Multi-Valued Model Checking Games

Sharon Shoham and Orna Grumberg

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

Abstract. This work extends the game-based framework of μ-calculus model checking to the *multi-valued* setting. In multi-valued model checking a formula is interpreted over a Kripke structure defined over a lattice. The value of the formula is also an element of the lattice. We define a new game for this problem and derive from it a *direct* model checking algorithm that handles the multi-valued structure without any reduction. We investigate the properties of the new game, both independently, and in comparison to the automata-based approach. We show that the usual resemblance between the two approaches does *not* hold in the multi-valued setting and show how it can be regained by changing the nature of the game.

1 Introduction

Model checking [8] is a successful approach for verifying whether a system model M satisfies a specification φ, written as a temporal logic formula. In multi-valued model checking the system is defined over a lattice $\mathcal{L}$. Both the labelling of states and the transitions of the system are interpreted as elements from the lattice. The meaning of a formula in the model is then also given by an element of the lattice.

Multi-valued model checking has many important applications within the verification framework. For example, 3-valued model checking, where the logic is based on the lattice L_3 (see Fig. 1), has been used to reason about abstract structures or structures with partial information [2, 24, 13]. In this context the value U is used to model *uncertainty*, with the meaning that the value can either be $\top$ or $\bot$. Recently, [1] has used a 6-valued logic as an extension of this approach for falsification of properties. There, the value U is refined to recognize that at least one concrete state falsifies the property or at least one concrete state satisfies the property. Another useful lattice is the lattice $L_{2,2}$, with the values $\top\bot$ and $\bot\top$ representing *disagreement* (see Fig. 1). Model checking using this lattice (or its generalizations) has been used to handle inconsistent views of a system [11, 17]. Temporal logic query checking [5, 3, 15] can also be reduced to multi-valued model checking, where the elements of the lattice are sets of propositional formulas.

One way of handling the multi-valued model checking problem is the *reduction* approach, where the problem is reduced to several traditional 2-valued problems [12, 17, 18, 14, 4] or 3-valued problems [19].

As opposed to the reduction approach, the *direct* approach checks the property on the multi-valued structure directly. It thus has the advantage of a more "on-the-fly" nature. Furthermore, a direct model checker can provide auxiliary information that explains its result [24, 13]. When using the reduction approach such information can only be gathered for each problem separately. It thus becomes less useful.

Several direct model checking algorithms for various multi-valued logics have been suggested in the literature. [2, 24, 13] studied the 3-valued case of CTL ([2, 24]) and the μ-calculus

Fig. 1. Examples of Lattices

([13]). In [6] the logic LTL was considered over finite linear orders. The general multi-valued version of CTL was handled in [7]. Finally, an *almost* direct automata-based algorithm for the multi-valued μ-calculus was suggested in [4]. Their approach handled the multi-valued labelling directly, but still used the reduction approach to handle multi-valued transitions.

In this paper we suggest a *fully direct* model checking for the multi-valued μ-calculus, where both the multi-valued labelling and the multi-valued transitions are handled directly. The *μ-calculus* [20] is a powerful formalism for expressing properties of transition systems using fixpoint operators. It contains, for example, both CTL and LTL as its fragments. Our approach refers to its multi-valued semantics based on any finite distributive DeMorgan lattice.

We base our algorithm on the *game-theoretic* approach [25] and thus gain all of its advantages [24, 13]. In the traditional game-based approach to model checking two players, the verifier (called $\exists$loise) and the refuter (called $\forall$belard), try to win a game. A formula φ is true in a model M iff the verifier has a winning strategy, meaning that the verifier can win any play, no matter what the refuter does.

We adapt this approach for the multi-valued case. In particular, we now talk about the *value* of the game. It turns out that in the multi-valued case there does *not* necessarily exist a *best* strategy for $\exists$loise. Instead, strategies may be incomparable and the value of the game is determined by their combination.

We suggest two definitions of a multi-valued game for the μ-calculus and prove their correctness. The proof turns out to be interesting in itself, as it uses similar techniques to those used in the reduction approach of [4]. This is in spite of the fact that our approach handles the multi-valued structure directly and uses no reductions.

When comparing our definitions to the work of [4], a surprising property is revealed. The direct algorithm of [4] is based on automata [21]. It is usually the case that the game-based and the automata-based approaches to model checking have a strong resemblance [22]. Yet, in the multi-valued case we find that our definition of the multi-valued game is different in essence from the automata-based approach of [4]. We discuss this difference and suggest an alternative multi-valued game that regains the similarity to automata. More importantly, our resulting framework in fact generalizes the work of [4], as it handles directly not only the multi-valued labelling, but also the multi-valued transitions.

The game-based approach to model checking was already generalized to the 3-valued case [24, 13]. However, it turns out that handling a general lattice, where there is more than one intermediate value and the elements are only partially ordered, is substantially more complex (see Section 7).

The rest of the paper is organized as follows. In Section 2 we give some background on lattice theory, multi-valued μ-calculus and model checking games. In Section 3 we provide our main definition of the multi-valued model checking game and prove its correctness. A model checking algorithm, based on the game, is then described in Section 4. In Section 5 we suggest an alternative definition for the game. We then discuss the relation to the automata-theoretic approach, which yields another definition of a multi-valued game, in Section 6. Finally, we compare the general multi-valued game to the much simpler 3-valued case in Section 7.

2 Preliminaries

Lattices A *lattice* is a partially ordered set $(\mathcal{L}, \leq)$ where for each finite subset of elements there exists a unique *greatest lower bound* (*glb*) and *least upper bound* (*lub*). The glb is also called *meet* and is denoted by $x \wedge y$ or $\bigwedge A$ (for $x, y \in \mathcal{L}$, $A \subseteq \mathcal{L}$). The lub is also called *join* and is denoted $x \vee y$ or $\bigvee A$.

Throughout this paper we refer to finite distributive DeMorgan lattices. Every finite lattice is *complete*, meaning that it has a greatest element, called *top*, denoted $\top$, and a least element, called *bottom*, denoted $\bot$. In a *distributive* lattice $x \wedge (y \vee z) = (z \wedge y) \vee (x \wedge y)$ and $x \vee (y \wedge z) = (z \vee y) \wedge (x \vee y)$ for all lattice elements x, y, z. In a *DeMorgan* lattice every element $x \in \mathcal{L}$ has a unique complement $\neg x \in \mathcal{L}$ such that $\neg\neg x = x$, DeMorgan's laws hold, and $x \leq y$ implies $\neg y \leq \neg x$ [1].

A *join-irreducible* element x of a distributive lattice $\mathcal{L}$ is an element $\neq \bot$ for which $x = y \vee z$ implies $x = y$ or $x = z$ for every $y, z \in \mathcal{L}$. We denote the set of join-irreducible elements of $\mathcal{L}$ by $\mathcal{J}(\mathcal{L})$. For example $\bot\top \in \mathcal{J}(L_{2,2})$, but $\top \notin \mathcal{J}(L_{2,2})$ (see Fig. 1).

μ-calculus [20] Let $\mathcal{P}$ be a finite set of atomic propositions and $\mathcal{V}$ a set of propositional variables. We consider the logic *μ-calculus* in *negation normal form*, defined as follows:

$$\varphi \ ::= \ q \ \mid \ \neg q \ \mid \ Z \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \Diamond\varphi \ \mid \ \Box\varphi \ \mid \ \mu Z.\varphi \ \mid \ \nu Z.\varphi$$

where $q \in \mathcal{P}$ and $Z \in \mathcal{V}$. Let $\mathcal{L}_\mu$ denote the set of *closed* formulas generated by the above grammar, where the fixpoint quantifiers μ and ν are variable binders. We will also write η for either μ or ν. Furthermore we assume that formulas are well-named, i.e. no variable is bound more than once in any formula. Thus, every variable Z *identifies* a unique subformula $fp(Z) = \eta Z.\psi$ of φ, where the set $Sub(\varphi)$ of *subformulas* of φ is defined in the usual way.

Semantics The *concrete semantics* of a μ-calculus formula is given with respect to a Kripke structure $\mathcal{M} = (\mathcal{S}, \mathcal{R}, \Theta)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is a transition relation, which must be *total*, and $\Theta : \mathcal{S} \to 2^{\mathcal{P}}$ is a labelling function [8].

In this work we consider the multi-valued μ-calculus [4], where formulas are interpreted with respect to a Kripke structure defined over a lattice (also called χKripke structure). In a Kripke structure over a lattice $\mathcal{L}$, both the labelling and the transition relation have a multi-valued nature: Θ maps a state to a mapping from $\mathcal{P}$ to elements of $\mathcal{L}$, that is $\Theta : \mathcal{S} \to (\mathcal{P} \to \mathcal{L})$. Furthermore, $\mathcal{R}$ maps pairs of states to lattice elements, that is $\mathcal{R} : \mathcal{S} \times \mathcal{S} \to \mathcal{L}$ (see Example 1). The totality requirement of $\mathcal{R}$ is now given by the requirement that for each $s \in \mathcal{S}$ there exists some state $s' \in \mathcal{S}$ with $\mathcal{R}(s, s') \neq \bot$.

The *semantics* $[\![\varphi]\!]^{\mathcal{M}}_{\rho}$ of a $\mathcal{L}_\mu$ formula φ w.r.t. a Kripke structure $\mathcal{M} = (\mathcal{S}, \mathcal{R}, \Theta)$ over a lattice $\mathcal{L}$ and an *environment* $\rho : \mathcal{V} \to (\mathcal{S} \to \mathcal{L})$, where ρ explains the meaning of free variables in φ, is a mapping from $\mathcal{S}$ to $\mathcal{L}$.

We assume $\mathcal{M}$ to be fixed and do not mention it explicitly anymore. With $\rho[Z \mapsto g]$ we denote the environment that maps Z to g and agrees with ρ on all other arguments. Later, when only closed formulas are considered, we will also drop the environment from the semantic brackets. In the following definition f is an element of $(\mathcal{S} \to \mathcal{L}) \to (\mathcal{S} \to \mathcal{L})$, defined by $\lambda g.[\![\varphi]\!]_{\rho[Z \mapsto g]}$ and νf, μf stand for the greatest and least fixpoints of f. According to [26], least and greatest fixpoints of this functional exist since the functions in $\mathcal{S} \to \mathcal{L}$ form a complete lattice under pointwise ordering and the functional f is monotone w.r.t. this ordering.

[1] Since we refer to temporal logic in negation normal form, negation can be defined arbitrarily. We chose to refer to DeMorgan lattices since they are most commonly used in this context.

$$\frac{s \vdash \varphi_0 \vee \varphi_1}{s \vdash \varphi_i}\ \exists:\ i \in \{0,1\} \qquad \frac{s \vdash \varphi_0 \wedge \varphi_1}{s \vdash \varphi_i}\ \forall:\ i \in \{0,1\}$$

$$\frac{s \vdash \eta Z.\varphi}{s \vdash Z}\ \exists/\forall \qquad \frac{s \vdash Z}{s \vdash \varphi}\ \exists/\forall : \text{if } fp(Z) = \eta Z.\varphi$$

$$\frac{s \vdash \Diamond \varphi}{t \vdash \varphi}\ \exists:\ \mathcal{R}(s,t) \neq \bot \qquad \frac{s \vdash \Box \varphi}{t \vdash \varphi}\ \forall:\ \mathcal{R}(s,t) \neq \bot$$

Fig. 2. The 2-valued model checking game rules for $\mathcal{L}_\mu$.

$$\begin{aligned}
[\![q]\!]_\rho &:= \lambda s.\Theta(s)(q) \\
[\![\neg q]\!]_\rho &:= \lambda s.\neg\Theta(s)(q) \\
[\![Z]\!]_\rho &:= \rho(Z) \\
[\![\varphi_1 \vee \varphi_2]\!]_\rho &:= \lambda s.[\![\varphi_1]\!]_\rho \vee [\![\varphi_2]\!]_\rho \\
[\![\varphi_1 \wedge \varphi_2]\!]_\rho &:= \lambda s.[\![\varphi_1]\!]_\rho \wedge [\![\varphi_2]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho &:= \lambda s.\bigvee\{\mathcal{R}(s,s') \wedge [\![\varphi]\!]_\rho(s') \mid \mathcal{R}(s,s') \neq \bot\} \\
[\![\Box\varphi]\!]_\rho &:= \lambda s.\bigwedge\{\neg\mathcal{R}(s,s') \vee [\![\varphi]\!]_\rho(s') \mid \mathcal{R}(s,s') \neq \bot\} \\
[\![\mu Z.\varphi]\!]_\rho &:= \mu f \\
[\![\nu Z.\varphi]\!]_\rho &:= \nu f
\end{aligned}$$

Given φ, $(\mathcal{M}, s)$ and $\mathcal{L}$, computing the value of $[\![\varphi]\!]^{\mathcal{M}}(s)$ is called the *multi-valued model checking problem*.

A regular Kripke structure $\mathcal{M}$ can be viewed as a Kripke structure over lattice L_2 (see Fig. 1), by referring to the set of transitions and the set of atomic propositions that label a state by their characteristic functions. In this case we write $(M, s) \models \varphi$ for $[\![\varphi]\!]^{\mathcal{M}}(s) = \top$ and $(M, s) \not\models \varphi$ for $[\![\varphi]\!]^{\mathcal{M}}(s) = \bot$.

Model Checking Games The *2-valued model checking game* $\Gamma_{\mathcal{M}}(s_0, \varphi_0)$ on a (regular) Kripke structure $\mathcal{M}$ with state set $\mathcal{S}$, state $s_0 \in \mathcal{S}$ and a μ-calculus formula φ_0 is played by players ∃loise (the prover) and ∀belard (the refuter) in order to determine the truth value of φ_0 in s_0, cf. [25]. Configurations are elements of $\mathcal{C} \subseteq \mathcal{S} \times Sub(\varphi_0)$, and written $t \vdash \psi$. Each *play* of $\Gamma_{\mathcal{M}}(s_0, \varphi_0)$ is a maximal sequence of configurations that starts with $s_0 \vdash \varphi_0$. The game rules are presented in Fig. 2. Each rule is marked by ∃ / ∀ to indicate which player makes the move. A rule is applied when the player is in configuration C_i, which is of the form of the upper part of the rule. C_{i+1} is then the configuration in the lower part of the rule. The rules shown in the first and third lines present a choice which the player can make. Since no choice is possible when applying the rules shown in the second line, both players can apply them. If no rule can be applied the play terminates.

Winning Criteria: Player ∃ *wins* a play $C_0, C_1, \ldots$ iff

1. there is an $n \in \mathbb{N}$, s.t. $C_n = t \vdash q$ with $\Theta(t)(q) = \top$ or $C_n = t \vdash \neg q$ with $\Theta(t)(q) = \bot$, or
2. the outermost variable that occurs infinitely often is of type ν.

Player ∀ *wins* a play $C_0, C_1 \ldots$ iff

4. there is an $n \in \mathbb{N}$, s.t. $C_n = t \vdash q$ with $\Theta(t)(q) = \bot$ or $C_n = t \vdash \neg q$ with $\Theta(t)(q) = \top$, or
5. the outermost variable that occurs infinitely often is of type μ.

Configurations of the form $t \vdash p$ or $t \vdash \neg p$ are called *terminal* configurations.

A (memoryless) *strategy* for player Q is a partial function $\sigma : \mathcal{C} \to \mathcal{C}$, such that its domain is the set of configurations where player Q moves. Player Q plays a game according to a strategy σ if all his choices agree with σ. A strategy for player Q is called a *winning strategy* if player Q wins every play where he plays according to this strategy.

We have the following relation between the game and the semantics.

Theorem 1. *Given a Kripke structure $\mathcal{M} = (\mathcal{S}, \mathcal{R}, \Theta)$, an $s \in \mathcal{S}$, and $\varphi \in \mathcal{L}_\mu$:*

(a) $[\![\varphi]\!]^{\mathcal{M}}(s) = \top$ iff Player $\exists$ has a winning strategy for $\Gamma_{\mathcal{M}}(s, \varphi)$,
(b) $[\![\varphi]\!]^{\mathcal{M}}(s) = \bot$ iff Player $\forall$ has a winning strategy for $\Gamma_{\mathcal{M}}(s, \varphi)$

3 A Multi-Valued Game for the μ-Calculus

In this section we investigate the multi-valued model checking problem from the game-theoretic point of view. For the rest of the section let $\mathcal{M}$ be a Kripke structure over lattice $\mathcal{L}$, s_0 a state in $\mathcal{M}$ and φ_0 a μ-calculus formula. We suggest a *multi-valued* model checking game, $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$, for evaluating φ_0 in state s_0 of $\mathcal{M}$.

The new game is still played by two players, $\exists$loise and $\forall$belard, and the moves of the players are defined as in the 2-valued game (see Fig. 2). In particular, the players can base their moves on the multi-valued transitions. However, the concept of winning needs to be adapted. In fact, to capture the multi-valued nature of the problem, we no longer talk about *winning* a play versus *losing* it. Instead, we now associate with each play a *value* which is an element from the lattice.

In our definitions we take the point of view of $\exists$loise (we could dually describe the game from the point of view of $\forall$belard)). Intuitively, we think of the value of a play as a measure for how close $\exists$loise is to winning; Winning of $\exists$loise in the 2-valued case now corresponds to the *top* value. Winning of $\forall$belard corresponds to the *bottom* value, but more values are possible. In these terms, the goal of the players is no longer to *win* the play. Instead, the goal of $\exists$loise is to maximize the resulting value, whereas the goal of $\forall$belard is to minimize this value.

Notation We refer to the configurations of $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ as *nodes* in a *game graph*. Nodes are divided to $\vee$-nodes, where $\exists$loise plays, versus $\wedge$-nodes, where $\forall$belard plays. Moves between configurations are *edges* in the graph. Each edge (move) has a value from the lattice: moves that use a transition of the model get the value of the transition. The rest get the $\top$ value. We abuse the notation of the transition relation and denote the value of an edge from n to n' by $\mathcal{R}(n, n')$. We refer to edges with values $\neq \top, \bot$ as *indefinite* edges.

Example 1. Consider the Kripke structure $\mathcal{M}$ of Fig. 3 over lattice $\mathcal{L}$, such that $x, y, z, w \in \mathcal{L}$. The labels of the transitions define their values. Unlabelled transitions have value $\top$. The states labelling denotes that $\Theta(s_0)(r) = z$, $\Theta(s_0)(h) = w$ and $\Theta(s_1)(q) = \Theta(s_2)(q) = \top$, where q, r, h are atomic propositions. Fig. 3 also shows the game-graph of the model checking game $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$, where $\varphi_0 = \Diamond q \wedge (r \vee h)$. Again, the edges are labelled by their values.

3.1 Plays and their Values

A play in $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ is defined as before. To understand how we determine the value of a multi-valued play, consider again a 2-valued play. As explained above, if the winner is $\exists$loise,

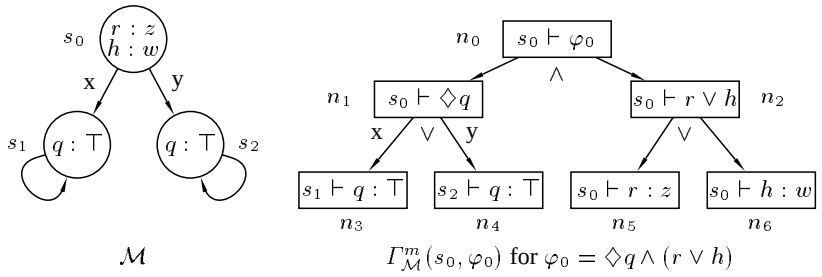

Fig. 3. Example of a Multi-Valued Game

then in the multi-valued context we view its value as $\top$. Similarly, if the winner is $\forall$belard, then we view the value as $\bot$. However, in the multi-valued case we have two extensions, which introduce more values. First, the terminal nodes ($t \vdash q$, $t \vdash \neg q$) are no longer classified as winning or losing, but they have a value which results from the value of q in the state t. This affects the values of finite plays.

Furthermore, the moves are also multi-valued, due to the multi-valued nature of the model's transitions. The value that a player gains in the play also depends on the values of the transitions that were used. Intuitively, one can think of the moves of $\exists$loise as attempts at proving the formula and the moves of $\forall$belard as attempts at refuting it. In this context, the use of indefinite edges in the multi-valued case is interpreted as a weak attempt at proving or refuting (depending on the player).

Recall that we think of the value of the play as a measure for how close $\exists$loise is to winning. Therefore, when evaluating a play we take the point of view of $\exists$loise. Conceptually, we first give the play a *base* value, while ignoring the values of edges used. We then update the resulting value based on the edges.

Definition 1. *For a terminal node $n = t \vdash q$, we define $val(n)$ to be $\Theta(t)(q)$. For $n = t \vdash \neg q$ we define $val(n)$ to be $\neg\Theta(t)(q)$.*

Definition 2. *Let p be a play in the game. We define its* base value, *denoted $base(p)$, as follows. If p is a finite play, $base(p) = val(n)$, where n is the terminal node in which p ends. If p is infinite, then $base(p) = \top$ if p is won by $\exists$loise in the 2-valued game. Otherwise $base(p) = \bot$.*

We update the base value by taking into consideration the values of the edges used by *both* players in the play. Intuitively, when $\exists$loise plays, she tries to show an evidence for truth. For her evidence to be "convincing", she needs to both continue to a position which is good for her (meaning that the certainty of her verification from it is high), *and* also use an edge with a high value (which corresponds in a way to high certainty). Consequently, the value of the play is given by the *glb* of the value of the edge and the value of the rest of the play. On the other hand, when $\forall$belard plays, he tries to refute. When looking at the situation from the point of view of $\exists$loise, she succeeds in her goal better if $\forall$belard either reaches a position that is good for her, *or* if he uses an edge of low value (alternatively: high negated value), in which case the certainty of his refutation is low. Therefore, the value of the play in this case is given by the *lub* of the *negation* of the value of the edge and the value of the rest of the play.

This intuition leads to a bottom-up computation of the value of a play. In order to formally define it we need the following definitions.

Definition 3. *Let $p = n_0, n_1, \ldots, n_k$ be a finite prefix of a play, and let $x \in \mathcal{L}$ be a base value. We define $update(p, x)$ by reverse induction. Initially, $val_k = x$. Given val_i, we define val_{i-1}*

depending on the player that made the move from n_{i-1} to n_i. If it is ∃loise, then $val_{i-1} = \mathcal{R}(n_{i-1}, n_i) \wedge val_i$. If the player is ∀belard, then $val_{i-1} = \neg\mathcal{R}(n_{i-1}, n_i) \vee val_i$. Finally, we let $update(p, x) = val_0$.

Note that edges with value $\top$ do not change the base value since $\top \wedge x = x$ and $\neg\top \vee x = x$ for all $x \in \mathcal{L}$ (since in a DeMorgan lattice $\neg\top = \bot$).

This definition is directly applicable to defining the value of a finite play by taking x to be the base of p. Unfortunately, it is not suitable for infinite plays.

To handle infinite plays, we use the following key observations. First, since the set of edges in the game graph is finite, we know that the set of edges used in the play is finite, and thus there exists a *finite* prefix of the (infinite) play that contains all of them. Furthermore, it turns out that computing the value of the play by considering only such a (finite) prefix is sufficient, in the following sense. We define the value $val(p_i)$ of a prefix p_i of an infinite play p similarly to the definition of the value of a finite play, except that the base value is set to the base value of the entire infinite play p. That is, $val(p_i) = update(p_i, base(p))$. We now have the property that the value of *any* prefix that contains all the edges used in p is the same.

Lemma 1. *Let p be an infinite play and let p_i, p_j be two finite prefixes of p that contain all of the edges that appear in p. Then $val(p_i) = val(p_j)$.*

In other words, the play has a *limit* value. This property is surprising since when considering the values of increasingly longer prefixes, the resulting sequence is not necessarily monotonic. Lemma 1 also implies that any finite prefix that contains all the edges of the play is a good representative for computing this value. Intuitively, this results from the property that an instance of an edge that is closer to the initial node, "absorbs" the effect of a further instance of the same edge. We therefore define the value of an infinite play to be the value of the minimal prefix that contains all the edges.

To sum up, the value of a play is defined as follows.

Definition 4. *For a finite play p, $val(p) = update(p, base(p))$. For an infinite play p, $val(p) = update(p_i, base(p))$, where p_i is the minimal prefix of p that contains all edges used in p.*

Example 2. Consider again the game described in Fig. 3. Terminal nodes in the game-graph are labelled by their values. One possible play in the game is $< n_0, n_1, n_3 >$. Its value is $\neg\top \vee (x \wedge \top) = x$. Another example is the play $< n_0, n_2, n_5 >$ whose value is $\neg\top \vee (\top \wedge z) = z$. More plays exist.

3.2 Strategies and their Values

As always, in order to talk about the relation to model checking, we need to talk about strategies, rather than a single play. In the 2-valued game we talked about *winning* strategies and we were guaranteed that exactly one of the players had one. In the multi-valued case, we no longer talk about winning. Instead, we talk about the *gain* of each player in the game. Therefore, the notion of a winning strategy is no longer what we need. Instead we need to talk about strategies for gaining a value.

Consider again the 2-valued game. A winning strategy for ∃loise in the 2-valued game guarantees that every play, where ∃loise plays by the strategy is winning for ∃loise (or has value $\top$). On the other hand, a non-winning strategy for ∃loise is such that there exists a play where ∃loise plays by the strategy, but the play is winning for ∀belard (has value $\bot$). Thus,

we can say that a winning strategy for ∃loise ensures the value $\top$, whereas a non-winning strategy ensures only $\bot$ (as it ensures a value $\geq \bot$, but not better than that). Furthermore, each strategy is either winning for ∃loise (ensures value $\top$) or non-winning (ensures only $\geq \bot$). Thus, strategies are comparable, and there always exists a *best* strategy. The best strategy is a winning strategy if one exists, or a non-winning one otherwise.

When we move to the general multi-valued case, a strategy for ∃loise is defined as usual. However, unlike the usual case, here plays can have many values, which may be *incomparable* to one another. Given a strategy $\sigma_\exists$ for ∃loise, the value that will be achieved in practice depends on the choices of ∀belard. We want the value of $\sigma_\exists$ to be a *lower bound* on the set of all possible values that can be achieved in plays where ∃loise plays by $\sigma_\exists$, with the meaning that the strategy ensures a value which is greater or equal than its value. We choose the *greatest* possible lower bound, which characterizes the strategy as precisely as possible.

Definition 5. *Let $\sigma_\exists$ be a strategy for ∃loise. We define $val(\sigma_\exists) = \bigwedge\{val(p) \mid p$ is a play by $\sigma_\exists\}$*

This definition implies that ∃loise can always achieve a value $\geq val(\sigma_\exists)$ in any play where she plays by the strategy $\sigma_\exists$. Note that since $val(\sigma_\exists)$ is given by the glb of possibly incomparable values, it is now possible that there does *not* exist a play with value $val(\sigma_\exists)$ by this strategy, but still the strategy cannot ensure a strictly better (higher) value.

Similarly to the phenomenon of several values achieved by a single strategy, it may be the case that ∃loise has several different strategies, with incomparable values. ∃loise chooses which of her strategies to use. We therefore define the value that she achieves in the *game* to be the *least upper bound* on the values of all her strategies. It gives us an upper bound, precise as possible, for the values that ∃loise can achieve in the game, with the meaning that ∀belard cannot enforce any value which is strictly lower than that.

Definition 6. *Let $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ be a multi-valued play. Then*

$$val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = \bigvee\{\alpha \mid \exists\textit{loise has a strategy } \sigma_\exists \textit{ with value } val(\sigma_\exists) = \alpha\}$$

Note that in the general case, ∃loise does *not* necessarily have a *best* strategy that achieves the lub. However, if the lattice has a total order then such a best strategy exists.

Example 3. In the game of Fig. 3 ∃loise has two possible moves from n_1 and n_2 (these are the nodes where she moves). Therefore she has four possible (memoryless) strategies – one for each combination. Consider for example the strategy σ_1 in which ∃loise always proceeds to the left successor. The choice in n_0 is of ∀belard, therefore there are two possible plays by this strategy: $< n_0, n_1, n_3 >$ (when ∀belard chooses the left successor of n_0) and $< n_0, n_2, n_5 >$ (when ∀belard chooses the right successor of n_0) whose values are x and z respectively (see Example 2). Since the choice between the plays is of ∀belard, the value of the strategy is the glb of their values. That is, $val(\sigma_1) = x \wedge z$. This means that by σ_1, ∃loise can only ensure a value which is $\geq x \wedge z$, where possibly $x \wedge z$ is strictly smaller than both x and z (see for example $\bot\top$ and $\top\bot$ in $L_{2,2}$).

Similarly, we get $val(\sigma_2) = x \wedge w$, $val(\sigma_3) = y \wedge z$ and $val(\sigma_4) = y \wedge w$. Since ∃loise chooses which strategy to use, the value of the game is then defined to be $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = val(\sigma_1) \vee val(\sigma_2) \vee val(\sigma_3) \vee val(\sigma_4) = (x \wedge z) \vee (x \wedge w) \vee (y \wedge z) \vee (y \wedge w)$. Note, that if all the latter values are incomparable, then ∃loise does *not* have a unique *best* strategy. By distributivity, we now get that $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = (x \wedge (z \vee w)) \vee (y \wedge (z \vee w)) = (x \vee y) \wedge (z \vee w)$. An inspection of the model shows that this is the value of $[\![\varphi_0]\!]^{\mathcal{M}}(s_0)$, which demonstrates the correctness of the game (see Theorem 2 in the following section).

Remark 1. One can think of the value of the game in the regular 2-valued case (from the point of view of ∃loise), as defined by the following formula

$$\exists \sigma_\exists \forall \sigma_\forall : val(outcome(\sigma_\exists, \sigma_\forall)) = \top$$

where $\sigma_\forall$ denotes a strategy for ∀belard and $outcome(\sigma_\exists, \sigma_\forall)$ is the (unique) play defined by the combination of $\sigma_\exists$ and $\sigma_\forall$. This formula describes the condition for a game to be won by ∃loise: it requires that ∃loise has a winning strategy $\sigma_\exists$, meaning that for each possible strategy $\sigma_\forall$ of ∀belard, the resulting play is winning for ∃loise (has value $\top$).

Similarly, in the multi-valued case, the definition of $val(\sigma_\exists)$ can be rephrased as $val(\sigma_\exists) = \bigwedge_{\sigma_\forall} \{val(outcome(\sigma_\exists, \sigma_\forall))\}$. This makes

$$val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = \bigvee_{\sigma_\exists} \{\bigwedge_{\sigma_\forall} \{val(outcome(\sigma_\exists, \sigma_\forall))\}\}$$

That is, we in fact replace the ∃ quantifier by the lub operator and replace the ∀ quantifier by the glb operator, since there is no longer a *best* strategy for ∃loise, and no longer a *best* strategy for ∀belard. A similar phenomenon happens when considering probabilistic games [10], where it is possible that the limit probability in which ∃loise wins is 1, but there is no strategy that achieves probability 1. Instead, for every probability, as close to 1 as we want, there is a strategy that achieves it. We then also replace the ∃ and ∀ quantifiers by *supremum* and *infimum* respectively.

3.3 Correctness

We now formalize and prove the correctness of the multi-valued model checking game.

Theorem 2. *Let $\mathcal{M}$ be a Kripke structure over lattice $\mathcal{L}$, s_0 a state in $\mathcal{M}$ and φ_0 a μ-calculus formula. Then $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = [\![\varphi_0]\!]^{\mathcal{M}}(s_0)$.*

To prove the theorem, we first give an alternative definition for the value of $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$, which mainly results from Birkhoff's representation theorem for finite distributive lattices.

Lemma 2. $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = \bigvee\{\alpha \mid \exists$*loise has a strategy* $\sigma_\exists$ *with value* $val(\sigma_\exists) \geq \alpha\}$
$= \bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid \exists$*loise has a strategy* $\sigma_\exists$ *with value* $val(\sigma_\exists) \geq \alpha\}$

We now use similar techniques to those used in the reduction approach of [4]. There, the multi-valued model checking problem is reduced to several 2-valued model checking problems. First, to avoid a technical problem with negated atomic propositions, the formula is transformed to a formula with no negation symbols, by replacing each negated proposition $\neg q$ by a new atomic proposition q'. The labelling function Θ of $\mathcal{M}$ is extended to Θ' by setting $\Theta'(s)(q') = \neg\Theta(s)(q)$. Then, the Kripke structure $\mathcal{M}$ over $\mathcal{L}$ is reduced to several Kripke Modal Transition Systems (KMTSs).

Definition 7. *[16] A* Kripke Modal Transition System (KMTS) *is a tuple $\tilde{\mathcal{M}} = (\tilde{\mathcal{S}}, R^+, R^-, \tilde{\Theta})$ with a* must *transition relation $R^+ \subseteq \mathcal{S} \times \mathcal{S}$ and a* may *transition relation $R^- \subseteq \mathcal{S} \times \mathcal{S}$. The labelling is given by $\tilde{\Theta} : \tilde{\mathcal{S}} \to (\mathcal{P} \to L_3)$.*

More specifically, given a join-irreducible element α of $\mathcal{L}$, a reduced KMTS $\mathcal{M}_\alpha$ is defined by setting

$$\Theta_\alpha(s)(q) = \Theta(s)(q) \geq \alpha$$
$$R^+_\alpha(s, s') = R(s, s') \geq \alpha$$
$$R^-_\alpha(s, s') = \neg R(s, s') \not\geq \alpha$$

The formula is then interpreted over the KMTS $\mathcal{M}_\alpha$ w.r.t. a *2-valued* semantics, with the main difference being that

$$[\![\Diamond\varphi]\!]_\rho^{\mathcal{M}_\alpha} := \lambda s. \bigvee\{R_\alpha^+(s,s') \wedge [\![\varphi]\!]_\rho^{\mathcal{M}_\alpha}(s') \mid \text{all } s'\}$$
$$[\![\Box\varphi]\!]_\rho^{\mathcal{M}_\alpha} := \lambda s. \bigwedge\{\neg R_\alpha^-(s,s') \vee [\![\varphi]\!]_\rho^{\mathcal{M}_\alpha}(s') \mid \text{all } s'\}$$

As shown in [4], it then holds that $(\mathcal{M}_\alpha, s_0) \models \varphi_0 \Leftrightarrow \alpha \leq [\![\varphi_0]\!]^{\mathcal{M}}(s_0)$, and the following lemma is implied.

Lemma 3. *[4]* $[\![\varphi_0]\!]^{\mathcal{M}}(s_0) = \bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid (\mathcal{M}_\alpha, s_0) \models \varphi_\prime\}$.

We use the above results to prove the correctness of our multi-valued game. For this purpose we show that

Lemma 4. $\bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid \exists$*loise has a strategy* $\sigma_\exists$ *with value* $val(\sigma_\exists) \geq \alpha\} = \bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid (\mathcal{M}_\alpha, s_0) \models \varphi_0\}$.

Combining this lemma with Lemmas 2 and 3 implies that $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = [\![\varphi_0]\!]^{\mathcal{M}}(s_0)$.

Proof. To prove Lemma 4 we refer to the 2-valued game for KMTSs, defined in [23]. The 2-valued game for KMTSs is similar to the 2-valued game for Kripke structures. The difference is that $\exists$loise uses only must-transitions, whereas $\forall$belard uses may-transitions. The winning conditions remain as before, with the exception that a player can get stuck (if R^+ or R^- is not total), in which case he loses. Theorem 1 holds for this case as well. In our case this means that $\exists$loise has a winning strategy in the 2-valued game over $\mathcal{M}_\alpha$ iff $(\mathcal{M}_\alpha, s_0) \models \varphi_0$.

We show a 1-1 correspondence between strategies of $\exists$loise with value $\geq \alpha$ in the multi-valued game over $\mathcal{M}$ and winning strategies for $\exists$loise in the 2-valued game over the KMTS $\mathcal{M}_\alpha$.

We use the following property of join-irreducible elements in a distributive lattice $\mathcal{L}$. If α is a join-irreducible element of $\mathcal{L}$ and $y, z \in \mathcal{L}$, then $\alpha \leq y \vee z$ iff $\alpha \leq y$ or $\alpha \leq z$ [9].

Let σ be a strategy for $\exists$loise with $val(\sigma) \geq \alpha$. We show that the same strategy is a winning strategy in the 2-valued game. Consider a play played by this strategy in the 2-valued case. We show that $\exists$loise wins it. We know that in the multi-valued game its value is $\geq \alpha$.

First, if the play is infinite (in the 2-valued case) then the value y of each edge used by $\forall$belard is such that $\neg y \not\geq \alpha$ (otherwise it does not exist as a may-edge). Thus for the value of the play to be $\geq \alpha$, its base value has to be $\top$. This is because only edges of $\forall$belard can increase the value and by the previous property they cannot increase a base value of $\bot$ to be $\geq \alpha$, since α is join-irreducible. Since the base value is $\top$, we conclude that the play fulfills the winning criteria of $\exists$loise in the 2-valued game.

If the play is finite (in the 2-valued case) then we first show that it cannot be the case that $\exists$loise is stuck. If $\exists$loise is stuck it means that the strategy defines for her to use an edge with a value $y \not\geq \alpha$ (that does not exist as a must-edge in the 2-valued case). The same reasoning as before shows that for the value of the play to be $\geq \alpha$, there had to be an earlier edge of $\forall$belard with value y such that $\neg y \geq \alpha$, but such an edge does not exist as a may-edge in the 2-valued play, which leads to contradiction. Thus, either $\forall$belard gets stuck, in which case $\exists$loise wins, or the play ends in a terminal node labeled by an atomic proposition $n = s \vdash q$. In the latter case, we again conclude by the same reasons that $val(n) \geq \alpha$, thus $\Theta(s)(q) \geq \alpha$ and in the KMTS $\mathcal{M}_\alpha$ this implies $\Theta_\alpha(s)(q) = \top$ and $\exists$loise wins.

For the other direction, let σ be a winning strategy for $\exists$loise in the 2-valued game. Once again, we show that the same strategy has a value $\geq \alpha$ in the multi-valued game, with the exception that if σ does not define a move from some configuration, we extend it arbitrarily. To prove that $val(\sigma) \geq \alpha$ we show that the value of every play where $\exists$loise plays by (the extended) σ in the multi-valued game has value $\geq \alpha$. Consider such a play.

First, if the same play exists in the 2-valued game, then it is winning for ∃loise, making its base value $\top$. Furthermore, all the edges used by ∃loise are must-edges, with values $\geq \alpha$. Since only edges of ∃loise can decrease the value of the play, this ensures that the value of the play is $\geq \alpha$.

If the play does not exist in the 2-valued game, it means that one of two possibilities occurred. The first one is that ∀belard used an edge that does not exist as a may-edge in the 2-valued game, meaning that its value y fulfills $\neg y \geq \alpha$. But this immediately increases the value of the suffix of the play from that point to be $\geq \alpha$. By the same reasons as before the prefix of the play does not decrease the value below α, and thus it remains $\geq \alpha$. The second possibility is that ∃loise used an edge that does not exist in the 2-valued game. This could only happen if the play reached a configuration where σ was extended. This means that originally, in the 2-valued game, this configuration was not reachable by σ. But then it has to be the case that in order to reach it ∀belard made a move that was not possible in the 2-valued game, and we return to the first possibility. □

4 Solving the Multi-Valued Game

In this section we discuss how to solve the multi-valued model checking game. Given a game $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ our purpose is to compute its value. By Theorem 2 this gives us the result of the multi-valued model checking problem for $\mathcal{M}$, s_0 and φ_0. Since the game is defined directly on the multi-valued Kripke structure, we get a *direct* model checking algorithm for the multi-valued problem, that has all the advantages of the game-theoretic approach [24, 13].

As usual, we solve the game by processing the game-graph and evaluating each node in it. The difference as opposed to the 2-valued case is that we need to propagate values from the lattice. We demonstrate this change for the *alternation-free* fragment of the μ-calculus, where no nesting of fixpoints is allowed.

We partition the game graph to *Maximal Strongly Connected Components* (MSCCs) and determine a (total) order on them, reflected by their numbers: $Q_1, \ldots, Q_k$. The order fulfills the rule that if $i < j$ then there are no edges from Q_i to Q_j. Such an order exists because the MSCCs of the game-graph form a *directed acyclic graph* (DAG). The components are handled bottom-up.

Consider a single component Q_i. We label each of its nodes n with a value, denoted $res(n)$, as follows. A terminal node n is given the value $val(n)$. For an $\vee$-node n we set $res(n)$ to be $\bigvee\{\mathcal{R}(n, n') \wedge res(n') \mid \mathcal{R}(n, n') \neq \bot\}$. Similarly, if n is an $\wedge$-node then $res(n) = \bigwedge\{\neg\mathcal{R}(n, n') \vee res(n') \mid \mathcal{R}(n, n') \neq \bot\}$.

To handle Q_i's that form a non-trivial MSCC, we use the following observation: when dealing with the alternation-free fragment of the μ-calculus, an infinite play has exactly one variable that occurs infinitely often [25]. Therefore, if Q_i is a non-trivial MSCC then it contains exactly one fixpoint variable Z. In this case we first label the nodes in Q_i with temporary values, $temp(n)$, that are updated iteratively. For nodes of the form $n_w = t \vdash Z$ we initialize $temp(n_w)$ to $\top$ if Z is of type ν, or to $\bot$ if Z is of type μ (the rest of the nodes remain uninitialized). We then apply the previous rules until the temporary values do not change anymore. Finally, we set $res(n) = temp(n)$ for every node n in Q_i. Intuitively, this algorithm imitates the iterative computation of the fixpoint.

Several optimizations can be made on this computation. For example, consider an $\vee$-node n with a successor n' for which $res(n')$ is already computed. Furthermore, suppose that the values of edges leading to the rest of the successors of n have values $\leq \mathcal{R}(n, n') \wedge res(n')$. This

$$\frac{s \vdash \Diamond\varphi}{(s,t) \vdash \Diamond\varphi}\ \exists:\ \mathcal{R}(s,t) \neq \bot \qquad \frac{s \vdash \Box\varphi}{(s,t) \vdash \Box\varphi}\ \forall:\ \mathcal{R}(s,t) \neq \bot$$

$$\frac{(s,t) \vdash \Diamond\varphi}{(s,t) \vdash \top \text{ or } t \vdash \varphi}\ \forall \qquad \frac{(s,t) \vdash \Box\varphi}{(s,t) \vdash \bot \text{ or } t \vdash \varphi}\ \exists$$

Fig. 4. New rules for $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$.

means that the rest of the successors cannot increase the result of the lub over all successors of n and we can immediately set $res(n)$ to be $\mathcal{R}(n,n') \wedge res(n')$, regardless of whether or not the rest of its successors were handled. Such optimizations can spare us the need to process big subgraphs.

Theorem 3. *Let $\mathcal{M}$ be a Kripke structure over $\mathcal{L}$. Then for every state s_0 in $\mathcal{M}$ and every closed μ-calculus formula φ_0, we have that $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = res(s_0 \vdash \varphi_0)$. We conclude that $[\![\varphi_0]\!]^{\mathcal{M}_0}(s_0) = res(s_0 \vdash \varphi_0)$.*

5 Avoiding Multi-Valued Edges in the Game

Recall that the multi-valued edges used in the game posed a problem when we wanted to define the value of an infinite play. Our treatment of such plays strongly relied on the finite nature of the Kripke structure and the game graph. In this section we suggest a different way of overcoming the problem, which is also suitable for infinite structures. The new definition makes the value of a play much simpler to define.

We change the rules of the game, and in fact reduce the multi-valued edges into more multi-valued terminal nodes. We wish to emphasize that the reduction is performed in the game level, rather than the model level. That is, the Kripke structure that the game is based on still has multi-valued transitions.

The main idea is to split every move along a multi-valued transition (of the model) into two moves: first the player who is supposed to play chooses a transition. Then, the opponent chooses whether he wants to examine the value of the transition or to continue in the play. If he chooses the transition, the play ends with this value. This means that there are no longer multi-valued edges in the game. We only have multi-valued terminal nodes.

Formally, given a Kripke structure $\mathcal{M}$ over lattice $\mathcal{L}$, a state s_0 and a μ-calculus formula φ_0, we define $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ as follows.

The configurations of the game are as before, with additional configurations of the form $(s,t) \vdash \Diamond\varphi$, $(s,t) \vdash \Box\varphi$, $(s,t) \vdash \top$ and $(s,t) \vdash \bot$. The new configurations act as intermediate configurations for the new rules of the game. The rules are given by Fig. 2, where the rules in the third line are replaced by the rules in Fig. 4.

For example, in a configuration of the form $s \vdash \Diamond\varphi$, $\exists$loise chooses, as usual, a transition that is supposed to show evidence for $\Diamond\varphi$. Since it is a move of $\exists$loise, we have the meaning of the lub of all possibilities. However, the next move is a move of $\forall$belard, with the meaning of the glb between the two options. This means that for each possibility of $\exists$loise we examine the glb of both the value of the transition and the value of the rest of the play. Configurations of the form $s \vdash \Box\varphi$ are handled dually.

Configurations of the form $(s,t) \vdash \top$ and $(s,t) \vdash \bot$ are (new) terminal configurations. A configuration of the form $(s,t) \vdash \top$ is reached when $\forall$belard challenges the transition that $\exists$loise chose from $s \vdash \Diamond\varphi$. It expresses the fact that we are interested in the value of $\mathcal{R}(s,t)$

that determines the certainty in which ∃loise tries to prove the existential property. Dually, a configuration of the form $(s,t) \vdash \bot$ is reached when ∃loise challenges the transition that ∀belard chose from $s \vdash \Box\varphi$. In this case, we are interested in the value of $\neg\mathcal{R}(s,t)$, since from the point of view of ∃loise, her chances of proving are better as the value of $\mathcal{R}(s,t)$ used by ∀belard for refutation is lower (alternatively: $\neg\mathcal{R}(s,t)$ is higher). Following this intuition, we add the following definition.

Definition 8. *For a terminal node $n = (s,t) \vdash \top$, we define $val(n)$ to be $\mathcal{R}(s,t)$. For $n = (s,t) \vdash \bot$ we define $val(n)$ to be $\neg\mathcal{R}(s,t)$.*

Since there are no longer multi-valued edges in the game, the value of a play is now determined to be the base value, as defined earlier (see Definition 2) – no update is needed. The rest of the definitions of strategies, their values and the value of the game remain unchanged. The following theorem ensures that we get the same value in the new game, thus the correctness of the game is maintained.

Theorem 4. *Let $\mathcal{M}$ be a Kripke structure over lattice $\mathcal{L}$, with a state s_0 and let φ_0 be a μ-calculus formula. Then $val(\Gamma^d_{\mathcal{M}}(s_0,\varphi_0)) = val(\Gamma^m_{\mathcal{M}}(s_0,\varphi_0))$.*

6 Discussion: Games versus Automata

In this paper we have investigated the multi-valued model checking problem from the game-theoretic point of view. In [4] the same problem was considered from the automata-theoretic point of view. There, model checking is performed by checking the nonemptiness of an automaton that represents the product of the model and the checked formula. In this section we discuss the essential difference between the two approaches in the multi-valued case.

It is well-known that the two approaches – game-based and automata-based – are closely related in the 2-valued setting: an accepting run corresponds to a winning strategy for ∃loise and vice versa [22]. Surprisingly, the same relation does *not* hold anymore in the multi-valued case. More specifically, in [4] *extended alternating automata* (*EAAs*) were used as the basis for model checking. To capture the multi-valued nature they referred to the *value* of an accepting run. They showed that there always exists an accepting run of the EAA with a *maximal* value. This maximal value defines the value of the emptiness of the automaton. In the multi-valued game, on the other hand, it is *not* necessarily the case that there exists a strategy of ∃loise with a maximal value. This clearly demonstrates the discrepancy between automata and games in the multi-valued setting.

It is possible to regain the relation to the automata-theoretic approach by defining the game differently. The alternative game is still played over the same game-graph, but the moves are different. Initially, ∃loise makes a statement with respect to the value of the initial node n_0, denoted $bet(n_0)$. In each node she proceeds by associating (possibly a subset) of its successors with a value in a consistent way based on the type of the node: in an $\vee$-node n the values have to fulfill the rule $bet(n) = \bigvee\{\mathcal{R}(n,n') \wedge bet(n') \mid \mathcal{R}(n,n') \neq \bot\}$. In an $\wedge$-node the values have to fulfill the rule $bet(n) = \bigwedge\{\neg\mathcal{R}(n,n') \vee bet(n') \mid \mathcal{R}(n,n') \neq \bot\}$. Once a bet is made on the value of a node it cannot be changed. The role of ∀belard is then to choose one successor n' for which ∃loise needs to continue and prove the value $bet(n')$. Intuitively, ∀belard will try to choose a successor for which the value is incorrect.

In this definition we return to talking about *winning* versus losing. Intuitively, ∃loise wins if she manages to proceed without contradictions. Formally, if ∃loise is stuck (meaning she

cannot associate the successors of the current node with values according to the above rules) then ∀belard wins. If the play ends in a terminal configuration of the form $s \vdash q$ or $s \vdash \neg q$, then ∃loise wins iff the value she gave the node matches its real value ($\Theta(s)(q)$ or $\neg\Theta(s)(q)$ resp.). In an infinite play the winner is determined by the 2-valued winning conditions.

Note that here ∃loise moves in *both* types of nodes, which changes the basic nature of the game. However, we now have the desired property that the game is equivalent to the definition used in the context of EAA. It now holds that an accepting run of the automaton with value α corresponds to a winning strategy for ∃loise with an initial bet of value α, and vice versa. Thus, there exists a *maximum* value for which ∃loise has a winning strategy and this value is the multi-valued model checking result.

Our definition of the game is in fact more general than the automaton used in [4] as it handles the multi-valued transitions of the structure directly.

7 Comparison to the 3-Valued Game

One of the most useful applications of multi-valued model checking is the 3-valued case. In [24, 13] the regular model checking game has been generalized to a 3-valued game over a KMTS (see Definition 7). A KMTS $\mathcal{M}$ can be viewed as a Kripke structure over lattice L_3 by giving the must transitions in R^+ value $\top$, the may transitions in $R^- \setminus R^+$ value U and the rest value $\bot$. In this section we compare the game of [13] to our general multi-valued game $\Gamma^m_{\mathcal{M}}(s, \varphi)$ and point out the main differences that make the 3-valued game much simpler.

When considering the 3-valued case, a play can have three possible values: $\top$, $\bot$ or U (see L_3 in Fig. 1). In this case it is possible to give the indefinite value U an intuitive meaning of a *tie*. If in addition we define $\top$ as winning of ∃loise and $\bot$ as winning of ∀belard we have the following correspondence between the value of the game and the formula, given by a variant of Theorem 1, with an additional possibility [13]:

(c) $[\![\varphi]\!]^{\mathcal{M}}(s) = U$ iff no player has a winning strategy for $\Gamma_{\mathcal{M}}(s, \varphi)$

That is, in the 3-valued case we can still talk about the notion of *winning* in a way that corresponds to the three possible values $\{\top, U, \bot\}$ in the logic [13]. This is unlike the multi-valued case where we need to talk about the general notion of a *value* of a play or a game.

Another major difference arises from the fact that the lattice L_3 has a *total* order, meaning that all values are comparable. As a result, in the 3-valued case a strategy has a *precise* value (rather than a lower bound) and the same holds for the game. That is, strategies are comparable and there always exists a *best* strategy (either winning or non-winning) that determines the value of the game.

The combination of these differences results in another interesting property of the 3-valued game. As in the general multi-valued case, the result of the play in the 3-valued case also depends on the values of the edges that were used. However, in [13] this effect is captured by a *consistency* requirement that says that in order to win, the winner has to use only must edges (with value $\top$). The surprising part is that the opponent can use either type. Recall that in the general multi-valued case, on the other hand, we need to consider not only the edges that one player uses, but also those used by the opponent.

This results from the fact that in the 3-valued case only one intermediate result is possible. Furthermore, we have a total order on the elements of the lattice, which means that a value cannot be achieved by a combination of values that are all different from it. Thus the values of the edges that the opponent uses in the 3-valued game cannot improve the result for the other

player beyond a tie (or U). Thus they are irrelevant when we determine a *winner* in the play – recall that in the 3-valued case we are interested in the *winner* of the play. This is no longer the case in the multi-valued case, where we are interested in the (more general notion of a) *value* that each player achieves and this value can be achieved by a combination of several values, possibly incomparable ones.

References

1. T. Ball, O. Kupferman, and G. Yorsh. Abstraction for falsification. In *CAV*, 2005.
2. G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Computer Aided Verification*, pages 274–287, 1999.
3. G. Bruns and P. Godefroid. Temporal logic query checking. In *LICS*. IEEE, 2001.
4. G. Bruns and P. Godefroid. Model checking with multi-valued logics. In *ICALP*, 2004.
5. W. Chan. Temporal-logic queries. In *CAV*, volume 1855 of *LNCS*, pages 450–463, 2000.
6. M. Chechik, B. Devereux, and A. Gurfinkel. Model-checking infinite state-space systems with fine-grained abstractions using spin. In *SPIN Workshop*, volume 2057 of *LNCS*, 2001.
7. M. Chechik, B. Devereux, A. Gurfinkel, and S. Easterbrook. Multi-valued symbolic model-checking. Technical Report CSRG-448, University of Toronto, April 2002.
8. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT press, 1999.
9. B.A. Davey and H.A. Priestly. *Introduction to Lattices and Order*. Cambrifge University Press, 1990.
10. L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. In *STOC*, 2001.
11. S.M. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *ICSE*, pages 411–420, 2001.
12. P. Godefroid and R. Jagadeesan. Automatic abstraction using generalized model checking. In *CAV*, volume 2404 of *LNCS*, pages 137–150, 2002.
13. O. Grumberg, M. Lange, M. Leucker, and S. Shoham. Don't know in the μ-calculus. In *VMCAI*, 2005.
14. A. Gurfinkel and M. Chechik. Multi-valued model checking via classical model checking. In *CONCUR*, pages 263–277, 2003.
15. A. Gurfinkel, B. Devereux, and M. Chechik. Model exploration with temporal logic query checking. In *FSE*, pages 139–148. ACM, 2002.
16. M. Huth, R. Jagadeesan, and D. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In *ESOP*, volume 2028, pages 155–169, 2001.
17. M. Huth and S. Pradhan. Lifting assertion and consistency checkers from single to multiple viewpoints. Technical Report 2002/11, Dept. of Computing, Imperial College, London, 2002.
18. B. Konikowska and W. Penczek. Reducing model checking from multi-valued CTL* to CTL*. In *CONCUR*, volume 2421 of *LNCS*, 2002.
19. B. Konikowska and W. Penczek. Model checking multi-valued modal mu-calculus: Revisited. In *Proc. of CS&P'04*, 2004.
20. D. Kozen. Results on the propositional μ-calculus. *TCS*, 27, 1983.
21. O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM (JACM)*, 47(2):312–360, 2000.
22. M. Leucker. Model checking games for the alternation free mu-calculus and alternating automata. In *LPAR*, 1999.
23. S. Shoham. A game-based framework for CTL counterexamples and abstraction-refinement. Master's thesis, Dept. of Computer Science, Technion - Israel Institute of Technology, 2003.
24. S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. In *CAV*, volume 2725 of *LNCS*, pages 275–287, 2003.
25. C. Stirling. Local model checking games. In *CONCUR*, volume 962 of *LNCS*, 1995.
26. A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific J.Math.*, 5:285–309, 1955.