Multi-Valued Model Checking Games ⋆

Sharon Shoham and Orna Grumberg

Computer Science Department, Technion, Haifa, Israel

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. This problem has many applications in verification, such as handling abstract or partial models, analyzing systems in the presence of inconsistent views, and performing temporal logic query checking. We define a new game for the multi-valued model checking problem of the full μ-calculus, and demonstrate how to derive from it a *direct* model checking algorithm for its alternation-free fragment. The algorithm 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 automata-based and the game-based approach 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 [2] 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

⋆ A preliminary version of this paper appeared in [1].

Email addresses: sharonsh@cs.technion.ac.il (Sharon Shoham), orna@cs.technion.ac.il (Orna Grumberg).

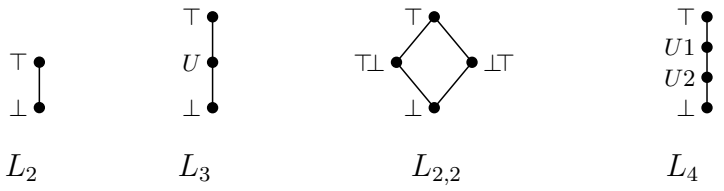

Fig. 1. Examples of Lattices

structures or structures with partial information [3–5]. In this context, the value U is used to model *uncertainty*, with the meaning that the value can either be $\top$ or $\bot$. 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 [6,7]. Temporal logic query checking [8–10] 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 [11,7,12–14] or 3-valued problems [15].

As opposed to the reduction approach, the *direct* approach checks the property on the multi-valued structure directly. Thus, it has the advantage of a more "on-the-fly" nature. Furthermore, a direct model checker can provide auxiliary information that explains its result. Such information can help analyzing the result. For example, in [4,5] the result of a direct model checking is used to suggest refinement of a 3-valued abstract structure. The same information cannot be retrieved from the model checking of two 2-valued structures [16].

Several direct model checking algorithms for various multi-valued logics have been suggested in the literature. [3–5] studied the 3-valued case of CTL ([3,4]) and the μ-calculus ([5]). In [17] the logic LTL was considered over finite linear orders. The general multi-valued version of CTL was handled in [18]. Finally, an *almost* direct automata-based algorithm for the multi-valued μ-calculus was suggested in [14]. Their approach handles the multi-valued labelling directly, but still uses a reduction 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. *μ-calculus* [19] 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 uses its multi-valued semantics based on any finite distributive De Morgan lattice.

We base our algorithm on the *game-theoretic* approach [20] and thus gain all of its advantages. For example, the game-based approach combines the system model and the checked formula into one structure (the *game graph*),

thus it enables *local* model checking, where only the parts of the model that are relevant to the satisfaction of the formula are explored [21]. This combined structure can be computed *on-the-fly*, limited to the reachable states of the model, which carries another advantage. Moreover, strategies of the players in the model checking game can be used as witnesses (or counterexamples) to support the model checking result. More examples are the works of [4,5], where game-based methods are used to produce counterexamples and to refine an abstract model when refinement is needed.

In the traditional game-based approach to model checking two players, the verifier (called ∃loise) and the refuter (called ∀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 strategies and of the game. It turns out that in the multi-valued case there does *not* necessarily exist a *best* strategy for ∃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 μ-calculus and prove their relation to the multi-valued model checking problem. The proof turns out to be interesting in itself, as it uses similar techniques to those used in the reduction approach of [14]. This is in spite of the fact that our approach handles the multi-valued structure directly and uses no reductions. In this sense, the proof establishes a connection between our direct approach and the reduction-based approaches.

To demonstrate the applicability of our work, we show how to derive from the game a *direct* multi-valued model checking algorithm that handles the multi-valued structure without any reduction. Since game-based model checking algorithms for the full μ-calculus are complicated even in the 2-valued case, we demonstrate this for the *alternation-free* fragment of the μ-calculus, where no nesting of fixpoints is allowed.

When comparing our multi-valued game to the work of [14], a surprising property is revealed. The direct algorithm of [14] is based on automata [22]. Usually, the game-based and the automata-based approaches to model checking have a strong resemblance [23]. Namely, a winning strategy of ∃loise in the model checking game corresponds to an accepting run of the automaton describing the product of the formula and the model, and vice versa. Yet, in the multi-valued setting, we find that our definition of the multi-valued game is different in essence from the automata-based approach of [14]. There, an accepting run of the automaton has a value and the multi-valued model checking result is the maximum value among all accepting runs, where a maximum value always *exists*. This demonstrates the discrepancy between the two approaches, as in the

multi-valued game ∃loise does *not* necessarily have a best strategy (one with a maximum value). 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 [14], 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 [4,5]. 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 5. In Section 4, 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. We conclude in Section 8.

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 $\bigvee A$ (see Fig. 1 for examples).

A lattice is *complete* if every subset of elements (not necessarily finite) has a glb and a lub. In particular, a complete lattice has a greatest element, called *top*, denoted $\top$, and a least element, called *bottom*, denoted $\bot$.

Throughout this paper we refer to finite distributive De Morgan lattices. Every finite lattice is *complete*, which ensures that it has a top element and a bottom element. In a *distributive* lattice $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ and $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ for all lattice elements x, y, z. In a *De Morgan* lattice every element $x \in \mathcal{L}$ has a unique complement $\neg x \in \mathcal{L}$ such that $\neg\neg x = x$, De Morgan's laws hold, and $x \leq y$ implies $\neg y \leq \neg x$.

A *join-irreducible* element x of a distributive lattice $\mathcal{L}$ is an element $\neq \bot$ s.t. $x = y \vee z$ 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).

The *height* of a lattice $\mathcal{L}$, denoted $h(\mathcal{L})$, is the maximum length of a sequence of elements $x_1, x_2, \ldots$ from $\mathcal{L}$ such that $x_1 < x_2 < \ldots$. If $\mathcal{L}$ is finite then its height is also finite, and bounded by $|\mathcal{L}|$. For example, $h(L_{2,2}) = 3$, since the longest strictly increasing sequences of elements from $L_{2,2}$ are either $\bot, \top\bot, \top$ or $\bot, \bot\top, \top$, both of which are of length 3.

μ-calculus [19] 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 write η for either μ or ν. 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 as usual.

A μ-calculus formula φ is *alternation-free* if for every subformula $\eta Z.\psi \in Sub(\varphi)$, no variable other than Z occurs freely in ψ. This means that there is no nesting of fixpoints.

Throughout this paper, unless explicitly stated otherwise, we refer to the *full* μ-calculus. Only Section 5 refers to the alternation-free fragment of the μ-calculus.

Semantics The *concrete semantics* of a μ-calculus formula is given with respect to a Kripke structure. A (finite) Kripke structure is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{R}, \Theta)$, 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* [1], and $\Theta : \mathcal{S} \to 2^{\mathcal{P}}$ is a labelling function [2].

In this work we consider the multi-valued μ-calculus [14], 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

[1] Since the totality assumption of the transition relation in a concrete Kripke structure is quite standard, we continue with this assumption in the multi-valued case as well. However, as we explain in Section 3 (see Remark 20), our results can easily by adapted to the case where the transition relation is not total.

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 2 in Section 3). 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. For $g : \mathcal{S} \to \mathcal{L}$, we denote with $\rho[Z \mapsto g]$ 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 $f = \lambda g.[\![\varphi]\!]_{\rho[Z \mapsto g]}$ is an element of $(\mathcal{S} \to \mathcal{L}) \to (\mathcal{S} \to \mathcal{L})$ and $\mathit{gfp}(f)$, $\mathit{lfp}(f)$ stand for the greatest and least fixpoints of f. These fixpoints exist according to [24], 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.

$$
\begin{array}{lcl}
[\![q]\!]_{\rho} & := & \lambda s.\Theta(s)(q) \\
[\![\neg q]\!]_{\rho} & := & \lambda s.\neg\Theta(s)(q) \\
[\![\varphi_1 \vee \varphi_2]\!]_{\rho} & := & \lambda s.\left([\![\varphi_1]\!]_{\rho}(s) \vee [\![\varphi_2]\!]_{\rho}(s)\right) \\
[\![\varphi_1 \wedge \varphi_2]\!]_{\rho} & := & \lambda s.\left([\![\varphi_1]\!]_{\rho}(s) \wedge [\![\varphi_2]\!]_{\rho}(s)\right) \\
[\![Z]\!]_{\rho} & := & \rho(Z) \\
[\![\mu Z.\varphi]\!]_{\rho} & := & \mathit{lfp}(\lambda g.[\![\varphi]\!]_{\rho[Z \mapsto g]}) \\
[\![\nu Z.\varphi]\!]_{\rho} & := & \mathit{gfp}(\lambda g.[\![\varphi]\!]_{\rho[Z \mapsto g]}) \\
[\![\Diamond\varphi]\!]_{\rho} & := & \lambda s.\bigvee\{\mathcal{R}(s, s') \wedge [\![\varphi]\!]_{\rho}(s') \mid \mathcal{R}(s, s') \neq \bot\} \\
[\![\Box\varphi]\!]_{\rho} & := & \lambda s.\bigwedge\{\neg\mathcal{R}(s, s') \vee [\![\varphi]\!]_{\rho}(s') \mid \mathcal{R}(s, s') \neq \bot\}
\end{array}
$$

Intuitively, in the multi-valued case, the value of a formula φ in a state s measures how close the formula is to being satisfied by s. This is a generalization of the concrete case, where a state either satisfies a formula φ or falsifies it. For example, in the concrete case, the $\Diamond$ and $\Box$ operators stand for "exists a successor" and "all successors", respectively. In the multi-valued case, their semantics takes into account the values of the transitions. For example, $[\![\Diamond\varphi]\!]_{\rho}(s) = \bigvee\{\mathcal{R}(s, s') \wedge [\![\varphi]\!]_{\rho}(s') \mid \mathcal{R}(s, s') \neq \bot\}$. This means that rather than determining whether there exists a successor of s that satisfies φ, the value of $\Diamond\varphi$ determines how close s is to having such a successor. Namely, each successor s' contributes the value of the transition that leads to it, met with the value measuring the extent to which s' satisfies φ. This means that if the value of the transition leading to s' is "low", then the contribution of the successor

s' decreases. The lub of these values, which generalizes the boolean notion of existence, determines the value of $\Diamond\varphi$ in s. Dually, the value of $\Box\varphi$ in a state s determines how close s is to having all of its successors satisfy φ. In this case, the contribution of each successor is the lub of the negated value of the transition leading to it and the value of φ in it. This means that a transition with a "low" value increases the value that the successor contributes, or, in other words, decreases the extent to which it interferes with the satisfaction of $\Box\varphi$ in s. Finally, the value of $\Box\varphi$ in s is determined by the glb of these values, which generalizes the boolean notion of universality.

Note that removing the restriction $\mathcal{R}(s, s') \neq \bot$ in the definitions of $[\![\Diamond\varphi]\!]_\rho$ and $[\![\Box\varphi]\!]_\rho$ yields an equivalent definition. We keep the restriction to emphasize that such transitions do not affect the semantics of formulas of the form $\Diamond\varphi$ and $\Box\varphi$, and need not be considered when evaluating them.

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 $(\mathcal{M}, s) \models \varphi$ 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 $\mathcal{M} = (\mathcal{S}, \mathcal{R}, \Theta)$ with $s_0 \in \mathcal{S}$ and a formula $\varphi_0 \in \mathcal{L}_\mu$ is played by players $\exists$loise (the prover) and $\forall$belard (the refuter) in order to determine the truth value of φ_0 in s_0, cf. [20]. Configurations are elements of $\mathcal{C} \subseteq \mathcal{S} \times \mathit{Sub}(\varphi_0)$, 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 $\exists$ / $\forall$ 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 in the second line, both players can apply them. If no rule can be applied the play terminates. This happens in *terminal* configurations of the form $t \vdash p$ or $t \vdash \neg p$.

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

(1) the play terminates in $t \vdash q$ with $\Theta(t)(q) = \top$ or $t \vdash \neg q$ with $\Theta(t)(q) = \bot$, or
(2) the outermost variable that occurs infinitely often is of type ν.

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

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

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

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

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

(3) the play terminates in $t \vdash q$ with $\Theta(t)(q) = \bot$ or $t \vdash \neg q$ with $\Theta(t)(q) = \top$, or
(4) the outermost variable that occurs infinitely often is of type μ.

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 *[20] For a regular Kripke structure $\mathcal{M} = (\mathcal{S}, \mathcal{R}, \Theta)$, $s \in \mathcal{S}$, and μ-calculus formula $\varphi \in \mathcal{L}_\mu$:*

(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, in the rules of the third line the players can make a move along any transition whose value is not $\bot$. However, the concept of winning needs to be adapted. In fact, to capture the multi-valued nature of the problem, we no longer talk

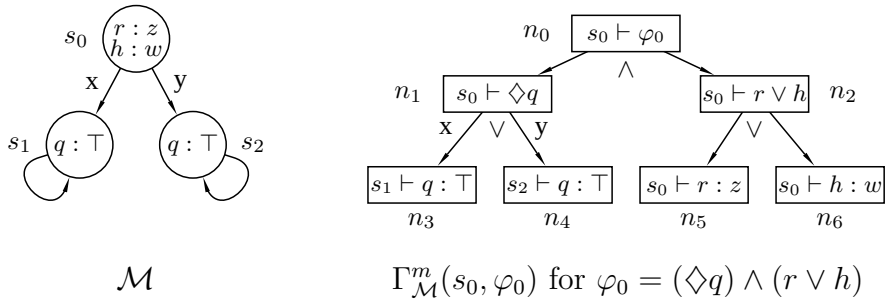

Fig. 3. Example of a Multi-Valued Game

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 ∃loise (we could dually describe the game from the point of view of ∀belard). Intuitively, we think of the value of a play as a measure for how close ∃loise is to winning. Winning of ∃loise in the 2-valued case now corresponds to the *top* value. Winning of ∀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 ∃loise is to maximize the resulting value, whereas the goal of ∀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*, divided to $\vee$-nodes, where ∃loise plays, versus $\wedge$-nodes, where ∀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 its value. 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 2 *Consider the Kripke structure $\mathcal{M}$ of Fig. 3 over lattice $\mathcal{L}$, s.t. $x, y, z, w \in \mathcal{L}$. 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 $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$, where $\varphi_0 = (\Diamond q) \wedge (r \vee h)$. Again, the edges are labelled by their values. The labelling of the terminal nodes will become clear later on.*

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 ∃loise, 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.

In an infinite play, the base value is $\top$ if $\exists$loise wins it by the 2-valued conditions, and $\bot$ if $\forall$belard wins it. In a finite play, the base value is given by the value of the terminal node where the play ended.

Definition 3 *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 4 *For a play p in the game, we define its* base value*, $base(p)$, as follows. If p is finite, $base(p) = val(n)$, 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 5 *Let $p_k = n_0, n_1, \ldots, n_k$ be a finite prefix of a play, and let $x \in \mathcal{L}$ be a base value. We define $update(p_k, x)$ 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 $\exists$loise, then $val_{i-1} = \mathcal{R}(n_{i-1}, n_i) \wedge val_i$. If the player is $\forall$belard, then $val_{i-1} = \neg\mathcal{R}(n_{i-1}, n_i) \vee val_i$. Finally, we let $update(p_k, x) = val_0$.*

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 De Morgan lattice $\neg\top = \bot$).

Definition 5 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. We say that a prefix p_i of a play p is *total* if for each player, the set of values of edges used by the player in p_i is equal to the set of values used by the same player in p. Since the underlying lattice is finite, the set of values of edges used in the play by each player is finite. Thus, there always exists a *finite total* prefix of the (possibly infinite) play. 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 a 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 play p. That is, $val(p_i) = update(p_i, base(p))$. We now have the property that the value of *any* total prefix of p is the same.

Lemma 6 *Let p_i, p_j be two finite total prefixes of a play p. Then $val(p_i) = val(p_j)$.*

PROOF.

Given a finite total prefix p_i of p as above, it suffices to show that if the value of its last edge appears earlier in the prefix in the use of the same player, then removing it will not change the value. This will show that the values of all such prefixes are equal to the value of the minimal one, and in particular are equal to each other.

Let $val_1 = x_j *_j (x_{j+1} *_{j+1} (\ldots (x_{i-1} *_{i-1} (x_i *_i base(p))) \ldots))$ and $val_2 = x_j *_j (x_{j+1} *_{j+1} (\ldots (x_{i-1} *_{i-1} base(p))) \ldots))$, where $x_j, \ldots, x_i \in \mathcal{L}$ and $*_j, \ldots, *_i \in \{\vee, \wedge\}$. Since $val(p_i) = update(p_i, base(p))$, and due to the definition of $update$, it suffices to show that if $x_j = x_i$ and $*_j = *_i$ then $val_1 = val_2$. This will imply that $val(p_i) = val(p_{i-1})$, where p_{i-1} is the result of removing the last edge from p_i.

We proceed by induction on the distance $d = i - j$ that represents the distance of the last edge from the previous instance of the same value when used by the same player. The base case is when $d = 1$, meaning that $j = i + 1$.

In this case, the claim holds by associativity and the idempotence property: $x_i *_i (x_i *_i base(p)) = x_i *_i base(p)$.

For the induction step, we claim that $val_1 = (x_j *_j x_{j+1}) *_{j+1} (x_j *_j (x_{j+2} *_{j+2} (\ldots (x_{i-1} *_{i-1} (x_i *_i base(p))) \ldots)))$ and $val_2 = (x_j *_j x_{j+1}) *_{j+1} (x_j *_j (x_{j+2} *_{j+2} (\ldots (x_{i-1} *_{i-1} base(p)) \ldots)))$ (by either distributivity, if $*_j \neq *_{j+1}$, or assocaitivity and idempotence, if $*_j = *_{j+1}$). By the induction hypothesis, $x_j *_j (x_{j+2} *_{j+2} (\ldots *_{i-1} (x_{i-1}, (x_i *_i base(p))) \ldots)) = x_j *_j (x_{j+2} *_{j+2} (\ldots *_{i-1} (x_{i-1}, base(p)) \ldots))$, thus $val_1 = val_2$. □

In other words, the play has a *limit* value. This property is surprising since the sequence of values of increasingly longer prefixes is not necessarily monotonic. Lemma 6 also implies that any finite total prefix of the play is a good representative for computing this value. Intuitively, this results from the property that an instance of an edge value that is closer to the initial node, "absorbs" the effect of a further instance of the same value when used by the same player. We therefore define the value of a play as follows.

Definition 7 *For a play p, $val(p) = update(p_i, base(p))$, where p_i is the minimal total prefix of p.*

Note that Definition 7 applies to both finite and infinite plays. For a finite play p, it results in the same value as $update(p, base(p))$.

Example 8 *Consider again the game described in Fig. 3. Terminal nodes in the game-graph are labelled by their values. For example, $n_5 = s_0 \vdash r$ is labelled by $s_0 \vdash r : z$ to indicate that $val(s_0 \vdash r) = z$. One possible play in the game is $\langle n_0, n_1, n_3 \rangle$. Its value is $\neg\top \vee (x \wedge \top) = x$. Another example is the play $\langle n_0, n_2, n_5 \rangle$ whose value is $\neg\top \vee (\top \wedge z) = z$. More plays exist.*

In this example, all the plays are finite. However, to demonstrate the value of an infinite play, suppose that the node n_3 is an $\wedge$-node that has an outgoing edge to n_1 with value x. This results in an infinite play $\langle n_0, n_1, n_3, n_1, \ldots \rangle$. Note that such an infinite play cannot occur in a real model checking game, since it does not contain any fixpoint variable. We only use it for demonstration purposes. Then, the prefix $\langle n_0, n_1, n_3, n_1 \rangle$ is a minimal total prefix of the infinite play $\langle n_0, n_1, n_3, n_1, \ldots \rangle$. Note that $\langle n_0, n_1, n_3 \rangle$ is not a total prefix since it does not contain an edge with value x which is used by $\forall$belard. Therefore, the value of the infinite play is $\neg\top \vee (x \wedge (\neg x \vee b))$, where b is the base value that depends on the winner of the infinite play in the 2-valued game. If the winner is $\exists$loise then $b = \top$ and the value of the play is $\neg\top \vee (x \wedge (\neg x \vee \top)) = x$. If the winner is $\forall$belard then $b = \bot$ and the value of the play is $\neg\top \vee (x \wedge (\neg x \vee \bot)) = x \wedge \neg x$. Note that $x \wedge \neg x$ is not necessarily equal to $\bot$ (e.g., in L_3, $U \wedge \neg U = U$).

As always, to relate the game to model checking, we need to talk about strategies, rather than a single play. In the 2-valued game, we talked about *winning* strategies, which ensure winning of the player, and we were guaranteed that exactly one player 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. We, therefore, need to replace the notion of winning strategies by 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 strategy which is *best*. 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 2-valued 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 a given play depends on the choices of ∀belard. Since we take the point of view of ∃loise and her goal is to achieve a high value, 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 that value. We choose the *greatest* possible lower bound, which characterizes the strategy as precisely as possible.

Definition 9 *For a strategy $\sigma_\exists$ for ∃loise,*

$$val(\sigma_\exists) = \bigwedge\{val(p) \mid p \text{ is a play by } \sigma_\exists\}.$$

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 possible that there does *not* exist a play with value $val(\sigma_\exists)$ by this strategy, but instead every such play has a strictly greater value. Still, the strategy cannot ensure a strictly better (greater) 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 strategy to use, and her goal is to maximize the resulting value. 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 implies that ∀belard cannot enforce any value which is strictly lower than that value. Thus, it characterizes most precisely the value that ∃loise can enforce in the game. This value is defined to be the value of the game.

Definition 10 *Let $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ be a multi-valued game. Then*

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

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 11 *In the game of Fig. 3 ∃loise has two possible moves from n_1 and n_2 (the nodes where she moves). She thus 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: $\langle n_0, n_1, n_3\rangle$ (when ∀belard chooses the left successor of n_0) and $\langle n_0, n_2, n_5\rangle$ (when ∀belard chooses the right successor of n_0) whose values are x and z respectively (see Example 8). 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 $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = val(\sigma_1) \vee val(\sigma_2) \vee val(\sigma_3) \vee val(\sigma_4) = (x \wedge z) \vee (x \wedge w) \vee (y \wedge z) \vee (y \wedge w)$. If all the latter values are incomparable, then ∃loise does not *have a unique* best *strategy. By distributivity, $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = (x \wedge (z \vee w)) \vee (y \wedge (z \vee w)) = (x \vee y) \wedge (z \vee w)$. 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 14 in the following section).*

Remark 12 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 replace the $\exists$ and $\forall$ quantifiers by the lub and glb operators respectively, since there are no longer *best* strategies for ∃loise and ∀belard. A similar phenomenon happens when considering probabilistic games [25], 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. There also, the $\exists$ and $\forall$ quantifiers are replaced by *supremum* and *infimum* respectively.

Remark 13 The discussion so far was from the point of view of ∃loise. We talked about strategies for ∃loise, their values and the value that ∃loise achieves in a game. We can define the dual notions for ∀belard. Unlike ∃loise, who is the verifier and whose goal is to gain as high a value as possible, the goal of ∀belard is the opposite. Therefore, all the definitions are dual. The value $val(\sigma_\forall)$ of a strategy $\sigma_\forall$ for ∀belard is the *lub* of the set of values of plays by the strategy, with the meaning that it ensures a value $\leq val(\sigma_\forall)$ (instead of $\geq val(\sigma_\exists)$ for ∃loise). The value $val_\forall(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$ of the game from ∀belard's point of view is the *glb* of the set of values that ∀belard has strategies for, with the meaning that ∃loise cannot enforce a value which is strictly greater than $val_\forall(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$. By the distributivity of the underlying lattice of $\mathcal{M}$, we have the desirable property that both definitions result in the same value for the game. That is, $val_\exists(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = val_\forall(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$, where $val_\exists(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$ denotes the value of the game as defined in Definition 10, from ∃loise's point of view.

3.3 Correctness

Theorem 14 *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 Theorem 14, we first give an alternative characterization for $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$, which mainly results from Birkhoff representation theorem for finite distributive lattices, implying that every element of $\mathcal{L}$ can be described as the lub of join-irreducible elements [26].

Lemma 15 *Let $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ be a multi-valued game. Then*

$$\begin{aligned} val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) &= \bigvee\{\alpha \mid \exists \textit{loise has a strategy } \sigma_\exists \textit{ with } val(\sigma_\exists) \geq \alpha\} \\ &= \bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid \exists \textit{loise has a strategy } \sigma_\exists \textit{ with } val(\sigma_\exists) \geq \alpha\}. \end{aligned}$$

PROOF. Consider the first equality. Recall that $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = \bigvee\{\alpha \mid \exists$loise has a strategy $\sigma_\exists$ with value $val(\sigma_\exists) = \alpha\}$ (see Definition 10). Clearly we can only increase this value when considering the expression in the right hand side since there we compute the lub with respect to more α's than considered in the expression defining $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$. Namely, in the right hand side, we also consider α's such that $\exists$loise has a strategy $\sigma_\exists$ with value $val(\sigma_\exists) > \alpha$. On the other hand, when such an α is considered in the right hand side, it means that $\alpha' = val(\sigma_\exists) > \alpha$ was already considered in $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$. Thus, every α that we add in the right hand side is smaller than some α' that was already considered in $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0))$, thus the value of the lub does not increase when using the expression on the right hand side.

As for the second equality: in this case the value can only decrease from the left hand side to the right hand side since we add a limitation on the α's. Namely, we restrict to join-irreducible α's. However, by Birkhoff's representation theorem every value α can be described as the lub of join-irreducible elements. In particular, let α be some element that was considered in the left hand side. That is, $\exists$loise has a strategy $\sigma_\exists$ with value $val(\sigma_\exists) \geq \alpha$. Then each α_i which belongs to the join-irreducible elements of α is such that $val(\sigma_\exists) \geq \alpha \geq \alpha_i$, thus α_i fulfills the condition of the right hand side. This ensures that all the join-irreducible elements of α are considered in the right hand side and their lub gives us α. Thus the value does not decrease when using the expression on the right hand side. □

We now use similar techniques to those used in the reduction approach of [14]. There, the multi-valued model checking problem of $\mathcal{M}$ is reduced to several 2-valued model checking problems. Namely, they define a reduced model $\mathcal{M}_\alpha$ for each join-irreducible element $\alpha \in \mathcal{J}(\mathcal{L})$, such that the multi-valued model checking result can be deduced from the 2-valued model checking of the reduced models. We consider the 2-valued model checking games over these reduced models $\mathcal{M}_\alpha$ and show a correspondence between winning strategies in these games and strategies in the multi-valued game (see the proof of Lemma 18 below). This enables us to relate the multi-valued model checking result to the value of the multi-valued game, via the reduced models and their 2-valued model checking games. Note, however, that the reductions are only part of the proof scheme, and are not used in the game definition.

In [14], in order to avoid a technical problem with negated atomic propositions,

the formula is first 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 16 *[27] 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* $\tilde{\Theta} : \tilde{\mathcal{S}} \to (\mathcal{P} \to L_3)$*.*

Specifically, given an element $\alpha \in \mathcal{J}(\mathcal{L})$, a reduced KMTS $\mathcal{M}_\alpha$ is defined by setting

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

Note that in this definition of the KMTS $\mathcal{M}_\alpha$ the value of $\Theta_\alpha(s)(q)$ is in fact defined over L_2. The formula is then interpreted over the KMTS $\mathcal{M}_\alpha$ w.r.t. a *2-valued* semantics, rather than a 3-valued semantics, with the main difference being that

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

It then holds that $(\mathcal{M}_\alpha, s_0) \models \varphi_0 \Leftrightarrow [\![\varphi_0]\!]^{\mathcal{M}}(s_0) \geq \alpha$ [14], and the following is implied.

Lemma 17 *[14]* $[\![\varphi_0]\!]^{\mathcal{M}}(s_0) = \bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid (\mathcal{M}\alpha, s_0) \models \varphi_0\}$*.*

Now, to prove the correctness of our multi-valued game we combine Lemmas 15 and 17 with the following lemma. Together they imply $val(\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)) = [\![\varphi_0]\!]^{\mathcal{M}}(s_0)$.

Lemma 18 $\bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid \exists$*loise has a strategy* $\sigma_\exists$ *with* $val(\sigma_\exists) \geq \alpha\} = \bigvee\{\alpha \in \mathcal{J}(\mathcal{L}) \mid (\mathcal{M}\alpha, s_0) \models \varphi_0\}$*.*

PROOF. We refer to the 2-valued game for KMTSs, defined in [16]. This game 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 are 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 ∃loise has a winning strategy in the 2-valued game over $\mathcal{M}_\alpha$ iff $(\mathcal{M}_\alpha, s_0) \models \varphi_0$.

To prove Lemma 18 we show a 1-1 correspondence between strategies of ∃loise with value $\geq \alpha$ in the multi-valued game over $\mathcal{M}$ and winning strategies for ∃loise in the 2-valued game over $\mathcal{M}_\alpha$, for $\alpha \in \mathcal{J}(\mathcal{L})$. We use the following property of join-irreducible elements of a distributive lattice.

Lemma 19 *[26] Let $\mathcal{L}$ be a distributive lattice. If $\alpha \in \mathcal{J}(\mathcal{L})$ and $y, z \in \mathcal{L}$, then $y \vee z \geq \alpha$ iff $y \geq \alpha$ or $z \geq \alpha$.*

Now, for $\alpha \in \mathcal{J}(\mathcal{L})$, we show that ∃loise has a strategy with value $\geq \alpha$ in the multi-valued game over $\mathcal{M}$ iff she has a winning strategy in the 2-valued game over $\mathcal{M}_\alpha$.

($\Rightarrow$) Let σ be a strategy for ∃loise with $val(\sigma) \geq \alpha$. We show that the same strategy is winning in the 2-valued game. Consider a play played by σ in the 2-valued case. We show that ∃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 ∀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 ∀belard can increase the value and by Lemma 19 they cannot increase a base value of $\bot$ to be $\geq \alpha$, since α is a join-irreducible element. Since the base value is $\top$, we conclude that the play fulfills the winning criteria of ∃loise in the 2-valued game.

If the play is finite (in the 2-valued case), we first rule out the possibility that ∃loise is stuck. If ∃loise is stuck it means that the strategy defines for her to use an edge with 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 ∀belard with value y s.t. $\neg y \geq \alpha$, but such an edge does not exist as a may-edge in the 2-valued play, which leads to contradiction. Thus, either ∀belard gets stuck, in which case ∃loise wins, or the play ends in a terminal node of the form $n = s \vdash q$. In the latter case, we again conclude by the same reasoning 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 ∃loise wins.

($\Leftarrow$) For the other direction, let σ be a winning strategy for ∃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 ∃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 $\top$ if it is infinite, and some value $\geq \alpha$ if it is finite (due to the winning conditions in the 2-valued game and the definition of the labelling function Θ_α of $\mathcal{M}_\alpha$). 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 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 this implies 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. □

Remark 20 We assumed that the transition relation of $\mathcal{M}$ is total, in the sense that for each $s \in \mathcal{S}$ there exists some state $s' \in \mathcal{S}$ with $\mathcal{R}(s, s') \neq \bot$. If this assumption is removed, then the game requires simple adaptations. Namely, if a state s has no successor state $s' \in \mathcal{S}$ with $\mathcal{R}(s, s') \neq \bot$, then nodes of the form $s \vdash \Diamond\varphi$ and $s \vdash \Box\varphi$ become terminal nodes, meaning that a finite play can now end in such a node. Therefore, to handle transition relations which are not total, we extend Definition 3 to such terminal nodes. For a terminal node $n = s \vdash \Diamond\varphi$, we define $val(n)$ to be $\bot$. For $n = s \vdash \Box\varphi$, we define $val(n)$ to be $\top$. This is consistent with the semantics, which collapses in this case to $[\![\Diamond\varphi]\!]_\rho(s) = \bigvee\{\mathcal{R}(s, s') \wedge [\![\varphi]\!]_\rho(s') \mid \mathcal{R}(s, s') \neq \bot\} = \bigvee \emptyset = \bot$, and $[\![\Box\varphi]\!]_\rho(s) = \bigwedge\{\neg\mathcal{R}(s, s') \vee [\![\varphi]\!]_\rho(s') \mid \mathcal{R}(s, s') \neq \bot\} = \bigwedge \emptyset = \top$ (for any environment ρ). No further adaptation is required to the definitions of the values of plays, strategies and the game. Moreover, the relation of the game to the multi-valued model checking problem, as formalized by Theorem 14, is maintained.

4 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 relied on the finite nature of the lattice. In this section, we suggest a different way of overcoming the problem. The new definition makes the value of a play much simpler to define.

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

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

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

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

The idea is to split each 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 value of 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. That is, we reduce the multi-valued edges into more multi-valued terminal nodes. We emphasize that the reduction is performed in the game level, rather than the model level. The underlying Kripke structure still has multi-valued transitions.

Formally, given a Kripke structure $\mathcal{M}$ over lattice $\mathcal{L}$, a state s_0 and a μ-calculus formula φ_0, we define $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ (where d stands for *definite* edges) 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 $(s,t) \vdash \bot$ that 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 (s,t) that is supposed to show evidence for $\Diamond\varphi$, and proceeds to the configuration $(s,t) \vdash \Diamond\varphi$. 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: $(s,t) \vdash \top$, which stands for examining the value of the transition (see below), or $t \vdash \varphi$, which stands for continuing the play as usual. 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 $\exists$loise tries to prove the existential property. Dually, a configuration of the form $(s,t) \vdash \bot$ is reached when $\exists$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 21 *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 4) – no update is needed. The rest of the definitions of strategies, their values and the value of the game remain unchanged. Theorem 22 ensures that we get the same value in the new game, thus Theorem 14 regarding the correctness of the game is maintained.

Theorem 22 *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 $val(\Gamma^d_{\mathcal{M}}(s_0,\varphi_0)) = val(\Gamma^m_{\mathcal{M}}(s_0,\varphi_0))$.*

PROOF.

We use the alternative definition of the value of the game, as given in Lemma 15, and show a 1-1 correspondence between strategies of ∃loise with value $\geq \alpha$ in both games, for α which is a join irreducible element of $\mathcal{L}$. This implies that the value of the game in both versions, computed by the join of all these α's, is equal.

($\Rightarrow$) From a strategy σ for ∃loise in $\Gamma^m_{\mathcal{M}}(s_0,\varphi_0)$ with value $\geq \alpha$ we construct a strategy σ' with value $\geq \alpha$ in the second version. The strategy σ' for $\Gamma^d_{\mathcal{M}}(s_0,\varphi_0)$ is the same as σ with the following addition. In $\Gamma^d_{\mathcal{M}}(s_0,\varphi_0)$, when ∀belard moves from some configuration of the form $s \vdash \Box\varphi$ to $(s,t) \vdash \Box\varphi$, ∃loise now has to choose whether she challenges the value of the transition (s,t) or the rest of the play. If $\neg\mathcal{R}(s,t) \geq \alpha$, she will choose the transition. Otherwise, she will choose to continue the play.

We now show that the value of σ' in $\Gamma^d_{\mathcal{M}}(s_0,\varphi_0)$ is $\geq \alpha$. It suffices to show that the value of every play in the game where ∃loise plays by σ' is $\geq \alpha$. Suppose to the contrary that there exists a play p with value $\not\geq \alpha$ in $\Gamma^d_{\mathcal{M}}(s_0,\varphi_0)$ where ∃loise plays by σ'.

If p is infinite, then its value in $\Gamma^d_{\mathcal{M}}(s_0,\varphi_0)$ is either $\top$ or $\bot$. Thus for the value to be $\not\geq \alpha$, it has to be $\bot$. Furthermore, we know that in every configuration of the form $(s,t) \vdash \Box\varphi$ along the play, ∃loise chose to challenge the rest of the play (otherwise the play would be finite). By the definition of σ' this implies that $\neg\mathcal{R}(s,t) \not\geq \alpha$. When looking at the same play in $\Gamma^m_{\mathcal{M}}(s_0,\varphi_0)$ (where the

moves are no longer split), this means that its value is $\not\geq \alpha$. This is because the base value is $\bot$ and it can only be increased by edges of $\forall$belard. However, as explained above their values are $\not\geq \alpha$ and thus cannot increase the value of the play to be $\geq \alpha$ (recall that α is join-irreducible, thus by Lemma 19 a value $\geq \alpha$ cannot be achieved if all values are $\not\geq \alpha$).

If p is finite, then it ends in a terminal node with value $val \not\geq \alpha$. Clearly, it is not one of the new terminal nodes of the form $(s,t) \vdash \bot$ because σ' only defines to use them when their value is $\geq \alpha$. If the terminal node is of the form $(s,t) \vdash \top$, then in the game $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$, p is a prefix of a longer play whose suffix has value $\leq val$ and thus $\not\geq \alpha$. By the same reasoning as before its prefix cannot increase the value of the play to be $\geq \alpha$. The last possibility is that p ends in a terminal node which is not one of the new ones and has value $\not\geq \alpha$. When looking at p in $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$, this makes its base value $\not\geq \alpha$ and again, this value cannot be increased to become $\geq \alpha$.

In any case, we get a play in $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$, where $\exists$loise plays by σ and the value is $\not\geq \alpha$, in contradiction.

($\Leftarrow$) A strategy σ for $\exists$loise in $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ whose value is $\geq \alpha$ is in itself a strategy that achieves a value $\geq \alpha$ in the first version, when disregarding the decisions that no longer exist in this version and adding arbitrary choices when needed.

Consider a play p in $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ where $\exists$loise plays by (the adapted) σ. If the same play exists in $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ by σ then its value there is $\geq \alpha$ (we know that $val(\sigma) \geq \alpha$), which makes its base value in $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ also $\geq \alpha$. Furthermore, all the edges used by $\exists$loise along the play have values $\geq \alpha$: otherwise, in $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$, $\forall$belard could challenge the value of the corresponding transition, resulting in a play with value $\not\geq \alpha$, in contradiction. Since only edges of $\exists$loise can decrease the value of the play, we conclude that the value of p in $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ is $\geq \alpha$.

If p does not exist in $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ by σ, it has to be the case that $\forall$belard used an edge for which in $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ $\exists$loise challenges the value of the transition. This means that the transition (which determines the value of the play in this case) has a value y such that $\neg y \geq \alpha$. 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 that, and thus it remains $\geq \alpha$. $\square$

While the multi-valued game presented in this section is equivalent to the one presented in Section 3 in terms of their relation to the multi-valued model checking problem, we find that each of the games has its advantages. We therefore present both of them. On the one hand, the definition presented

in this section has the advantage that the value of a play is defined more easily. This definition also gives more insight to the effect of the multi-valued transitions, as the players can explicitly challenge their values. On the other hand, the definition presented in Section 3 has the advantage that the size of the game-graph is smaller. Moreover, the treatment of the multi-valued transitions is more direct than in the game presented in this section, where the value of a transition is "separated" from the transition itself: the play either proceeds along the transition, or proceeds to a node that reflects its value. The last two advantages of the game presented in Section 3 make it more suitable for the design of a direct model checking algorithm.

5 Solving the Multi-Valued Game

In this section, we discuss how to solve the multi-valued model checking game. We consider the game presented in Section 3, since its treatment of the multi-valued transitions is more direct and results in a smaller game-graph. We also explain how to adapt the algorithm in order to solve the game from Section 4.

Given a game $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ our purpose is to compute its value. By Theorem 14 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 as detailed in Section 1.

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. Since game-based model checking algorithms for the full μ-calculus are complicated even in the 2-valued case, 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 Q_i. We label each node $n \in Q_i$ with a value, denoted $res(n)$, as follows. For a terminal node n, $res(n) = val(n)$. 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 (i.e., contain cycles), 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 [20]. This variable is called a *witness*, as it determines the winner of the play. 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 rules described above (for $\vee$-nodes and $\wedge$-nodes) 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, where the initialization is based on the witness.

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 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. They thus improve the efficiency of the algorithm.

Theorem 23 *Let $\mathcal{M}$ be a Kripke structure over $\mathcal{L}$. Then for every state s_0 in $\mathcal{M}$ and every closed alternation-free 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 $[\![\varphi_0]\!]^{\mathcal{M}_0}(s_0) = res(s_0 \vdash \varphi_0)$.*

PROOF. [sketch] The proof is by induction on the structure of φ_0. The interesting case is when $\varphi_0 = \eta Z.\psi$. There, the correctness of the algorithm holds since the algorithm imitates the fixpoint computation of the formula. □

The algorithm for solving $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$ can also be used to solve the game $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ from Section 4. In fact, since all the edges in the game-graph of $\Gamma^d_{\mathcal{M}}(s_0, \varphi_0)$ have value $\top$, the labelling rules of the algorithm can be simplified. Namely, instead of updating the value of an $\vee$-node n to $res(n) = \bigvee\{\mathcal{R}(n, n') \wedge res(n') \mid \mathcal{R}(n, n') \neq \bot\}$, its value is simply updated to the equivalent expression $res(n) = \bigvee\{res(n') \mid \mathcal{R}(n, n') \neq \bot\}$. Similarly, if n is an $\wedge$-node then $res(n) = \bigwedge\{res(n') \mid \mathcal{R}(n, n') \neq \bot\}$ (instead of $res(n) = \bigwedge\{\neg\mathcal{R}(n, n') \vee res(n') \mid \mathcal{R}(n, n') \neq \bot\}$). A similar simplification is applicable for the temporary values as well.

Complexity. In a trivial MSCC, the value of a node is set exactly once. Thus in such MSCCs the algorithm that labels nodes by $res(n)$ traverses each edge at most once, making the running time of the algorithm in such MSCCs linear with respect to the size of the game-graph of $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$. However, in a non-trivial MSCC, the temporary value of a node might change multiple times. Each update of a temporary value $temp(n)$ requires to re-traverse the ingoing edges of n. Thus, in such MSCCs the running time is bounded by the size of the MSCC multiplied by the number of times the temporary value $temp(n)$ of some node n can change. It is easy to verify that the temporary values are updated monotonically: in a MSCC with a ν-witness the values are monotonically decreasing, whereas in a MSCC with a μ-witness the values are monotonically increasing. In particular, the temporary value of a node never changes to some value which is incomparable to its previous value. These arguments can be proved by induction on the updates of the temporary values. The monotonicity property ensures that the temporary value of each node can be updated at most $h(\mathcal{L})$ times, where $h(\mathcal{L})$ is the height of the lattice. Therefore, the total running time of the algorithm is linear with respect to the size of the game-graph of $\Gamma^m_{\mathcal{M}}(s_0, \varphi_0)$, and with respect to the height of the lattice. The size of the game-graph is bounded by the size of the underlying Kripke structure times the length of the formula, thus the overall complexity is $O(|\mathcal{M}| \cdot |\varphi_0| \cdot h(\mathcal{L}))$.

The resulting complexity is comparable to the complexity of other multi-valued model checking algorithms for the alternation-free μ-calculus. For example, the direct algorithm of [14], which is based on extended alternating automata, has the same complexity. The reduction approach of [14] yields an algorithm which is linear in the number of join-irreducible elements of the lattice, $|\mathcal{J}(\mathcal{L})|$, rather than in $h(\mathcal{L})$. Similarly, the complexity of the symbolic algorithm of [18] for CTL is $O(|\mathcal{S}| \cdot |\mathcal{M}| \cdot |\varphi_0| \cdot |\mathcal{J}(\mathcal{L})|)$.

Strategies. As explained in Section 3, ∃loise does not always have a best strategy in the multi-valued game. However, if the underlying lattice of $\mathcal{M}$ has a total order, then such a best strategy exists. In this case, the algorithm for solving the game can also be used to generate such a strategy for ∃loise [2]. For each $\vee$-node n, the algorithm will keep track of the successor of n which "determined" $res(n)$. For this purpose, whenever the value of n (either $res(n)$ or $temp(n)$) is updated, the algorithm will update $strategy(n)$ to the node n' such that $\mathcal{R}(n, n') \wedge res(n') = \bigvee\{\mathcal{R}(n, n') \wedge res(n') \mid \mathcal{R}(n, n') \neq \bot\}$ or $\mathcal{R}(n, n') \wedge temp(n') = \bigvee\{\mathcal{R}(n, n') \wedge temp(n') \mid \mathcal{R}(n, n') \neq \bot\}$, resp. Note that if the lattice has a total order, then such n' always exists. The final values of $strategy(n)$ for all $\vee$-nodes define the best strategy for ∃loise.

[2] The algorithm for solving the multi-valued game can also be used to generate strategies for ∀belard, by keeping track of the updates of $\wedge$-nodes.

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 [14], the same problem was considered from the automata-theoretic point of view. In this section, we discuss the essential difference between the two approaches in the multi-valued case.

In the automata-theoretic approach, model checking is performed by checking nonemptiness of a (word) automaton that represents the product of the model and the checked formula. Similarly to the nodes of the model checking game-graph, each state of the automaton represents a state of the model and a subformula. It is well-known that the game-based and the automata-based approaches are closely related in the 2-valued setting: an accepting run corresponds to a winning strategy for ∃loise and vice versa [23]. Surprisingly, the same relation does *not* hold anymore in the multi-valued case.

More specifically, in [14], *extended alternating automata* (*EAAs*) were used as the basis for multi-valued model checking. We briefly describe their approach in order to set a basis for comparison. The formal definitions appear in [14].

Recall that in a (word) alternating automaton the transition relation δ defines for each automaton state a and each input letter l a positive boolean formula over the states of the automaton. It has the following meaning. Assume that in a run of the automaton on an input word w, the automaton is in the state a and reads the position n of w, labeled by $l(n)$. Then the automaton has to send copies to the successor of n in w, each copy with some automaton state, such that the chosen successor automaton states satisfy the formula $\delta(a, l(n))$. A run of the alternating automaton on an input word w is then a tree, where each node is associated with a state a of the automaton and a position n of the input word w, with the meaning that the node (a, n) represents a copy of the automaton that reads the position n of w in the state a of the automaton. The successors of a node (a, n) in the run tree represent the copies that the automaton sent to the successor of n in w. As such, the automaton states that appear in the successors of the node (a, n) in the run tree need to satisfy the formula $\delta(a, l(n))$ defining the transition relation of a when it reads n. If all the branches of the run tree satisfy the acceptance conditions, then the run is *accepting*.

An EAA [14] [3] is defined similarly to an alternating automaton, except that the formulas defining its transition relation can contain constants which are elements from a lattice $\mathcal{L}$. In addition, to capture the multi-valued nature, [14] associated each run of an EAA, and in particular an accepting run, with a

[3] [14] defined EAAs over trees. We describe only the case of a word EAA since for the purpose of model checking a word automaton is constructed.

value, which is an element from $\mathcal{L}$.

Namely, a run r of an EAA on an input word w is a tree in which each node (a, n) is also labelled by a value x from $\mathcal{L}$, where now the values of the successors of a node (a, n, x) in the run tree must yield the value x when they substitute the corresponding automaton states in the formula $\delta(a, l(n))$. In this sense, the *values* of the nodes in the run must obey the transition relation. The value of the run, $val(r)$, is the value associated with its root.

[14] showed that there always exists an accepting run of the EAA with a *maximum* value. This maximum value defines the value of the emptiness of the automaton. When the EAA represents the product of the checked formula with a Kripke structure $\mathcal{M}$ over $\mathcal{L}$, this value defines the multi-valued model checking result.

In the multi-valued game, on the other hand, it is *not* necessarily the case that there exists a strategy of ∃loise with a maximum value. Instead, the value of the game, which determines the model checking result, is a combination of incomparable strategies.

Intuitively speaking, the difference between strategies and runs results from the fact that a strategy defines for each node in the game-graph a *single* node that the player should proceed to. This is as opposed to a run of the automaton, where from an automaton state (which corresponds to a node in the game-graph) the automaton can proceed with *several* automaton states simultaneously by using several copies of the automaton. In a regular alternating automaton, an accepting run can always be translated into an accepting run where the automaton proceeds with a single copy in $\vee$-choices (which correspond to $\vee$-nodes in the game-graph). This ensures the 1-1 correspondence between strategies of ∃loise and accepting runs in the 2-valued case. However, such a translation that maintains the value of the run does not exist in the multi-valued case.

Still, the value of the multi-valued model checking game is equal to the emptiness value of the EAA, given by $val(r_{max})$, where r_{max} is an accepting run of the EAA with the maximum value. This is because both values are equal to the model checking result. By Lemma 15, this ensures that ∃loise has a strategy with value $\geq \alpha$ for each join-irreducible element $\alpha \leq val(r_{max})$. More generally, it can be shown that an accepting run r of the EAA with value $val(r)$ corresponds to a collection of strategies for ∃loise: a strategy with value $\geq \alpha$ for each join-irreducible element $\alpha \leq val(r)$. Furthermore, these strategies can be extracted from the run r. However, the 1-1 correspondence between runs and strategies that existed in the 2-valued setting no longer exists. This demonstrates the discrepancy between automata and games in the multi-valued setting.

It is possible to regain the relation between the game-theoretic approach and 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 n with $bet(n) \neq \bot$ she proceeds by associating (possibly a subset) of its successors with a bet (a value from the lattice) 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 $bet(n) = \bigwedge\{\neg\mathcal{R}(n,n') \vee bet(n') \mid \mathcal{R}(n,n') \neq \bot\}$. Note that the bets of ∃loise can be any value from the lattice, including $\bot$. If ∃loise cannot proceed while fulfilling the rules, then the play terminates. Otherwise, the role of ∀belard is 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, and will try to make ∃loise contradict herself.

In this definition we return to talking about *winning* versus losing. Intuitively, ∃loise wins if she manages to proceed without contradictions. Formally, if the play terminates since ∃loise cannot proceed while fulfilling the above rules, then she loses. If a node with $bet(n) = \bot$ is reached then ∃loise wins (as there is nothing left to prove). If the play ends in a terminal node 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 [14] as it handles the multi-valued transitions of the Kripke 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 [4,5] the regular model checking game has been generalized to a 3-valued game over a KMTS (see Definition 16). 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 [5] 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, it is possible to give the indefinite value U an intuitive meaning of a *tie*. We can thus still talk about the notion of *winning* in a way that corresponds to the three possible values $\{\top, U, \bot\}$ in the logic (see L_3 in Fig. 1) [5]. 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. The correspondence between the value of the game and the formula is then given by a variant of Theorem 1, with an additional possibility [5]:

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

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, the value of a strategy in the 3-valued case is determined by the value of the "worst" play (rather than a lower bound), as such a play always exists, 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 [5] 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, because of the total order on the elements of the lattice, 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). They are therefore 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.

8 Conclusion

This work suggests a characterization of the multi-valued model checking problem of the μ-calculus in terms of two players games: a verifier and a falsifier. We suggest two multi-valued games in which the players hold on to their traditional roles, as in the 2-valued case. However, the concept of *winning* is replaced by the *value* of the game: the model checking result is determined by the value of the game rather than by the winner. The two games differ in their treatment of the multi-valued transitions of the Kripke structure.

We also suggest a third game, of a different nature, where the roles of the players are altered. In this game, the concept of winning in reintroduced, but it is still accompanied with a value. The model checking result is determined by the maximum value for which the verifier has a winning strategy in the game. This game, while not being a natural generalization of the 2-valued case as the other two games, regains the resemblance to the automata-based approach, which does not exist for the first two.

To demonstrate the applicability of our work, we derive from the game-based characterization a *direct* multi-valued model checking algorithm for the alternation free fragment of the μ-calculus, that handles the multi-valued structure without any reduction. Such an algorithm can be used as a basis for exploiting many of the advantages of the game-based approach to model checking.

References

[1] S. Shoham, O. Grumberg, Multi-valued model checking games, in: Third International Symposium on Automated Technology for Verification and Analysis (ATVA), Vol. 3707 of LNCS, 2005, pp. 354–369.

[2] E. Clarke, O. Grumberg, D. Peled, Model Checking, MIT press, 1999.

[3] G. Bruns, P. Godefroid, Model checking partial state spaces with 3-valued temporal logics, in: Computer Aided Verification, 1999, pp. 274–287.

[4] S. Shoham, O. Grumberg, A game-based framework for CTL counterexamples and 3-valued abstraction-refinement, in: Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03), Vol. 2725 of LNCS, Springer, Boulder, CO, USA, 2003, pp. 275–287, to appear in TOCL.

[5] O. Grumberg, M. Lange, M. Leucker, S. Shoham, Don't know in the μ-calculus, in: 6th international conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05), Vol. 3385 of LNCS, Paris, France, 2005, pp. 233–249.

[6] S. M. Easterbrook, M. Chechik, A framework for multi-valued reasoning over inconsistent viewpoints, in: ICSE, 2001, pp. 411–420.

[7] M. Huth, S. Pradhan, Lifting assertion and consistency checkers from single to multiple viewpoints, Tech. Rep. 2002/11, Department of Computing, Imperial College, London (2002).

[8] W. Chan, Temporal-logic queries, in: CAV, Vol. 1855 of LNCS, Springer-Verlag, 2000, pp. 450–463.

[9] G. Bruns, P. Godefroid, Temporal logic query checking, in: LICS, IEEE, 2001, pp. 409–417.

[10] A. Gurfinkel, B. Devereux, M. Chechik, Model exploration with temporal logic query checking, in: FSE, ACM, 2002, pp. 139–148.

[11] P. Godefroid, R. Jagadeesan, Automatic abstraction using generalized model checking, in: Proc. of Conference on Computer-Aided Verification (CAV), Vol. 2404 of LNCS, Springer-Verlag, Copenhagen, Denmark, 2002, pp. 137–150.

[12] B. Konikowska, W. Penczek, Reducing model checking from multi-valued CTL* to CTL*, in: CONCUR, Vol. 2421 of LNCS, 2002.

[13] A. Gurfinkel, M. Chechik, Multi-valued model checking via classical model checking, in: CONCUR, 2003, pp. 263–277.

[14] G. Bruns, P. Godefroid, Model checking with multi-valued logics, in: ICALP, 2004.

[15] B. Konikowska, W. Penczek, Model checking multi-valued modal mu-calculus: Revisited, in: Proc. of CS&P'04, 2004.

[16] S. Shoham, A game-based framework for CTL counterexamples and abstraction-refinement, Master's thesis, Department of Computer Science, Technion - Israel Institute of Technology (2003).

[17] M. Chechik, B. Devereux, A. Gurfinkel, Model-checking infinite state-space systems with fine-grained abstractions using spin, in: SPIN Workshop, Vol. 2057 of LNCS, Springer-Verlag, 2001.

[18] M. Chechik, B. Devereux, A. Gurfinkel, S. Easterbrook, Multi-valued symbolic model-checking, Tech. Rep. CSRG-448, University of Toronto (April 2002).

[19] D. Kozen, Results on the propositional μ-calculus, TCS 27.

[20] C. Stirling, Local model checking games, in: Proceedings of the 6th International Conference on Concurrency Theory (CONCUR'95), Vol. 962 of LNCS, Springer, Berlin, Germany, 1995, pp. 1–11.

[21] C. Stirling, D. Walker, Local model checking in the modal μ-calculus, Theoretical Computer Science 89 (1) (1991) 161–177.

[22] O. Kupferman, M. Vardi, P. Wolper, An automata-theoretic approach to branching-time model checking, Journal of the ACM (JACM) 47 (2) (2000) 312–360.

[23] M. Leucker, Model checking games for the alternation free mu-calculus and alternating automata, in: 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), 1999.

[24] A. Tarski, A lattice-theoretical fixpoint theorem and its application, Pacific J.Math. 5 (1955) 285–309.

[25] L. de Alfaro, R. Majumdar, Quantitative solution of omega-regular games, in: STOC, 2001, pp. 675–683.

[26] B. Davey, H. Priestly, Introduction to Lattices and Order, Cambridge University Press, 1990.

[27] M. Huth, R. Jagadeesan, D. Schmidt, Modal transition systems: A foundation for three-valued program analysis, in: European Symposium on Programming (ESOP'01), Vol. 2028, 2001, pp. 155–169.