$.

**Example 2** Consider the latticed Kripke structures $M$ and $M'$ appearing in Figure 2. The following is an $\mathcal{L}$-simulation from $M$ to $M'$.

- $S(q_0, q_0') = S(q_0, q_1') = S(q_0, q_2') = S(q_1, q_0') = S(q_1, q_2') = S(q_2, q_1') = \{b\}$,
- $S(q_2, q_0') = S(q_2, q_2') = \{a, b\}$,
- $S(q_1, q_1') = \{a, c, b\}$.

Let us verify $S(q_2, q_0')$. First, $S_{AP}(q_2, q_0') = \{a, b, c\}$. Now, $S_R(q_2, q_0') = S_R^0 \wedge S_R^1 \wedge S_R^2$, where $S_R^i = R(q_2, q_i) \rightarrow \bigvee_{q' \in Q'} (R'(q_0', q') \wedge S(q_i, q'))$. For example, $S_R^0 = R(q_2, q_0) \rightarrow ((R'(q_0', q_0') \wedge S(q_0, q_0')) \vee (R'(q_0', q_1') \wedge S(q_0, q_1')) \vee (R'(q_0', q_2') \wedge S(q_0, q_2'))) = \{c\} \rightarrow ((\emptyset \wedge \{b\}) \vee (\{b, c\} \wedge \{b\}) \vee (\{a, c\} \wedge \{b\}) = \{a, b\}$. Also, $S_R^1 = \{a, b, c\}$, and $S_R^2 = \{a, b\}$. Hence, $S_R(q_2, q_0') = \{a, b\}$ and $S(q_2, q_0') = \{a, b\}$. □

In the Boolean setting, $S(q_1, q_2)$ guarantees that all universal $\mu$-calculus formulas that are satisfied in $q_2$ are also satisfied in $q_1$. In the latticed setting, we have the following.

**Theorem 3** *Consider an $\mathcal{L}$-simulation $S : Q_1 \times Q_2 \rightarrow \mathcal{L}$. For all states $q_1 \in Q_1$ and $q_2 \in Q_2$ and for all ALMC sentences $\psi$, we have $S(q_1, q_2) \leq [\![M_2, \psi]\!](q_2) \rightarrow [\![M_1, \psi]\!](q_1)$.*

**Proof:** The proof proceeds by induction of the structure of $\psi$. For atomic propositions, the claim follows from the fact that $S(q_1, q_2) \leq S_{AP}(q_1, q_2)$. Indeed, $S_{AP}(q_1, q_2) \leq [\![M_1, p]\!](q_1) \leftrightarrow [\![M_2, p]\!](q_2) \leq [\![M_2, p]\!](q_2) \rightarrow [\![M_1, p]\!](q_1)$.

For the induction step, we consider the different possible structures of $\psi$. Assume first that $\psi = \varphi_1 \vee \varphi_2$. By the induction hypothesis, we have $S(q_1, q_2) \leq \neg[\![M_2, \varphi_1]\!](q_2) \vee [\![M_1, \varphi_1]\!](q_1)$ and $S(q_1, q_2) \leq \neg[\![M_2, \varphi_2]\!](q_2) \vee [\![M_1, \varphi_2]\!](q_1)$. Now,

$$\begin{array}{rcl} S(q_1, q_2) & = & S(q_1, q_2) \wedge S(q_1, q_2) \\ & \leq & (\neg[\![M_2, \varphi_1]\!](q_2) \vee [\![M_1, \varphi_1]\!](q_1)) \wedge (\neg[\![M_2, \varphi_2]\!](q_2) \vee [\![M_1, \varphi_2]\!](q_1)) \\ & \leq & (\neg[\![M_2, \varphi_1]\!](q_2) \wedge \neg[\![M_2, \varphi_2]\!](q_2)) \vee ([\![M_1, \varphi_1]\!](q_1) \vee [\![M_1, \varphi_2]\!](q_1)) \\ & = & \neg([\![M_2, \varphi_1]\!](q_2) \vee [\![M_2, \varphi_2]\!](q_2)) \vee ([\![M_1, \varphi_1]\!](q_1) \vee [\![M_1, \varphi_2]\!](q_1)) \\ & = & \neg[\![M_2, \varphi_1 \vee \varphi_2]\!](q_2) \vee [\![M_1, \varphi_1 \vee \varphi_2]\!](q_1). \end{array}$$

The proof for $\psi = \varphi_1 \wedge \varphi_2$ is simular. By the induction hypothesis, we have $S(q_1, q_2) \leq \neg[\![M_1, \varphi_1]\!](q_1) \vee [\![M_2, \varphi_1]\!](q_2)$ and $S(q_1, q_2) \leq \neg[\![M_1, \varphi_2]\!](q_1) \vee [\![M_2, \varphi_2]\!](q_2)$. Now,

$$\begin{array}{rcl} S(q_1, q_2) & = & S(q_1, q_2) \wedge S(q_1, q_2) \\ & \leq & (\neg[\![M_2, \varphi_1]\!](q_2) \vee [\![M_1, \varphi_1]\!](q_1)) \wedge (\neg[\![M_2, \varphi_2]\!](q_2) \vee [\![M_1, \varphi_2]\!](q_1)) \\ & = & (\neg[\![M_2, \varphi_1]\!](q_2) \vee \neg[\![M_2, \varphi_2]\!](q_2)) \vee ([\![M_1, \varphi_1]\!](q_1) \wedge [\![M_1, \varphi_2]\!](q_1)) \\ & = & \neg([\![M_2, \varphi_1]\!](q_2) \wedge [\![M_2, \varphi_2]\!](q_2)) \vee ([\![M_1, \varphi_1]\!](q_1) \wedge [\![M_1, \varphi_2]\!](q_1)) \\ & = & \neg[\![M_1, \varphi_1 \wedge \varphi_2]\!](q_1) \vee [\![M_2, \varphi_1 \wedge \varphi_2]\!](q_2). \end{array}$$

For $\psi = AX\varphi$, we proceed as follows. By definition, $S(q_1,q_2) \leq S_R(q_1,q_2) \leq \bigwedge_{q_1'} \neg R_1(q_1,q_1') \vee \bigvee_{q_2'}(R_2(q_2,q_2') \wedge S(q_1',q_2'))$. By the induction hypothesis, we have:

$$\begin{array}{rcl} S(q_1,q_2) & \leq & \bigwedge_{q_1'}(\neg R_1(q_1,q_1') \vee \bigvee_{q_2'}(R_2(q_2,q_2') \wedge (\neg[\![M_2,\varphi]\!](q_2') \vee [\![M_1,\varphi]\!](q_1')))) \\ & \leq & \bigwedge_{q_1'}(\neg R_1(q_1,q_1') \vee \bigvee_{q_2'}((R_2(q_2,q_2') \wedge \neg[\![M_2,\varphi]\!](q_2')) \vee [\![M_1,\varphi]\!](q_1'))) \\ & = & (\bigvee_{q_2'}(R_2(q_2,q_2') \wedge \neg[\![M_2,\varphi]\!](q_2'))) \vee (\bigwedge_{q_1'}(\neg R_1(q_1,q_1') \vee [\![M_1,\varphi]\!](q_1')) \\ & = & \neg(\bigwedge_{q_2'}(\neg R_2(q_2,q_2') \vee [\![M_2,\varphi]\!](q_2'))) \vee (\bigwedge_{q_1'}(\neg R_1(q_1,q_1') \vee [\![M_1,\varphi]\!](q_1'))) \\ & = & \neg[\![M_2,AX\varphi]\!](q_2) \vee [\![M_1,AX\varphi]\!](q_1). \end{array}$$

It is left to prove the case $\psi$ is a fixed-point formula. We describe the case $\psi = \mu y.\varphi(y)$. The proof for $\nu y.\varphi(y)$ is similar. By the Tarski-Knaster Theorem, the value of a fixed point formula can be calculated iteratively: start with $y_0 = \mathcal{V}_{\perp/}$, and for all $i > 0$, define $y_{i+1} = \varphi(y_i)$ until a fixed-point is reached.

The proof of our claim follows from proving by induction on $i$ that for every $i \geq 0$, it holds that $S(q_1,q_2) \leq y_i(q_2) \rightarrow y_i(q_1)$. For the base case (i.e. $i = 0$), we have $S(q_1,q_2) \leq (\top \vee \perp) = \top$. To finish the proof, we have to prove that $S(q_1,q_2) \leq y_i(q_2) \rightarrow y_i(q_1)$ implies $S(q_1,q_2) \leq y_{i+1}(q_2) \rightarrow y_{i+1}(q_1)$. Since $y_i(q) = \varphi(q)$, the proof of the induction step can be done by an induction on the structure of $\varphi$. In order to prove the induction here, we need to prove the claim for all operators other than fixed-point operators. This, however, was already done above (outside the fixed-point operators case). □

Note that the relation $S_R$ depends on $S$, thus there may be several latticed-simulation relations. We define the *maximal simulation relation* to be the relation $S^*$ that maps every pair of states to the join of their image under every simulation relation. Formally, we define $S^*(q_1,q_2) = \bigvee_{S \in SL(M_1,M_2)} S(q_1,q_2)$, where $SL(M_1,M_2)$ is the set of simulation relations from $M_1$ to $M_2$.

We now justify the definition by showing that the maximal simulation relation is indeed a simulation relation, and furthermore, it can be easily computed.

**Theorem 4** *The relation $S^*$ is a simulation relation, and it can be computed in polynomial time.*

**Proof:** We compute the simulation relation $S^*$ in iterations. Let $S_0 = S_{AP}$. Note that $S_{AP}$ can be computed in time $O(|Q_1||Q_2||AP|)$. Next, we iteratively compute, for $i > 0$ the $\mathcal{L}$-relation $S_{i+1}$, defined as

$$S_{i+1}(q_1,q_2) = S_i(q_1,q_2) \wedge \bigwedge_{q_1' \in Q_1} R_1(q_1,q_1') \rightarrow \bigvee_{q_2' \in Q_2} (R_2(q_2,q_2') \wedge S_i(q_1',q_2')).$$

Note that $S_{i+1}$ can be computed from $S_i$ in time $O(|Q_1|^2,|Q_2|^2)$. Since $S_{i+1}(q_1,q_2) \leq S_i(q_1,q_2)$ it follows trivially that a fixed point $S^*$ will be reached within $|\mathcal{L}||Q_1||Q_2|$ iterations[g]. Thus, assuming $|\mathcal{L}|$ and $|AP|$ are fixed, $S^*$ can be computed in time $|Q_1|^3|Q_2|^3$.

It is not hard to see that $S^*$ (as a fixed point of the process) is a simulation relation. It is also not hard to prove by induction on $i$ that for every simulation

[g] In fact, since by Birkhoff's representation theorem every value can be represented as the set of join irreducible elements lesser or equal from it, it is not hard to see that a fixed point is be reached within $|JI(\mathcal{L})| \cdot |Q_1| \cdot |Q_2|$ iterations.

relation $S'$, and two states $q_1 \in Q_1, q_2 \in Q_2$, it holds that $S^*(q_1,q_2) \geq S(q_1,q_2)$. Since $S^*$ itself is a simulation relation, we get that $S^*$ is the maximal simulation relation. $\square$

We now show that logical characterization by ALMC is tight for the maximal simulation.

**Theorem 5** *For every pair of states $q_1 \in Q_1$ and $q_2 \in Q_2$, and value $l \in \mathcal{L}$, if $S^*(q_1,q_2) \not\geq l$, then there exists an ALML formula $\varphi$ such that $[\![M_2,\varphi]\!](q_2) \rightarrow [\![M_1,\varphi]\!](q_1) \not\geq l$.*

**Proof:** By Birkhoff's representation theorem, if two values $l_1$ and $l_2$ are such that $l_1 \not\geq l_2$, then there exists a join irreducible value $l'$ such that $l_2 \geq l'$ and $l_1 \not\geq l'$. Therefore, it is enough to prove the claim for values $l$ that are join irreducible.

For $i \geq 0$, denote by $S_i$ the $i$-th latticed relation defined in the proof of Theorem 4. Let $j$ be the minimal index for which $S_j(q_1,q_2) \not\geq l$. We prove the claim by induction on $j$.

For $j = 0$, we get that $S_{AP}(q_1,q_2) \not\geq l$. Therefore, by the definition of $S_{AP}$, there exists an atomic proposition $p$, or a negation of one $\neg p$ (assume w.l.o.g. we deal with an atomic proposition $p$), such that $[\![M_1,p]\!](q_1) \leftrightarrow [\![M_2,p]\!](q_2) \not\geq l$. Thus, either $[\![M_2,p]\!](q_2) \rightarrow [\![M_1,p]\!](q_1) \not\geq l$ (and we are done) or $[\![M_1,p]\!](q_1) \rightarrow [\![M_2,p]\!](q_2) \not\geq l$. In the latter case, both $\neg[\![M_1,p]\!](q_1) \not\geq l$ and $[\![M_2,p]\!](q_2) \not\geq l$. Thus, both $[\![M_1,\neg p]\!](q_1) \not\geq l$ and $\neg[\![M_2,\neg p]\!](q_2) \not\geq l$ (note the double negation in the last inequality). By join irreducibility of $l$, we get that $\neg[\![M_2,\neg p]\!](q_2) \vee [\![M_1,\neg p]\!](q_1) \not\geq l$. Therefore, the claim holds for either $\varphi = p$ or $\varphi = \neg p$.

Assume now that the claim holds for $j$, we prove it for $j+1$. By definition, $S_{j+1}(q_1,q_2) = S_j(q_1,q_2) \wedge \bigwedge_{q_1' \in Q_1} R_1(q_1,q_1') \rightarrow \bigvee_{q_2' \in Q_2}(R_2(q_2,q_2') \wedge S_j(q_1',q_2'))$.

By the minimality of $j+1$, we have $S_j(q_1,q_2) \geq l$. Therefore, there exists $q_1' \in Q_1$ for which $R_1(q_1,q_1') \rightarrow (R_2(q_2,q_2') \wedge S_j(q_1',q_2')) \not\geq l$, for every $q_2' \in Q_2$. This means that both $\neg R_1(q_1,q_1') \not\geq l$ and $R_2(q_2,q_2') \wedge S_j(q_1',q_2') \not\geq l$ for every $q_2' \in Q_2$. In particular, for all $q_2' \in Q_2$, either $R_2(q_2,q_2') \not\geq l$ or $S_j(q_1',q_2') \not\geq l$. In the latter case, let $\varphi_{q_2'}$ be such that $[\![M_2,\varphi_{q_2'}]\!](q_2') \rightarrow [\![M_1,\varphi_{q_2'}]\!](q_1') \not\geq l$. By the induction hypothesis, such a formula exists. (Note that both $\neg[\![M_2,\varphi_{q_2'}]\!](q_2') \not\geq l$ and $[\![M_1,\varphi_{q_2'}]\!](q_1') \not\geq l$.) Let $A \subseteq Q_2$ be the set of states $q_2'$ for which $R_2(q_2,q_2') \geq l$, and let $\varphi' = \bigwedge_{q_2' \in A} \varphi_{q_2'}$ be the meet of the formulas $\varphi_{q_2'}$ for $q_2' \in A$.

We show that $AX\varphi'$ satisfies our claim; that is, $[\![M_2,AX\varphi']\!](q_2) \rightarrow [\![M_1,AX\varphi']\!](q_1) \not\geq l$. To see this, let us decompose the latter.

$$
\begin{aligned}
&[\![M_2,AX\varphi']\!](q_2) \rightarrow [\![M_1,AX\varphi']\!](q_1) = \\
&\quad = \neg[\![M_2,AX\varphi']\!](q_2) \vee [\![M_1,AX\varphi']\!](q_1) \\
&\quad = [\neg \bigwedge_{s' \in Q}(R_2(q_2,s') \rightarrow [\![M_2,\varphi']\!](s'))] \vee \bigwedge_{s' \in Q}(R_1(q_1,s') \rightarrow [\![M_1,\varphi']\!](s')) \\
&\quad = [\neg \bigwedge_{s' \in Q}(\neg R_2(q_2,s') \vee [\![M_2,\varphi']\!](s'))] \vee \bigwedge_{s' \in Q}(\neg R_1(q_1,s') \vee [\![M_1,\varphi']\!](s')) \\
&\quad = [\bigvee_{s' \in Q}(R_2(q_2,s') \wedge \neg[\![M_2,\varphi']\!](s'))] \vee \bigwedge_{s' \in Q}(\neg R_1(q_1,s') \vee [\![M_1,\varphi']\!](s'))
\end{aligned}
$$

$$\leq \quad [\bigvee_{q_2' \in Q} (R_2(q_2, q_2') \wedge \neg[\![M_2, \varphi']\!](q_2'))] \vee (\neg R_1(q_1, q_1') \vee [\![M_1, \varphi']\!](q_1')).$$

The last expression is a join of several expressions none of which is greater than $l$: First, for every $q_2'$ (alternatively called $s'$), we have $R_2(q_2, q_2') \wedge \neg[\![M_2, \varphi']\!](q_2') \not\geq l$. Next, we have $\neg R_1(q_1, q_1') \not\geq l$ and $[\![M_1, \varphi']\!](q_1')) \not\geq l$. Therefore, from join irreducibility of $l$ we get that the join of all of the above is not greater or equal to $l$ either. □

For two Lattice Kripke Structures $M_1 = \langle \mathcal{L}, AP, Q_1, Q_0^1, R_1, \Theta_1 \rangle$ and $M_2 = \langle \mathcal{L}, AP, Q_2, Q_0^2, R_2, \Theta_2 \rangle$, we define the *simulation value* of $M_1$ by $M_2$ to be

$$S^*(M_1, M_2) = \bigwedge_{q_1 \in Q_1} \left( Q_0^1(q_1) \rightarrow ( \bigvee_{q_2 \in Q_2} (Q_0^2(q_2) \wedge S^*(q_1, q_2))) \right),$$

where $S^*$ is the maximal simulation relation of the two structures.

Theorem 6 below gives the full logical characterization of latticed simulation. It follows from Theorems 3 and 5, and the distributivity of the lattice.

**Theorem 6** *Let $M_1$ and $M_2$ be two Kripke structures.*

1. *For all ALMC sentences $\psi$, we have $S^*(M_1, M_2) \leq [\![M_2, \psi]\!] \rightarrow [\![M_1, \psi]\!]$.*
2. *For all $l \in \mathcal{L}$, if $S^*(M_1, M_2) \not\geq l$, then there exists an ALML formula $\psi$ such that $[\![M_2, \psi]\!] \rightarrow [\![M_1, \psi]\!] \not\geq l$.*

In [23], we defined the implication value between latticed automata. The definition extends to latticed Kripke structures: for two latticed Kripke structures $M_1$ and $M_2$ over a lattice $\mathcal{L}$, let $imp\_val(M_1, M_2)$ be the implication value of $M_2$ by $M_1$. Essentially (for details, see [23]), each word $w \in (2^{AP})^\omega$ has a "membership value" in $M_1$ and in $M_2$, and $imp\_val(M_1, M_2)$ denotes the truth value of the statement "for all words, the membership value in $M_1$ implies the membership value in $M_2$". As in the Boolean case, the branching setting is more general than the linear setting. Formally, we have the following.

**Theorem 7** *For all latticed Kripke structures $M_1$ and $M_2$, we have $S^*(M_1, M_2) \leq imp\_val(M_1, M_2)$.*

Thus, latticed simulation, which can be calculated in polynomial time, is a lower bound to the implication value, whose calculation is PSPACE-complete.

### *3.1. Latticed Bisimulation*

The Boolean simulation pre-order has a symmetric version, namely the bisimulation relation. Two Kripke structures that are bisimilar have exactly the same behaviors. Adding symmetry to our definition of latticed simulation results in a *latticed bisimulation* relation. Formally, for two lattice kripke structures $M_1 = \langle \mathcal{L}, AP, Q_1, Q_0^1, R_1, \Theta_1 \rangle$ and $M_2 = \langle \mathcal{L}, AP, Q_2, Q_0^2, R_2, \Theta_2 \rangle$, an $\mathcal{L}$-relation $S : Q_1 \times Q_2 \rightarrow \mathcal{L}$ is an *$\mathcal{L}$-bisimulation* between $M_1$ and $M_2$ if

$$S(q_1, q_2) = S_{AP}(q_1, q_2) \wedge S_R(q_1, q_2) \wedge S_R(q_2, q_1),$$

where $S_{AP}$ and $S_R$ are as in $\mathcal{L}$-simulation.

The logical characterization of $\mathcal{L}$-simulation extends $\mathcal{L}$-bisimulation, now with LMC.

**Theorem 8** *Let $S$ be the maximal $\mathcal{L}$-bisimulation relation between $M_1$ and $M_2$.*

1. *For all LMC sentences $\varphi$, it holds that $S(q_1, q_2) \leq [\![M_1, \varphi]\!](q_1) \leftrightarrow [\![M_2, \varphi]\!](q_2)$.*
2. *For all $l \in \mathcal{L}$, if $S(M_1, M_2) \not\geq l$, then there exists an LML formula $\psi$ such that $[\![M_2, \psi]\!] \leftrightarrow [\![M_1, \psi]\!] \not\geq l$.*

**Proof:** As the definition of bisimulation is symmetric, it is enough to prove $S(q_1, q_2) \leq [\![M_1, \varphi]\!](q_1) \rightarrow [\![M_2, \varphi]\!](q_2)$. The proof proceeds by induction on the formula structure as in the proof of Theorem 3. In fact, in the proof of Theorem 3 we already saw proofs for the base case, and for the inductive step for all the $\mu$-calculus connectives except $EX$. This case is dual to the $AX$ case, and proceeds as follows. Let $\psi = EX\varphi$. By definition, $S(q_1, q_2) \leq \bigwedge_{q_1'}(\neg R_1(q_1, q_1') \vee \bigvee_{q_2'}(R_2(q_2, q_2') \wedge S(q_1', q_2')))$.

By the induction hypothesis we have:

$$\begin{array}{rcl} S(q_1, q_2) & \leq & \bigwedge_{q_1'}(\neg R_1(q_1, q_1') \vee \bigvee_{q_2'}(R_2(q_2, q_2') \wedge (\neg [\![M_1, \varphi]\!](q_1') \vee [\![M_2, \varphi]\!](q_2')))) \\ & \leq & \bigwedge_{q_1'}(\neg R_1(q_1, q_1') \vee \bigvee_{q_2'}(\neg [\![M_1, \varphi]\!](q_1') \vee (R_2(q_2, q_2') \wedge [\![M_2, \varphi]\!](q_2')))) \\ & = & \bigwedge_{q_1'}((\neg R_1(q_1, q_1') \vee \neg [\![M_1, \varphi]\!](q_1')) \vee \bigvee_{q_2'}(R_2(q_2, q_2') \wedge [\![M_2, \varphi]\!](q_2'))) \\ & = & \neg(\bigvee_{q_1'} R_1(q_1, q_1') \wedge [\![M_1, \varphi]\!](q_1')) \vee (\bigvee_{q_2'}(R_2(q_2, q_2') \wedge [\![M_2, \varphi]\!](q_2'))) \\ & = & \neg [\![M_1, EX\varphi]\!](q_1) \vee [\![M_2, EX\varphi]\!](q_2) \end{array}$$

□

Other properties of latticed simulation extend easily to latticed bisimulation: it can be computed in polynomial time, and the bisimulation value between two latticed Kripke structures is a lower bound to the equivalence value (two-sided implication) between them.

## 4. Latticed Games

A *latticed game graph* is a pair $G = \langle V, E \rangle$, where $V$ is a set of vertices and $E : V \times V \rightarrow \mathcal{L}$ is an $\mathcal{L}$-set of transitions. The vertices are partitioned into two sets, $V_\vee$ and $V_\wedge$ (referred to as the $\vee$-vertices and the $\wedge$-vertices). A *latticed game* is a latticed game graph together with an initial state $v_0 \in V$ and a acceptance condition $\alpha$. We postpone the description of the acceptance condition since for that we need the notion of a play that we define next.

Intuitively, a play of the game proceeds as follows: a token is put on some initial vertex. If the token is placed on a $\vee$-vertex then the $\vee$-player chooses an edge originating at the vertex on which the token is on, and the token is advanced along that edge. Similarly, if the token is placed on a $\wedge$-vertex, then the $\wedge$-player is doing the choosing. After the token is advanced to the successor vertex, the process repeats. This proceeds forever and the *play* of the game is a sequence of vertices $p = \{v_i\}_{i=0}^{\infty}$ (the sequence of vertices the token has traversed during the play). Each play is assigned a value, in a fashion we will describe next. Intuitively, the $\vee$-player's objective is to maximize the value of the play, while the $\wedge$-player's

objective is to minimize it. We proceed to define the value of a play formally. Note that the value of a play is defined from the $\vee$-player perspective, and is in fact the value of the game for the $\vee$-player. We postpone the definition of the value of the game for the $\wedge$-player.

The acceptance value of the play, denoted $acc(p)$, stands for the value with which the play satisfies the acceptance condition. For example, an $\mathcal{L}$-Büchi acceptance condition is an $\mathcal{L}$-set of states $F \in \mathcal{L}^V$, and the acceptance value of the play $p$ is $\bigwedge_{i=0}^{\infty} \bigvee_{j>i} F(v_i)$.

The value of a play is set not only by its acceptance value, but also by the values of the edges traversed during the play. Intuitively, when a player traverses an edge with low value he gives up more then if would have traversed an edge with a higher value. Assume, for example, that the underlying lattice is $2^{\{a,b,c\}}$ and in the first move the $\vee$-player traversed an edge with value $\{a\}$. This means that the $\vee$-player gives up all values that are not smaller or equal to $\{a\}$, and "is willing" that at the end of the play the acceptance value would be met with $\{a\}$. By dual reasoning, if the first move is done by the $\wedge$-player over an edge with value $\{a\}$, then the $\wedge$-player (whose objective is to lower the total play value) "is willing" that at the end of the game the value would be joined with $\neg\{a\} = \{b, c\}$. Furthermore, the order in which edges are traversed is important: if one player already gave up some value $l$, the other player is assured the value $l$, and may move freely along edges that would have implied giving up $l$ on other circumstances.

We therefore define two "given up" values, $r^\vee$ and $r^\wedge$. These values are defined inductively along the play, where at the beginning neither player has given up anything, thus $r_0^\vee = \top$ and $r_0^\wedge = \bot$. If $p_i \in V_\vee$, then the next transition is taken by the $\vee$-player. If the $\vee$-player chooses to traverse an edge with value different than $\top$, then he gives up the value of the edge traversed, except the parts of the value already given up by the $\wedge$-player. Thus, $r_{i+1}^\vee = r_i^\vee \wedge (E(p_i, p_{i+1}) \vee r_i^\wedge)$. The $\wedge$-player, on the other hand, gives up nothing, and therefore $r_{i+1}^\wedge = r_i^\wedge$. Similarly, if $p_i \in V_\wedge$, then $r_{i+1}^\vee = r_i^\vee$, and $r_{i+1}^\wedge = r_i^\wedge \vee (\neg E(p_i, p_{i+1}) \wedge r_i^\vee)$. Since both $\{r_i^\vee\}$ and $\{r_i^\wedge\}$ are monotonic, their limit is defined. Let $r^\vee = \bigwedge_{i=0}^{\infty} r_i^\vee$ and $r^\wedge = \bigvee_{i=0}^{\infty} r_i^\wedge$

To define the value of a play we need one more technical observation. Let $val^{\vee\wedge}(p) = (acc(p) \wedge r^\vee) \vee r^\wedge$, and $val^{\wedge\vee}(p) = (acc(p) \vee r^\wedge) \wedge r^\vee$. Similarly, let $val_i^{\vee\wedge}(p) = (acc(p) \wedge r_i^\vee) \vee r_i^\wedge$, and $val_i^{\wedge\vee}(p) = (acc(p) \vee r_i^\wedge) \wedge r_i^\vee$. We will shortly prove that $val^{\wedge\vee}(p) = val^{\vee\wedge}(p)$ and define the *value* of the play, denoted $val(p)$, to equal both.

**Example 9** Consider the game $G$ over the lattice $2^{\{a,b,c\}}$ appearing in Figure 3. Assume that all the computations of $G$ are accepting with value $\{a, b, c\}$. We calculate the value of some plays in $G$.

- Consider the play $p = (v_0, v_0, v_1)^\omega$. By definition, $r_0^\vee = \{a, b, c\}$ and $r_0^\wedge = \emptyset$. Since the first transition is by the $\vee$-player, we have, $r_1^\vee = r_0^\vee \wedge (E(v_0, v_0) \vee r_0^\wedge) = \{a, b, c\} \wedge (\{a, c\} \vee \emptyset) = \{a, c\}$. Also, $r_1^\wedge = r_0^\wedge = \emptyset$. The second transition is also by the $\vee$-player, thus $r_2^\vee = r_1^\vee \wedge (E(v_0, v_1) \vee r_1^\wedge) = \{a, c\} \wedge (\{a, b\} \vee \emptyset) = \{a\}$. Also, $r_2^\wedge = r_1^\wedge = \emptyset$. The third transition is by the $\wedge$-player, thus $r_3^\wedge = r_2^\wedge \vee (\neg E(v_1, v_0) \wedge r_2^\vee) = \emptyset \vee (\{a\} \wedge \{a\}) = \{a\}$. Also, $r_3^\vee = r_2^\vee = \{a\}$. At

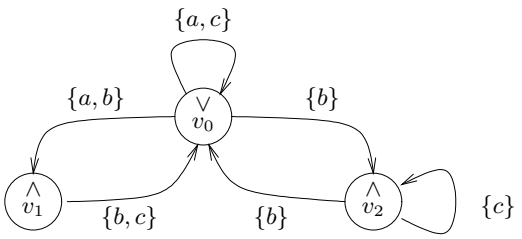


Figure 3: A game over the lattice $2^{\{a,b,c\}}$.

this point, the sequences $r_i^\vee$ and $r_i^\wedge$ reach a fixed-point, thus $r^\vee = r^\wedge = \{a\}$. Hence, $val^{\vee\wedge}(p) = val^{\wedge\vee}(p) = \{a\}$.

- Consider now the play $p = v_0^\omega$. Here, $r^\vee = \{a, c\}$ and $r^\wedge = \emptyset$. Accordingly, $val^{\vee\wedge}(p) = (\{a, b, c\} \wedge \{a, c\}) \vee \emptyset = \{a, c\}$, which equals $val^{\wedge\vee}(p) = (\{a, b, c\} \vee \emptyset) \wedge \{a, c\} = \{a, c\}$.
- Consider now the play $p = (v_0, v_2)^\omega$. Here, $r^\vee = \{b\}$ and $r^\wedge = \emptyset$. Accordingly, $val^{\vee\wedge}(p) = val^{\wedge\vee}(p) = \{b\}$.

□

Another definition of the value of a play was introduced by [29]. In [29], the authors define the value of a finite prefix of the play $p_0, \ldots, p_i$ in a backward manner. First, $val_i^i(p) = acc(p)$. Then, for $j \leq i$, we have $val_{j-1}^i(p) = val_j^i(p) \wedge E(p_{j-1}, p_j)$ if $p_{j-1}$ is a $\vee$-vertex, or $val_{j-1}^i = val_j^i \vee \neg E(p_{j-1}, p_j)$ if $p_{j-1}$ is a $\wedge$-vertex. The value of the prefix $p_0, \ldots, p_i$ is then $val^i = val_0^i$. It can be shown that the sequence $\{val^i\}_{i=0}^\infty$ stabilizes, and the value of the play is taken to be the limit. Thus, for the entire play, the value is calculated in a zig-zagged manner: in each iteration, a vertex is added, and then the calculation proceeds backwardly. Our definition, on the other hand, involves a forward traversal, and is similar to the definition in the Boolean case. As the next lemma shows, the intermediate values we get in our forward traversal coincide with these that [29] gets in the zig-zagged traversal.

**Lemma 1** *For every play $p = p_0 p_1 \ldots$, and every $i \geq 0$, we have $val_i^{\vee\wedge}(p) = val_i^{\wedge\vee}(p) = val^i$.*

**Proof:** It is enough to prove that for every join irreducible element $l \in \mathcal{L}$, it holds that $val_i^{\vee\wedge} \geq l$ iff $val_i^{\wedge\vee} \geq l$ iff $val^i \geq l$. Assume first that $val^i \geq l$. Denote by $k$ the largest index such that for all $j \leq k$ it holds that $val_j^i \geq l$. Since whenever a $\vee$-vertex $p_j$ is visits the value $val_j^i$ is met with $E(p_j, p_{j+1})$, we are assured that for all $j < k$, all the edges traversed by the $\vee$-player have value greater or equal to $l$. Therefore, for all $j \leq k$, we have $r_j^\vee \geq l$.

In addition, either $k = i$ in which case $acc(p) \geq l$, or $p_{k-1}$ is a $\wedge$-vertex and $\neg E(p_{k-1}, p_{k+1}) \geq l$. We show that $r_i^\vee \geq l$ and therefore both $val_i^{\vee\wedge} \geq l$ and $val_i^{\wedge\vee} \geq l$. If $k = i$ then $r_i^\vee \geq l$ and there is nothing to prove. Otherwise, $k < i$ and we have to show that for $j > k$, we have that $r_j^\vee \geq l$. However, $r_k^\vee \geq l$, and for

$j > k$ either $r^{\vee}_{j+1} = r^{\vee}_j$, or $r^{\vee}_{j+1} = r^{\vee}_j \wedge (E(p_j, p_{j+1}) \vee r^{\wedge}_j)$. Since $r^{\wedge}_k \geq l$ and $\{r^{\wedge}_j\}$ is monotonically non-decreasing, we get that for all $j > k$, we have $r^{\vee}_j \geq l$.

Assume now that either $r^{\wedge\vee} \geq l$ or $r^{\wedge\vee} \geq l$. Then, either $r^{\wedge}_j \geq l$ for some index $j$, or $acc(p) \geq l$. If for some index $j$ we have $r^{\wedge}_j \geq l$, let $k$ be the minimal index for which $r^{\wedge}_k \geq l$. Clearly $p_{k-1}$ is a $\wedge$-vertex and $\neg E(p_{k-1}, p_k) \geq l$. In addition, $r^{\vee}_{k-1} \geq l$ and by the monotonicity of $\{r^{\vee}_j\}$ we get that for every $j < k$ it holds that $r^{\vee}_j \geq l$. Therefore, for all $j < k$ we have $val^j \geq l$ as needed. We are left with the case in which for all indices $j$ it holds that $r^{\wedge}_j \not\geq l$. In that case, since either $r^{\wedge\vee} \geq l$ or $r^{\wedge\vee} \geq l$, it must be that both $acc(p) \geq l$ and $r^{\vee} \geq l$. Therefore, by monotonicity of $\{r^{\vee}_j\}$, we have $r^{\vee}_j \geq l$ for all $j$ and therefore $val^j \geq l$. □

Since both $\{r^{\vee}_j\}^{\infty}_{j=1}$ and $\{r^{\wedge}_j\}^{\infty}_{j=1}$ are monotone, they must stabilize eventually, implying an equivalence among the three values:

**Corollary 1** $val^{\wedge\vee}(p) = val^{\vee\wedge}(p) = \lim val^i$.

We define the value of a play $p$ to be $val^{\wedge\vee}(p)$ and denote it by $val(p)$. We also define the *value of a play p for the* $\wedge$*-player*, denoted $val_{\wedge}(p)$, as the negation of the value of the game for the $\vee$-player, i.e., $\neg val(p)$.

A *strategy* for a player is a function from prefixes of plays ending in one of his vertices, to the set of vertices. Thus, a $\vee$-player strategy is $f : V^* \cdot V_{\vee} \to V$. A prefix $p_0, \ldots, p_n$ is consistent with a strategy $f$ of the $\vee$-player, if for all $j \geq 0$ it holds that if $p_j$ is a $\vee$-vertex then $p_{j+1} = f(p_0, \ldots, p_j)$. Similarly, a strategy for the $\wedge$-player is a function $f : V^* \cdot V_{\wedge} \to V$, and a prefix $p_0, \ldots, p_n$ is consistent $f$, if for all $j \geq 0$ it holds that if $p_j$ is a $\wedge$-vertex then $p_{j+1} = f(p_0, \ldots, p_j)$. A play is consistent with a strategy if all its prefixes are consistent the strategy. It is easy to see that for every two strategies, one for the $\vee$-player and one for the $\wedge$-player, there is exactly one play consistent with both strategies. Thus, two strategies induce a play.

The *value of a strategy* $f$, denoted $val(f)$, is the meet of all the plays compatible with $f$ (this holds for strategies of either player). The *value of a game* for a player is the join of the values of that player's strategies. Thus, the value of a game $G$ for the $\vee$-player, denoted $val_{\vee}(G)$, is the join of all values of strategies of the $\vee$-player. Similarly, the value of a game for the $\wedge$-player, denoted $val_{\wedge}(G)$, is the join of all values of strategies of the $\wedge$-player. In Theorem 13 we prove that $val_{\wedge}(G) = \neg val_{\vee}(G)$.

**Example 10** Consider again the game $G$ discussed in Example 9. Consider a strategy $f$ for the $\vee$-player that whenever the play is in $v_0$, goes to $v_2$. There are infinitely many plays that are compatible of this strategy (depending on the move of the $\wedge$-player whenever the play is in $v_2$). All these plays, however, have value $\{b\}$. Thus, $val(f) = \{b\}$. Consider now a strategy $f'$ the $\vee$-player that whenever the play is in $v_0$, goes to $v_0$. Only the play $v_0^{\omega}$ is compatible with $f$. The value of this play is $\{a, c\}$, Thus, $val(f') = \{a, c\}$. It follows that $val_{\vee}(G) = \{a, b, c\}$. □

**Remark 11** Unlike the Boolean case, the $\vee$-player might not have a single strategy ensuring him the value of the game. In fact, it might be the case that the value of the game cannot be obtained as the value of a single play. As an example, consider the game $G$ described in Figure 3. In this game, over the lattice $2^{\{a,b,c\}}$, all the

vertices are $\vee$-vertices, the initial vertex is $v_0$, and all vertices have acceptance value $\{a, b, c\}$ (edges that are not in drawn have value $\emptyset$). It is not hard to see that the value of the game is $\{a, c\}$ although every single play has either value $\{a\}$ or value $\{c\}$. On the other hand, when the acceptance condition is one for which memoryless strategies suffice (in particular, in Büchi and parity games), the value of the game for the $\vee$-player can be obtained by following memoryless strategies. $\square$

*4.1. Properties of lattice games*

In the latticed case, unlike the Boolean case, it is not necessarily true that a game value can be obtained in a single play. Therefore, it does not make sense to search for a single strategy as a solution to the game. What is true, is that for each join irreducible element $l \in JI(\mathcal{L})$ it holds that if the game value is greater than or equal to $l$, then there exists a single strategy that ensures that a value of at least $l$, is obtained. Thus, Birkhoff's representation theorem enables us to decompose a latticed game to several Boolean games. Formally, we have the following.

**Theorem 12** *For a latticed game $G = \langle V, E\rangle$, over a lattice $\mathcal{L}$, there exists a family of Boolean games $\{G_l = \langle V, E_l\rangle\}_{l \in JI(\mathcal{L})}$ all sharing the same state space, such that the $\vee$-player has a winning strategy in $G_l$ iff there is a strategy in $G$ ensuring the $\vee$-player a value greater than or equal to $l$.*

*Furthermore, for every $l \in JI(\mathcal{L})$, the game $G_l$ can be computed in logarithmic space from $G$. In addition, the strategy ensuring a value greater than or equal to $l$ can be computed, in logarithmic space, from a winning strategy in $G_l$ and vice versa.*

**Proof:** The game $G_l = \langle V, E_l\rangle$ has the same set of vertices as $G$, with the same partition to $\vee$- and $\wedge$-vertices. There exists an edge $\langle v, u\rangle$ in $G_l$ iff either $v \in V_\vee$ and the edge $\langle v, u\rangle$ has value greater than or equal to $l$ in $G$, or $v \in V_\wedge$ and the edge $\langle v, u\rangle$ does not have value less than or equal to $\neg l$ in $G$. Formally, $E_l = \{(u, v) \in E \cap (V_\vee \times V) \mid E(\langle v, u\rangle) \geq l\} \cup \{(u, v) \in E \cap (V_\wedge \times V) \mid E(\langle v, u\rangle) \not\leq \neg l\}$. A state is accepting in $G_l$ iff its acceptance value in $G$ is greater than or equal to $l$.

Since all the games share the same state space, it is not hard to see that every a prefix of a play in $G_l$ is also a prefix of a play in $G$. Note that a prefix of a play in $G$ is also a prefix of a play in $G_l$ iff no player has given up $l$ yet. Formally, $v_1, \ldots, v_n$ is a prefix of a play in $G_l$ iff $r_n^\vee \geq l$ and $r_n^\wedge \not\geq l$.

Let $f_l$ be a winning strategy for the $\vee$-player in the Boolean game $G_l$. We prove that if the $\vee$-player plays according to $f_l$ in $G$ (as long as it can) then the $\vee$-player is assured that the value of the play would be greater than or equal to $l$. The main observation is that as long as no player gives up $l$, the (prefix of the) play in $G$ is a (prefix of a) play in $G_l$, and therefore the $\vee$ player can follow $f_l$ without giving up $l$ (as $l$ is winning in $G_l$). If, on the other hand, the $\wedge$-player does give up $l$ at some stage, then the $\vee$-player is ensured a that the value of the play would be greater than or equal to $l$ regardless of his future actions.

In case the $\wedge$-player never gives up $l$, then the $\vee$-player continues to play according to $f_l$ and the resulting play can be viewed as a play in $G_l$ compatible with $f_l$. Since $f_l$ is a winning strategy for the $\vee$-player in $G_l$, such a play is sure to visit

infinitely often accepting states of $G_l$ i.e. states with acceptance value greater than or equal to $l$ in $G$. Therefore, the acceptance value of the play in $G$ is greater than or equal to $l$.

As for the other direction, assume the $\vee$-player has a strategy $f$ ensuring him a value greater than or equal to $l$ in $G$. Clearly, if the $\vee$-player follows $f$, then if the $\wedge$-player has not given up $l$ neither does the $\vee$-player. Thus, the strategy can be used as a strategy of the game $G_l$. Furthermore, since the strategy ensures a value greater than or equal to $l$ in $G$, every play in $G_l$ that is compatible with $f$ must be accepting in $G_l$. □

Theorem 12 suggests a way to solve a latticed game by decomposing it into Boolean games and solving each of the Boolean games. A different algorithm is suggested in [29].

Recall that Boolean Büchi games are determined: in every game, one of the players has a winning strategy. Extending this result to the latticed setting amounts to proving that for every value $l$, if the value of the game for the $\vee$-player is greater than $l$, then $\neg l$ is greater than the value of the game for the $\wedge$-player.

**Theorem 13** *For a lattice game $G$, we have $val_\wedge(G) = \neg val_\vee(G)$.*

**Proof:** We denote by $\mathcal{S}^\wedge$ the set of $\wedge$-player strategies. For a strategy $S$ we denote by $c(S)$ the set of plays compatible with $S$. By definition,

$$val_\wedge(G) = \bigvee_{S\in\mathcal{S}^\wedge} val_\wedge(S) = \bigvee_{S\in\mathcal{S}^\wedge} \bigwedge_{p\in c(S)} val_\wedge(p) = \bigvee_{S\in\mathcal{S}^\wedge} \bigwedge_{p\in c(S)} \neg val_\vee(p).$$

Let $l$ be a join irreducible value (note that $\neg l$ is a meet irreducible value). If $val_\vee(G) \geq l$, then the $\vee$-player can ensure a value of at least $l$ by playing according to the winning strategy $f_l$ in the game $G_l$. Therefore, for every meet player strategy $S \in \mathcal{S}^\wedge$ and play $p$, if $p$ is compatible with both $f_l$ and $S$ then $val_\vee(p) \geq l$. Equivalently, for $p \in c(f_l) \cap c(S)$ we have $\neg val_\vee(p) \leq \neg l$. Therefore, for every $S \in \mathcal{S}^\wedge$, we have $\bigwedge_{p\in c(S)} \neg val_\vee(p) \leq \neg l$, and $\bigvee_{S\in\mathcal{S}^\wedge} \bigwedge_{p\in c(S)} \neg val_\vee(p) \leq \neg l$. Thus, if $val_\vee(G) \geq l$, then $val_\wedge(G) \leq \neg l$.

If, on the other hand, $val_\vee(G) \not\geq l$, then the $\vee$-player has no winning strategy in $G_l$. As $G_l$ is determined [25], this implies that the $\wedge$-player has a winning strategy in $G_l$, we denote it by $S_{\bar{l}}$. Just as in the proof of Theorem 12, for every play $p$ compatible with $S_{\bar{l}}$ it holds that $val_\vee(p) \not\geq l$. Equivalently, for $p \in c(S_{\bar{l}})$, it holds that $\neg val_\vee(p) \not\leq \neg l$. Therefore, by the meet irreducibility of $\neg l$, we have $\bigwedge_{p\in c(S_{\bar{l}})} \neg val_\vee(p) \not\leq \neg l$, and $val_\wedge(G) = \bigvee_{S\in\mathcal{S}^\wedge} \bigwedge_{p\in c(S)} \neg val_\vee(p) \not\leq \neg l$.

Thus, for every join irreducible element $l$, we get $val_\vee(G) \geq l$ iff $val_\wedge(G) \leq \neg l$, which implies $val_\vee(G) \geq l$ iff $\neg val_\wedge(G) \geq l$. By Birkhoff's representation theorem we get $val_\vee(G) = \neg val_\wedge(G)$. □

*4.2. The simulation game*

For two latticed Kripke structures $M_1 = \langle \mathcal{L}, AP, Q_1, Q_0^1, R_1, \Theta_1\rangle$ and $M_2 = \langle \mathcal{L}, AP,$
$Q_2, Q_0^2, R_2, \Theta_2\rangle$, the *simulation game for $M_1$ and $M_2$* is a latticed game defined

as follows. Intuitively, the $\wedge$-player "claims" that $M_1$ is not simulated by $M_2$, while the $\vee$-player "claims" that $M_1$ is simulated by $M_2$. The game value is then $S^*(M_1, M_2)$. Thus, in the beginning of the game, the $\wedge$-player chooses an initial state $q_1$ in $M_1$, and the $\vee$-player chooses an initial state $q_2$ in $M_2$ that is supposed to resemble $q_1$. We measure the resemblance value between $q_1$ and $q_2$ by $S_{AP}(q_1, q_2) = \bigwedge_{p \in AP}([\![M_1, p]\!](q_1) \leftrightarrow [\![M_2, p]\!](q_2))$. The game proceeds by the $\wedge$-player choosing a successor to $q_1$, denoted $q_1'$, followed by the $\vee$-player choosing a successor to $q_2$, denoted $q_2'$. Again, $q_2'$ is supposed to resemble $q_1'$. This process is iterated ad infinitum.

Naturally, the edges values correspond to the transitions taken, thus the edge chosen by the $\wedge$-player to move from $q_1$ to $q_1'$ has the value of the transition $R_1(q_1, q_1')$. The values of the $\vee$-player transitions reflect not only the value of the corresponding transition in $M_2$, but also the resemblance between the state in $M_1$ to the state in $M_2$. Therefore, the value of the $\vee$-player transition is the value of the transition in $M_2$ meet the value $S_{AP}(q_1', q_2')$. We now to define the game formally.

The game graph is $G_{\langle M_1, M_2 \rangle} = \langle V, E \rangle$, where $V = (Q_1 \times Q_2 \times \{\wedge, \vee\}) \cup \{in_\wedge\} \cup (Q_1 \times \{in_\vee\})$. The $\wedge$-vertices are $(Q_1 \times Q_2 \times \{\wedge\}) \cup \{in_\wedge\}$, and the $\vee$-vertices are $(Q_1 \times Q_2 \times \{\vee\}) \cup (Q_1 \times \{in_\vee\})$. The initial position is $in_\wedge$, and the edges are defined as follows. For every $q_1 \in Q_1$, there exists an edge from $in_\wedge$ to $\langle q_1, in_\vee \rangle$ with value $Q_0^1(q_1)$. For every $q_1 \in Q_1$ and $q_2 \in Q_2$ there is an edge from $\langle q_1, in_\vee \rangle$ to $\langle q_1, q_2, \wedge \rangle$ with value $Q_0^2(q_2) \wedge S_{AP}(q_1, q_2)$. For $q_1, q_1' \in Q_1$ and $q_2, q_2' \in Q_2$, the edge from $\langle q_1, q_2, \wedge \rangle$ to $\langle q_1', q_2, \vee \rangle)$ has value $R_1(q_1, q_1')$, and the edge from $\langle q_1', q_2, \vee \rangle$ to $\langle q_1', q_2', \wedge \rangle)$ has value $R_2(q_2, q_2') \wedge S_{AP}(q_1', q_2')$. All other edges have value $\bot$. The acceptance criteria is Büchi in which all vertices are accepting with value $\top$ (making the acceptance value of every play $\top$).

We now claim that the value of the simulation game is the simulation value of $M_1$ by $M_2$.

**Theorem 14** $val_\vee(G_{\langle M_1, M_2 \rangle}) = S^*(M_1, M_2)$.

**Proof:** Recall that the simulation value is $S^*(M_1, M_2) = \bigwedge_{q_1 \in Q_1} Q_0^1(q_1) \rightarrow (\bigvee_{q_2 \in Q_2} (Q_0^2(q_2) \wedge S^*(q_1, q_2)))$. By Birkhoff's representation theorem, it is enough to prove that for every join irreducible element $l \in \mathcal{L}$, the value of the simulation game is greater than or equal to $l$ iff $S^*(M_1, M_2)$ is greater than or equal to $l$. For the rest of the proof we fix such a join irreducible element $l$.

Assume first that $S^*(M_1, M_2) \geq l$. We describe a strategy for the $\vee$-player that ensures that the value of every play is greater than or equal to $l$. Since the acceptance value of a play is always $\top$, we only have to find a strategy ensuring that the $\vee$-player does not give up $l$ unless the $\wedge$-player did that first.

When a play begins at $in_\wedge$, the $\wedge$-player chooses a vertex $q_1$ and moves to $\langle q_1, in_\vee \rangle$. Since $S^*(M_1, M_2) \geq l$, we know that $(\neg Q_0^1(q_1) \vee (\bigvee_{q_2 \in Q_2} (Q_0^2(q_2) \wedge S^*(q_1, q_2))) \geq l$. As $l$ is join irreducible, either $\neg Q_0^1(q_1) \geq l$, in which case the $\wedge$-player gives up $l$ first, or $\bigvee_{q_2 \in Q_2} (Q_0^2(q_2) \wedge S^*(q_1, q_2)) \geq l$, in which case there exists a state $q_2$ for which $Q_0^2(q_2) \wedge S^*(q_1, q_2) \geq l$ (again by join irreducibility). Thus, the strategy advises the $\vee$-player move to $\langle q_1, q_2, \wedge \rangle$. The value of the edge is greater than or equal to $l$, since both $Q_0^2(q_2) \geq l$ and $S^*(q_1, q_2) \geq l$, implying

that $S_{AP}(q_1, q_2) \geq l$. By similar reasoning (regarding transition values rather than initial values) we construct the rest of the strategy.

Assume now that $S^*(M_1, M_2) \not\geq l$. We present a strategy for the $\wedge$-player that ensures that the $\vee$-player has given up $l$ before the $\wedge$-player does. Note that while the value given up by the $\wedge$-player is accumulated as a join, $l$ is join irreducible and therefore if it is given up it must be given up some in specific transition.

By the definition of $S^*(M_1, M_2)$, there must exist a state $q_1$ for which $(\neg Q_0^1(q_1) \vee (\bigvee_{q_2 \in Q_2}(Q_0^2(q_2) \wedge S^*(q_1, q_2))) \not\geq l$. Recall the way $S^*$ is computed in Theorem 4. There exists a minimal number $k \geq 0$ such that $(\neg Q_0^1(q_1) \vee (\bigvee_{q_2 \in Q_2}(Q_0^2(q_2) \wedge S_k(q_1, q_2)))$
$\not\geq l$. In his first move, the $\wedge$-player moves into $\langle q_1, in_\vee \rangle$. Since $\neg Q_0^1(q_1) \not\geq l$, the $\wedge$-player does not give up $l$ in this move. The $\vee$-player responses by moving into some state $\langle q_1, q_2, \vee \rangle$. If in this move the $\vee$-player gives up $l$, we are done. Otherwise, we are assured that $S_k(q_1, q_2) \not\geq l$. We now show that the $\vee$-player can force the game into a state $\langle q_1', q_2', \wedge \rangle$ such that $S_{k-1}(q_1', q_2') \not\geq l$ without giving up $l$. Recall that $S_k(q_1, q_1) = S_{k-1}(q_1, q_2) \wedge \bigwedge_{q_1' \in Q_1} \bigvee_{q_2' \in Q_2} R_1(q_1, q_1') \rightarrow (R_2(q_2, q_2') \wedge S_{k-1}(q_1', q_2'))$. Therefore, either $S_{k-1}(q_1, q_2) \not\geq l$ or by repeating the reasoning used when choosing the initial state (this time regarding transition values rather than initial values), the $\wedge$-player can force the game into a desired state (without giving up $l$). Clearly, within a finite number of moves the play is forced into some state $\langle q_1^*, q_2^*, \wedge \rangle$ such that $S_0(q_1^*, q_2^*) = S_{AP}(q_1^*, q_2^*) \not\geq l$ and from such a state the $\vee$-player must give up $l$. □

Thus, as in the Boolean setting, latticed simulation can be defined in terms of a game between two players.

## 5. Discussion

We lifted the notions of simulation and games to a multi-valued setting. We considered values taken from a lattice, and we were able to lift the known properties of simulation and games to the latticed setting. In the Boolean setting, bisimulation is an equivalence relation, and an abstraction of a system (one that agrees with the original system on all $\mu$-calculus specifications) can be obtained by merging bisimilar states to one state. In the latticed setting, bisimulation associates with each two states a lattice element denoting their bisimulation value. Therefore, even if we settle on a lattice element $l$ and seek an abstraction whose bisimulation value with the original system is $l$, it is not clear how to define the state space and the transitions of the abstraction. Finding a satisfying definition would enable us to find coarsest abstractions that may not agree with the original system on all specifications, but for which we can provide a lower bound on the value of agreement (i.e., most view-points agree).

Another open question is the extension of latticed simulation to Kripke structures with fairness. In the Boolean setting, the relation between simulation and games has led to a definition of fair simulation that retains the logical characterization and the computational advantages of simulation [19]. While the relation between simulation and games is maintained in the latticed setting, it is an open

question whether latticed games can be used in a definition of latticed fair simulation. Indeed, the definition in [19] relates a strategy that generates computations in the simulated structure with a strategy that generates computations in the simulating structure. In the latticed setting, the value of the game may depend on different strategies, thus a game-based definition of fair simulation has to take all strategies into an account.

## References


1. R. Alur, A. Kanade, and G. Weiss. Ranking automata and games for prioritized requirements. In *Proc 20th Int. Conf. on Computer Aided Verification*, volume 5123 of *Lecture Notes in Computer Science*. Springer, 2008.
2. S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. Property preserving simulations. In *Proc 4th Int. Conf. on Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 260–273. Springer, 1992.
3. Bruns and Godefroid. Model checking with multi-valued logics. In *Proc. 31st Int. Colloq. on Automata, Languages, and Programming*, volume 3142 of *Lecture Notes in Computer Science*, pages 281–293, 2004.
4. G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Proc 11th Int. Conf. on Computer Aided Verification*, pages 274–287, 1999.
5. G. Bruns and P. Godefroid. Temporal logic query checking. In *Proc. 16th IEEE Symp. on Logic in Computer Science*, pages 409–420. IEEE Computer Society, 2001.
6. W. Chan. Temporal-logic queries. In *Proc 12th Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 450–463. Springer, 2000.
7. K. Chatterjee, L. Doyen, and T. Henzinger. Quantative languages. 2008.
8. M. Chechik, B. Devereux, and S. Easterbrook. Implementing a multi-valued symbolic model checker. In *Proc. 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, number 2031 in Lecture Notes in Computer Science, pages 404–419. Springer, 2001.
9. M. Chechik, B. Devereux, and A. Gurfinkel. Model-checking infinite state-space systems with fine-grained abstractions using SPIN. In *Proc. 8th Int. SPIN Workshop on Model Checking Software*, volume 2057 of *Lecture Notes in Computer Science*, pages 16–36. Springer, 2001.
10. M. Chechik, S. Easterbrook, and V. Petrovykh. Model checking over multi-valued logics. In *Formal Methods Europe*, 2001.
11. R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transactions on Programming Languagues and Systems*, 15:36–72, 1993.
12. D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Transactions on Programming Languagues and Systems*, 19(2):253–291, 1997.
13. S. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *Proc. 23rd Int. Conf. on Software Engineering*, pages 411–420. IEEE Computer Society Press, 2001.
14. M.C. Fitting. Many-valued modal logics. *Fundamenta Informaticae*, XV:235–254, 1991.

15. P. Godefroid and R. Jagadeesan. Automatic abstraction using generalized model checking. In *Proc 14th Int. Conf. on Computer Aided Verification*, volume 2404, pages 137–150, 2002.
16. O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Transactions on Programming Languagues and Systems*, 16(3):843–871, 1994.
17. R. Hähnle. Automated deduction in multiple-valued logics. *International Series of Monographs on Computer Science*, 10, 1994.
18. M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. 36th IEEE Symp. on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.
19. T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.
20. M. Huth and S. Pradhan. Consistent partial model checking. *Electr. Notes Theor. Comput. Sci.*, 73:45–85, 2004.
21. IEEE. IEEE standard multivalue logic system for VHDL model interoperability (Std_logic_1164), 1993.
22. D. Kozen. Results on the propositional $\mu$-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
23. O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science*, pages 199 – 213. Springer, 2007.
24. K.G. Larsen and G.B. Thomsen. A modal process logic. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, 1988.
25. D.A. Martin. Borel determinacy. *Annals of Mathematics*, 65:363–371, 1975.
26. R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd Int. Joint Conf. on Artificial Intelligence*, pages 481–489. British Computer Society, 1971.
27. A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th Int. Colloq. on Automata, Languages, and Programming*, volume 194 of *Lecture Notes in Computer Science*, pages 15–32. Springer, 1985.
28. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Proc 9th Int. Conf. on Computer Aided Verification*, volume 1254, pages 72–83. Springer, 1997.
29. S. Shoham and O. Grumberg. Multi-valued model checking games. In *3rd Int. Symp. on Automated Technology for Verification and Analysis*, volume 3707, pages 354–369. Springer, 2005.
30. V. Sofronie-Stokkermans. Automated theorem proving by resolution for finitely-valued logics based on distributive lattices with operations. *Multiple-Valued Logics: An International Journal*, 5(2), 2000.
31. L.J. Stockmeyer and A.R. Meyer. Word problems requiring exponential time. In *Proc. 5th ACM Symp. on Theory of Computing*, pages 1–9, 1973.