# When Not Losing Is Better than Winning: Abstraction and Refinement for the Full $\mu$-Calculus

Orna Grumberg [a] Martin Lange [b] Martin Leucker [c]
Sharon Shoham [a]

[a] *Computer Science Department, The Technion, Haifa, Israel*
[b] *Department of Computer Science, Aarhus University, Denmark*
[c] *Institut für Informatik, Technical University of Munich, Germany*

**Abstract**

This work presents a novel game-based approach to abstraction-refinement for the full $\mu$-calculus, interpreted over 3-valued semantics.

A novel notion of *non-losing strategy* is introduced and exploited for refinement. Previous works on refinement in the context of 3-valued semantics require a direct algorithm for solving a 3-valued model checking game. This was necessary in order to have the information needed for refinement available on one game board. In contrast, while still considering a 3-valued model checking game, here we reduce the problem of solving the game to solving two 2-valued model checking (parity) games. In case the result is indefinite (*don't know*), the corresponding non-losing strategies, when combined, hold all the information needed for refinement. This approach is beneficial since it can use *any* solver for 2-valued parity games. Thus, it can take advantage of newly developed such algorithms with improved complexity.

## 1 Introduction

Abstraction is one of the most successful techniques for fighting the state explosion problem in model checking [1]. Abstractions hide some of the details of the verified system, thus result in a smaller model. Usually, they are designed to be *conservative* for *true*, meaning that if a formula is true of the abstract model then it is also true of the concrete (precise) model of the system. However, if it is false in the abstract model then nothing can be deduced of the concrete one.

In order to obtain more precise results, temporal logics can be interpreted over abstract models with respect to the 3-valued semantics [2–4]. This semantics evaluates a formula to either *true*, *false*, or *indefinite*. Abstract models can then be designed to be conservative for both *true* and *false*. Only if the value of a formula in the abstract model is indefinite, its value in the concrete model is unknown. In this case, a refinement is needed in order to make the abstract model more precise.

Refinement of indefinite results has been suggested for CTL in [5] and for the $\mu$-calculus in [6]. In both cases, the refinement is based on finding a cause for the indefinite result by following the run of an algorithm that solves a corresponding 3-valued model checking game. Being based on an especially tailored algorithm, a similar approach is not applicable when the 3-valued model checking game is solved via a reduction to two 2-valued model checking games.

In this work we present a novel approach, which shows that refinement information can be extracted from two 2-valued model checking games, provided that they are defined over the full game board of the 3-valued game. Our refinement is based on the new notion of *non-losing* rather then winning strategies. This approach is beneficial since it can take advantage of any game-based model checking algorithm for the $\mu$-calculus with respect to the 2-valued semantics [7,8].

We will now explain our new approach in more detail. We consider the *$\mu$-calculus* [9], which is a powerful formalism for expressing properties of transition systems using fixpoint operators. Many verification procedures can be solved by translating them into $\mu$-calculus model checking [10]. Such problems include (fair) CTL model checking, LTL model checking, bisimulation equivalence and language containment of $\omega$-regular automata.

Many algorithms for $\mu$-calculus model checking with respect to the 2-valued semantics have been suggested [11–15]. An elegant solution to this problem is the game-based approach [16], in which two players, the verifier $\exists$loise (denoted $\exists$) and the refuter $\forall$belard (denoted $\forall$), try to win a game. A formula $\varphi$ is true in a model $M$ iff the verifier has a winning strategy, meaning that she can win any play, no matter what the refuter does. The game is played on a *game board*, consisting of configurations $s \vdash \psi$, where $s$ is a state of the model $M$ and $\psi$ is a subformula of $\varphi$. The players make moves between configurations in which they try to verify or refute $\psi$ in $s$. These games can also be seen and studied as *parity games* [7,8] and we follow this approach as well.

In model checking games for the 2-valued semantics, exactly one of the players has a winning strategy, thus the model checking result is either true or false. For the 3-valued semantics, a third value should also be possible. Following [5],

we change the definition of a game for $\mu$-calculus so that a *tie* is also possible. We can now consider for each player, in addition to a winning strategy also a *non-losing* strategy, which guarantees that each play will end with either a win for this player or a tie, no matter what the other player does.

To simplify the presentation, we transform the 3-valued model checking game into an equivalent 3-valued parity game with players 0 and 1. In order to determine the winner, if there is one, we then reduce this game to two 2-valued parity games, $G_0$ and $G_1$. Player 0 has a winning strategy on game $G_0$ iff Player 0 has a winning strategy on the original 3-valued game $G$. Furthermore, Player 0 has a winning strategy on $G_1$ iff she has a non-losing strategy on $G$. The dual facts hold for Player 1.

When the game $G$ results in a tie, and a refinement is needed, non-losing strategies become extremely helpful. In this case none of the players have a winning strategy, which means that considering winning strategies does not provide a witness for the tie result. Non-losing strategies, however, take exactly this role: when the result is a tie, each player has a non-losing strategy (which corresponds to a winning strategy on one of the 2-valued games). These strategies can be combined to one play along which a cause for the tie can be found. A refinement criterion is then suggested and abstract states are refined (split) accordingly. The refinement is applied only to parts of the model from which a tie is possible. Vertices from which there is a winning strategy for one of the players are not changed. Thus, the refined abstract models do not grow unnecessarily. If the concrete model is finite then our abstraction-refinement is guaranteed to terminate with a definite result.

We note that the 3-valued model checking game still has an important role. Namely, a similar approach for refinement is not applicable when the 3-valued model checking problem itself is reduced to two 2-valued model checking problems (e.g. [4]), each solved by a separate 2-valued game. This is because then each of the 2-valued games considers a different part of the game board: one considers the part required for proving the formula, while the other considers the part required for proving its negation. This is whereas for refinement purposes it is important to consider the *full* game board of the 3-valued game (see Section 6 for more details).

**Organization of the paper** Abstract models are defined in the next section. The $\mu$-calculus with its 3-valued semantics is introduced in Section 3. In Section 4, a 3-valued model checking game for $\mu$-calculus is shown and is proved to be correct with respect to the 3-valued semantics. Section 5 presents 3-valued parity games and translates 3-valued model checking games into such games. It also suggests an algorithm for solving these games by reducing them to two 2-valued parity games. Section 6 then presents our new refinement

algorithm. We conclude in Section 7.

## 2 Abstraction

Let $\mathcal{P}$ be a set of *propositional constants*, and $\mathcal{A}$ be a set of *action names*. Every $a \in \mathcal{A}$ is associated with a so-called *must*-action $a!$ and a *may*-action $a?$. Let $\mathcal{A}! = \{a! \mid a \in \mathcal{A}\}$ and $\mathcal{A}? = \{a? \mid a \in \mathcal{A}\}$.

We use *Kripke Modal Transition Systems* (KMTS) [3,17] as abstract models that preserve satisfaction and falsification of three-valued $\mu$-calculus formulae. A KMTS is a tuple $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{x} \mid x \in \mathcal{A}! \cup \mathcal{A}?\}, L)$ where $\mathcal{S}$ is a set of states, and $\xrightarrow{x} \subseteq \mathcal{S} \times \mathcal{S}$ for each $x \in \mathcal{A}! \cup \mathcal{A}?$ is a binary relation on states, s.t. for all $a \in \mathcal{A}$: $\xrightarrow{a!} \subseteq \xrightarrow{a?}$. $L : \mathcal{S} \to \mathcal{P} \to M$ for some complete lattice $M$ assigns to each pair of states and propositions a truth value. Here we use the lattice $\mathbb{B}_3$ consisting of elements $\{\bot, ?, \top\}$ denoting falsity, uncertainty and truth respectively. They are partially ordered by $\bot \leq ? \leq \top$.

A Kripke structure in the usual sense can be regarded as a KMTS by setting $\xrightarrow{a!} = \xrightarrow{a?}$ for all $a \in \mathcal{A}$ and not distinguishing them anymore. Furthermore, its states labelling is over $\{\bot, \top\}$.

Let $\mathcal{T}_C = (\mathcal{S}_C, \{\xrightarrow{a}_C \mid a \in \mathcal{A}\}, L_C)$ be a (concrete) Kripke structure. Let $\mathcal{S}_A$ be a set of *abstract states* and $\gamma : \mathcal{S}_A \to 2^{\mathcal{S}_C}$ a total *concretization function* that maps each abstract state to the set of concrete states it represents. An abstract model, in the form of a KMTS $\mathcal{T}_A = (\mathcal{S}_A, \{\xrightarrow{x}_A \mid x \in \mathcal{A}! \cup \mathcal{A}?\}, L_A)$, can then be defined as follows.

The labelling of an abstract state is defined in accordance with the labelling of all the concrete states it represents. For $p \in \mathcal{P}$ : $L_A(s_a)(p) = \top$ $(\bot)$ only if $\forall s_c \in \gamma(s_a) : L_C(s_c)(p) = \top$ $(\bot)$. In the remaining cases $L_A(s_a)(p) = ?$.

The *may*-transitions in an abstract model are computed such that every concrete transition between two states is represented by them: for every action $a \in \mathcal{A}$, if $\exists s_c \in \gamma(s_a)$ and $\exists s'_c \in \gamma(s'_a)$ such that $s_c \xrightarrow{a}_C s'_c$, then there exists a may transition $s_a \xrightarrow{a?}_A s'_a$. Note that it is possible that there are additional may transitions as well. The *must*-transitions, on the other hand, represent concrete transitions that are common to all the concrete states that are represented by the source abstract state: a *must*-transition $s_a \xrightarrow{a!}_A s'_a$ exists only if $\forall s_c \in \gamma(s_a)\ \exists s'_c \in \gamma(s'_a)$ such that $s_c \xrightarrow{a}_C s'_c$. Note that it is possible that there are less must transitions than allowed by this rule. That is, the may and must transitions do not have to be *exact*, as long as they maintain these conditions.

**Example 1** *Consider the concrete system shown in Figure 1(a), employing a*

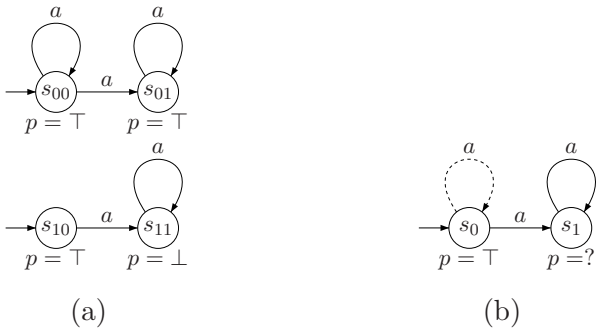



Fig. 1. (a) A concrete Kripke structure, and (b) an abstract KMTS for it

*single action a and a single proposition p. Joining* $s_{00}$ *and* $s_{10}$ *and respectively* $s_{01}$ *and* $s_{11}$ *yields the KMTS shown in Figure 1(b), where* may*-transitions are shown as dotted arrows only when no* must*-transitions are present.*

Other constructions of abstract models can be used as well. For example, if $\gamma$ is a part of a *Galois Connection* [18] ($\gamma : \mathcal{S}_A \to 2^{\mathcal{S}_C}, \alpha : 2^{\mathcal{S}_C} \to \mathcal{S}_A$) from $(2^{\mathcal{S}_C}, \subseteq)$ to $(\mathcal{S}_A, \sqsubseteq)$, then an abstract model can be constructed as described in [19] within the framework of *Abstract Interpretation* [18,20,19]. It is then not guaranteed that the must transitions are a subset of the may transitions, which complicates our further development.

## 3 The 3-Valued $\mu$-Calculus

**Syntax** We present our logic in positive normal form. Let $\mathcal{V}$ be a set of propositional variables. Formulae of the *3-valued modal* $\mu$*-calculus* in *positive normal form* are given by

$$\varphi \ ::= \ q \ \mid \ \neg q \ \mid \ Z \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \langle a \rangle \varphi \ \mid \ [a]\varphi \ \mid \ \mu Z.\varphi \ \mid \ \nu Z.\varphi$$

where $q \in \mathcal{P}$, $a \in \mathcal{A}$, and $Z \in \mathcal{V}$. Let 3–$\mathcal{L}_\mu$ denote the set of *closed* formulae generated by the above grammar, where the fixpoint quantifiers $\mu$ and $\nu$ are variable binders. We will also write $\eta$ for either $\mu$ or $\nu$. Furthermore we assume that formulae 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 $\varphi$, where the set $Sub(\varphi)$ of *subformulae* of $\varphi$ is defined in the usual way.

Given variables $Y, Z$ we write $Y \prec_\varphi Z$ if $Z$ occurs freely in $fp(Y)$ in $\varphi$, and $Y <_\varphi Z$ if $(Y, Z)$ is in the transitive closure of $\prec_\varphi$. The alternation depth $ad(\varphi)$ of $\varphi$ is the length of a maximal $<_\varphi$-chain of variables in $\varphi$ s.t. adjacent variables in this chain have different fixpoint types. A variable is called *outermost* if it is maximal w.r.t. $<_\varphi$.

**Semantics** Let $\mathbb{B}_3$ be the complete lattice consisting of elements $\{\bot, ?, \top\}$ denoting falsity, uncertainty and truth respectively, ordered by $\bot \leq ? \leq \top$. We use $\mathbb{B}_3$ not only to interpret the meaning of propositional constants in states of a KMTS but also for the semantics of formulae of the three-valued $\mu$-calculus. Negation in $\mathbb{B}_3$ is defined such that $\top$ and $\bot$ are complementary to each other (as usual), while $\bar{?} = ?$ (i.e., the complement of "don't know" is also a "don't know").

Note that $\mathbb{B}_3$ is not a boolean lattice since every finite boolean lattice is isomorphic to the subset lattice of some finite set. Hence, the size of a boolean lattice must be a power of 2. But having only 2 values deprives us of uncertainty while having at least 4 elements would ultimately force us to introduce multiple types of uncertainty. But then uncertainty would carry more information than simply "don't know". Instead, $\mathbb{B}_3$ forms a DeMorgan lattice [1] .

With logical conjunctions interpreted as meets in this lattice we obtain a seemingly strange effect: if you do not know whether $q$ holds you also do not know whether $\neg q$ holds. Hence, you do not know whether or not $q \wedge \neg q$ holds. This is desired though. The formula $q \wedge \neg q$ having truth value ? can simply be read as: we do not know whether everything inside of an abstract state is labelled with $q$ nor whether everything is labelled with $\neg q$.

Remember that the conventional $\mu$-calculus interprets formulae over the boolean lattice $(\mathbb{B}^{\mathcal{S}}, \subseteq)$ of sets of states of a transition system. Similarly, the semantics of a 3–$\mathcal{L}_\mu$ formula is an element of $\mathbb{B}_3^{\mathcal{S}}$ – the functions from $\mathcal{S}$ to $\mathbb{B}_3$ – which forms a complete (but not boolean) lattice when equipped with the following partial order: let $f, g : \mathcal{S} \to \mathbb{B}_3$. $f \sqsubseteq g$ iff $\forall s \in \mathcal{S} : f(s) \leq g(s)$. Joins and meets in this lattice are denoted by $f \sqcup g$ and $f \sqcap g$. Since we introduce formulae in positive normal form directly, we do not need to define a general negation symbol. Instead, the definition of the complement on a ground level suffices.

Then the *semantics* $[\![\varphi]\!]_\rho^{\mathcal{T}}$ of a 3–$\mathcal{L}_\mu$ formula $\varphi$ w.r.t. a KMTS $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{x} \mid x \in \mathcal{A}! \cup \mathcal{A}?\}, L)$ and an *environment* $\rho : \mathcal{V} \to \mathbb{B}_3^{\mathcal{S}}$, which explains the meaning of free variables in $\varphi$, is an element of $\mathbb{B}_3^{\mathcal{S}}$, defined as follows. We assume $\mathcal{T}$ to be fixed and do not mention it explicitly anymore. With $\rho[Z \mapsto f]$ we denote the environment that maps $Z$ to $f$ and agrees with $\rho$ on all other arguments. Later, when only closed formulae are considered, we will also drop the environment from the semantic brackets.

$$\begin{aligned} [\![q]\!]_\rho &:= \lambda s.L(s)(q) \\ [\![\neg q]\!]_\rho &:= \lambda s.\overline{L(s)(q)} \end{aligned}$$

[1] In a DeMorgan lattice every element $x$ has a unique complement $\neg x$ in the lattice such that $\neg\neg x = x$, DeMorgan's laws hold, and $x \leq y$ implies $\neg y \leq \neg x$.

$$
\begin{array}{rcl}
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\langle a \rangle \varphi]\!]_\rho & := & \lambda s. \begin{cases} \top \text{ , if } \exists t \in \mathcal{S}, \text{ s.t. } s \xrightarrow{a!} t \text{ and } [\![\varphi]\!]_\rho(t) = \top \\ \bot \text{ , if } \forall t \in \mathcal{S}, \text{ if } s \xrightarrow{a?} t \text{ then } [\![\varphi]\!]_\rho(t) = \bot \\ ? \text{ , otherwise} \end{cases} \\
[\![[a] \varphi]\!]_\rho & := & \lambda s. \begin{cases} \top \text{ , if } \forall t \in \mathcal{S}, \text{ if } s \xrightarrow{a?} t \text{ then } [\![\varphi]\!]_\rho(t) = \top \\ \bot \text{ , if } \exists t \in \mathcal{S}, \text{ s.t. } s \xrightarrow{a!} t \text{ and } [\![\varphi]\!]_\rho(t) = \bot \\ ? \text{ , otherwise} \end{cases} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap \{f \mid [\![\varphi]\!]_{\rho[Z \mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup \{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z \mapsto f]}\}
\end{array}
$$

Note that $s \xrightarrow{a!} t$ implies $s \xrightarrow{a?} t$.

The functionals $\lambda f.[\![\varphi]\!]_{\rho[Z \mapsto f]} : \mathbb{B}_3^{\mathcal{S}} \to \mathbb{B}_3^{\mathcal{S}}$ are monotone w.r.t. $\sqsubseteq$ for any $Z, \varphi$ and $\mathcal{S}$. According to [21], least and greatest fixpoints of these functionals exist.

*Approximants* of 3–$\mathcal{L}_\mu$ formulae are defined w.r.t. an environment $\rho$ in the usual way: if $fp(Z) = \mu Z.\varphi$ then $Z_\rho^0 := \lambda s.\bot$, $Z_\rho^{\alpha+1} := [\![\varphi]\!]_{\rho[Z \mapsto Z_\rho^\alpha]}$ for any ordinal $\alpha$, and $Z_\rho^\lambda := \bigsqcap_{\alpha<\lambda} Z_\rho^\alpha$ for any limit ordinal $\lambda$. Dually, if $fp(Z) = \nu Z.\varphi$ then $Z_\rho^0 := \lambda s.\top$, $Z_\rho^{\alpha+1} := [\![\varphi]\!]_{\rho[Z \mapsto Z_\rho^\alpha]}$, and $Z_\rho^\lambda := \bigsqcup_{\alpha<\lambda} Z_\rho^\alpha$.

The next theorem is a standard consequence of the Knaster-Tarski Theorem [21].

**Theorem 2** *For all KMTS $\mathcal{T}$ with state set $\mathcal{S}$, and all environments $\rho$ there is an ordinal $\alpha$ s.t. for all $s \in \mathcal{S}$ we have:*

$$\text{if } [\![\eta Z.\varphi]\!]_\rho(s) = x \text{ then } Z_\rho^\alpha(s) = x.$$

The following theorem relates the three-valued semantics of the $\mu$-calculus over an abstract KMTS with the conventional semantics over the concrete Kripke structure it represents.

**Theorem 3** [17] *Let $\mathcal{T}$ be a Kripke structure and let $\mathcal{T}'$ be a KMTS obtained from $\mathcal{T}$ with the abstraction process described in Section 2. Let $s$ be a state of*

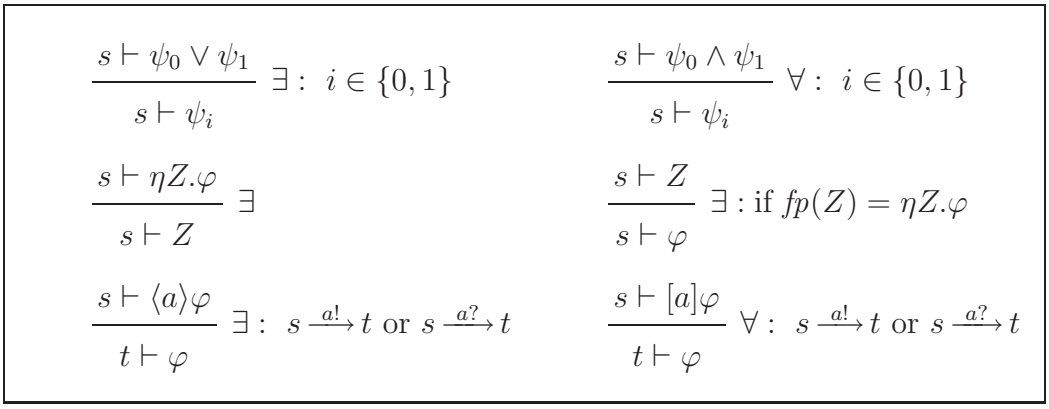



Fig. 2. The model checking game rules for 3–$\mathcal{L}_\mu$.

*$\mathcal{T}$ and $s'$ its corresponding abstract state in $\mathcal{T}'$. For all closed $\varphi \in$ 3–$\mathcal{L}_\mu$:*

$$[\![\varphi]\!]^{\mathcal{T}'}(s') \neq ?\ \textit{implies}\ [\![\varphi]\!]^{\mathcal{T}}(s) = [\![\varphi]\!]^{\mathcal{T}'}(s').$$

## 4 Model Checking Games for 3–$\mathcal{L}_\mu$

The *model checking game* $\Gamma_{\mathcal{T}}(s_0, \varphi_0)$ on a KMTS $\mathcal{T}$ with state set $\mathcal{S}$, initial state $s_0 \in \mathcal{S}$ and a 3–$\mathcal{L}_\mu$ formula $\varphi_0$ is played by players $\exists$ and $\forall$ in order to determine the truth value of $\varphi_0$ in $s_0$, cf. [22]. Configurations are elements of $\mathcal{C} \subseteq \mathcal{S} \times Sub(\varphi_0)$, and written $t \vdash \psi$. Each play of $\Gamma_{\mathcal{T}}(s_0, \varphi_0)$ is a maximal sequence of configurations that starts with $s_0 \vdash \varphi_0$. The game rules are presented in Figure 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 shown in the second line, we arbitrarily assign one player, let us say $\exists$, and call the rules *deterministic*. If no rule can be applied, the play terminates.

**Definition 4** *A player is said to play* eagerly [2] *if she or he never chooses a transition of type $\xrightarrow{a?} \setminus \xrightarrow{a!}$ for some $a \in \mathcal{A}$. A play is called* $\exists$-eager*, resp.* $\forall$-eager*, if Player $\exists$, resp. Player $\forall$, plays eagerly.*

Player $\exists$ *wins* an $\exists$-eager play $C_0, C_1, \ldots$ iff

(1) there is an $n \in \mathbb{N}$, s.t. $C_n = t \vdash q$ with $L(t)(q) = \top$ or $C_n = t \vdash \neg q$ with $L(t)(q) = \bot$, or
(2) there is an $n \in \mathbb{N}$, s.t. $C_n = t \vdash [a]\psi$ and there is no $t' \in \mathcal{S}$ s.t. $t \xrightarrow{a?} t'$, or
(3) the outermost variable that occurs infinitely often is of type $\nu$.

[2] The notion of 'eagerness' replaces the notion of 'consistency' from [5,6].

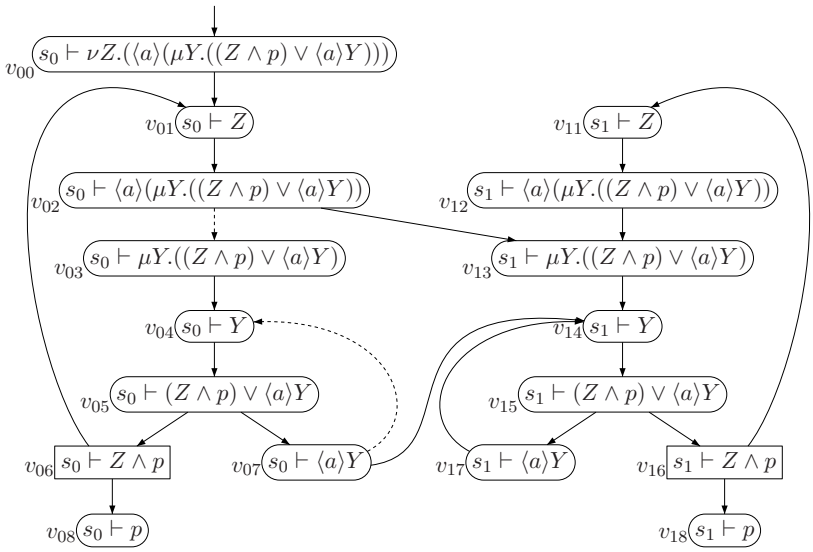



Fig. 3. All possible moves in the game over the KMTS from Figure 1(b)

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

(4) there is an $n \in \mathbb{N}$, s.t. $C_n = t \vdash q$ with $L(t)(q) = \bot$ or $C_n = t \vdash \neg q$ with $L(t)(q) = \top$, or
(5) there is an $n \in \mathbb{N}$, s.t. $C_n = t \vdash \langle a\rangle\psi$ and there is no $t' \in \mathcal{S}$ s.t. $t \xrightarrow{a?} t'$, or
(6) the outermost variable that occurs infinitely often is of type $\mu$.

In all other cases, the result of the play is a *tie*.

**Example 5** *For the model checking problem of the formula*

$$\nu Z.(\langle a\rangle(\mu Y.((Z \wedge p) \vee \langle a\rangle Y)))$$

*in the state $s_0$ of the abstract KMTS from Figure 1(b) all possible moves are shown in Figure 3. Configurations in which $\exists$ is to choose are drawn as circles while $\forall$-configurations are shown as squares. Moves based on* may*-transitions are not shown when the same move is possible using a* must*-transition.*

*The result of the play $v_{00}v_{01}v_{02}v_{03}v_{04}v_{05}v_{06}v_{08}$ is a tie: while the play ends in a configuration in which $p$ evaluates to $\top$, $\exists$ does not win since choosing the edge from $v_{02}$ to $v_{03}$ violates $\exists$-eagerness. $v_{00}v_{01}v_{02}v_{13}v_{14}v_{15}v_{16}v_{11}v_{12}v_{13}\ldots$, on the other hand, is an $\exists$-eager play in which the outermost variable occurring infinitely often is of type $\nu$. Thus, it is won by $\exists$.*

**Definition 6** *The* truth value *of a configuration $t \vdash \psi$ in the context of $\rho$ is the value of $[\![\psi]\!]_\rho(t)$. The value $\top$* improves *both ? and $\bot$, while ? only improves $\bot$. On the other hand, $x$* worsens *$y$ iff $y$ improves $x$. A configuration with truth value $x$ under the environment $\rho$ will also be called an $x_\rho$-configuration. We say that a move, i.e. an application of a game rule between configurations*

*$C$ and $C'$ is a $x_\rho$-$y_{\rho'}$-*improvement *if $C$ is an $x_\rho$-configuration, $C'$ is an $y_{\rho'}$-configuration and $y$ improves $x$. Similarly we define a $x_\rho$-$y_{\rho'}$-*worsening *and a $x_\rho$-$y_{\rho'}$-preservation. In the latter case we obviously have $x = y$.*

An inspection of the game rules and the semantics together with Theorem 2 proves the following.

**Lemma 7** *For all environments $\rho$, and all truth values $x, y$ we have:*

*a) Player $\exists$ cannot eagerly make a move that is a $x_\rho$-$y_\rho$-improvement, but can always eagerly make a $\top_\rho$-$\top_\rho$-preservation.*
*b) Player $\forall$ cannot eagerly make a move that is a $x_\rho$-$y_\rho$-worsening, but can always eagerly make a $\bot_\rho$-$\bot_\rho$-preservation.*
*c) There is an ordinal $\alpha$ s.t. the deterministic rule for a fixpoint formula $\eta Z.\varphi$ is a $x_\rho$-$x_{\rho'}$-preservation when $\rho' := \rho[Z \mapsto Z^\alpha_\rho]$.*
*d) Let $Z$ be a $\eta$-variable and $\rho(Z) = Z^\alpha_{\rho_i}$ for some environment $\rho_i$ and some ordinal $\alpha > 0$. There is an ordinal $\beta < \alpha$ s.t. the deterministic rule for unfolding $Z$ is an $x_\rho$-$x_{\rho'}$-preservation when $\rho' := \rho[Z \mapsto Z^\beta_{\rho_i}]$.*

(1) TODO: ML: add viability requirement. Please check and kill

(1) A *strategy* for player $p$ is a partial function $\zeta : \mathcal{C} \to \mathcal{C}$, such that its domain is the set of configurations where player $p$ moves and for all configurations $C$ and $C'$: $\zeta(C) = C'$ implies that there is a move from $C$ to $C'$. Player $p$ plays a game according to a strategy $\zeta$ if all his choices agree with $\zeta$. A strategy for player $p$ is called a *winning strategy* if player $p$ wins every play where he plays according to this strategy.

**Lemma 8** *Let $\varphi \in 3\text{–}\mathcal{L}_\mu$ be closed. Player $\exists$ does not have a winning strategy for the game $\Gamma_\mathcal{T}(s, \varphi)$ if $[\![\varphi]\!]^\mathcal{T}(s) \neq \top$.*

**PROOF.** Suppose that on one hand $[\![\varphi]\!]^\mathcal{T}(s) \neq \top$ but Player $\exists$ has a winning strategy $\zeta$ for $\Gamma_\mathcal{T}(s, \varphi)$. Take the partial game tree induced by this strategy, i.e. the tree of all plays in which all of Player $\forall$'s choices are preserved but only those of Player $\exists$'s choices which agree with $\zeta$.

First we show that this tree contains at least one play $C_0, C_1, \ldots$ for which there is a corresponding sequence of environments $\rho_0, \rho_1, \ldots$ s.t. for all $i \in \mathbb{N}$, $C_i$ is not a $\top_{\rho_i}$-configuration. Let $\rho_0$ be the empty environment. Since $\varphi$ is closed, the root of this tree is not a $\top_{\rho_0}$-configuration. According to Lemma 7, deterministic rules preserve the truth value of a configuration – possibly extending the environments – and Player $\exists$'s choices do not improve the truth value when considering the same environment, as she is playing eagerly (being that $\zeta$ is a winning strategy for her). This can only be done by Player $\forall$. However, suppose there is a configuration $C_i$ in which Player $\forall$ makes a choice and which has a truth value other than $\top$ under $\rho_i$. Then $C_i$

is of the form $t \vdash \psi_0 \wedge \psi_1$ or $t \vdash [a]\psi$. For the former case note that the truth value of $C_i$ under $\rho_i$ is the infimum in $\mathbb{B}_3$ of its two successor's truth values under $\rho_i$. Thus, it is not $\top$ only if there is a successor which has a truth value other than $\top$ under the same environment $\rho_i$ – which will also define $\rho_{i+1}$ (that is, $\rho_{i+1} = \rho_i$). For the latter case consider $[\![[a]\psi]\!]_{\rho_i}(t)$. It can only differ from $\top$ if there is a $t'$ s.t. $t \xrightarrow{a?} t'$ and $[\![\psi]\!]_{\rho_i}(t') \neq \top$. But $t' \vdash \psi$ is a possible successor configuration of $C_i$. Thus, it is included in the tree and has a truth value which is not $\top$ under $\rho_i$ – which will again define $\rho_{i+1}$ (i.e., $\rho_{i+1} = \rho_i$).

This argument can be iterated yielding a path $C_0, C_1, \ldots$ and a sequence $\rho_0, \rho_1, \ldots$ in which $\rho_i \neq \rho_{i+1}$ only if the move from $C_i$ to $C_{i+1}$ is deterministic. $C_0, C_1, \ldots$ is a path on which no $\top$-configuration under the according $\rho_i$ occurs. Now, this path can either be finite or infinite. The first case immediately leads to a contradiction since finite paths won by Player $\exists$ necessarily end in a $\top$-configuration under any environment.

Suppose therefore that the path represents a play which is won by Player $\exists$'s winning condition 3. Then there is an outermost variable $Z$ of fixpoint type $\nu$ which occurs infinitely often in this play. Take the last occurrence of a configuration $t \vdash \nu Z.\varphi$ and name it $C_i$. By assumption, $[\![\nu Z.\varphi]\!]_{\rho_i}(t) = x$ for some $x \neq \top$ and $\rho_i$ as constructed above.

According to Lemma 7, $\rho_{i+1}$ interprets $Z$ as an approximant with some index $\alpha \neq 0$. That is, $\rho_{i+1}(Z) = Z^{\alpha}_{\rho_i}$. Lemma 7 also shows that subsequent environments $\rho_j$, $j > i$ interpret $Z$ as approximants $Z^{\beta}_{\rho_i}$ with decreasing indices $\beta$. But the ordinals are well founded. Hence, there is a $j$ s.t. $C_j = t \vdash Z$ for some $t$ and $\rho_j(Z) = Z^0_{\rho_i}$, meaning that $[\![Z]\!]_{\rho_j}(t) = \top$. But on the other hand, since $C_j$ appears on the above path, we know that $C_j$ is not a $\top_{\rho_j}$-configuration. This is a contradiction. We conclude that Player $\exists$ cannot have a winning strategy. $\square$

**Lemma 9** *Let $\varphi \in 3\text{–}\mathcal{L}_\mu$ be closed. Player $\exists$ has a winning strategy for the game $\Gamma_{\mathcal{T}}(s, \varphi)$ if $[\![\varphi]\!]^{\mathcal{T}}(s) = \top$.*

**PROOF.** Suppose $[\![\varphi]\!]^{\mathcal{T}}(s) = \top$. According to Lemma 7, Player $\exists$ can play eagerly in such a way that every reached configuration has truth value $\top$ under some environment which is constructed successively using Lemma 7 and starting with the empty environment. Note that $\varphi$ is assumed to be closed.

Player $\forall$ cannot help but to make moves that result in $\top_\rho$-configurations under the corresponding $\rho$ as well. This defines a strategy for Player $\exists$. It remains to be seen that this strategy guarantees her to win every resulting play. First, by Lemma 7 again, every resulting play is $\exists$-eager. By preservation of the

truth value, a finite play must end in a $\top$-configuration under an irrelevant environment. But then it is won by Player $\exists$ with winning condition 1 or 2.

Suppose therefore that the play $C_0, C_1, \ldots$ at hand is of infinite length. By Player $\exists$'s strategy that uses the construction of environments in Lemma 7, there are $\rho_0, \rho_1, \ldots$, s.t. $C_i$ is a $\top_{\rho_i}$-configuration for all $i \in \mathbb{N}$.

Any infinite play has a unique outermost variable $Z$ that occurs infinitely often, cf. [22]. This variable has a unique fixpoint type $\eta \in \{\mu, \nu\}$. Assume for the sake of contradiction that $fp(Z) = \mu Z.\psi$ for some $\psi$. Then take the last occurrence of a configuration $C_i = t \vdash \mu Z.\psi$. Since $Z$ is outermost, it is guaranteed to exist, for otherwise there would be another fixpoint formula that generated $\mu Z.\psi$ infinitely often.

According to the construction of the strategy, there is an ordinal $\alpha$ s.t. $\rho_{i+1}$ interprets $Z$ in the following configuration $C_{i+1} = t \vdash Z$ by $Z^{\alpha}_{\rho_i}$. Again, by the construction of the strategy using Lemma 7, the next time $Z$ occurs it is interpreted as $Z^{\beta}_{\rho_i}$ for some $\beta < \alpha$. By the well-foundedness of the ordinals, there will eventually be a $\top_{\rho_k}$-configuration $C_k = t' \vdash Z$ s.t. $\rho_k(Z) = Z^0_{\rho_i}$ which is impossible since $Z^0_{\rho_i} = \lambda s.\bot$, provided that the fixpoint type of $Z$ is $\mu$. Thus, the fixpoint type of $Z$ must have been $\nu$ which makes Player $\exists$ the winner of the play at hand. $\Box$

**Lemma 10** *Let $\varphi \in 3\text{–}\mathcal{L}_\mu$ be closed. Player $\forall$ has a winning strategy for the game $\Gamma_{\mathcal{T}}(s, \varphi)$ iff $[\![\varphi]\!]^{\mathcal{T}}(s) = \bot$.*

**PROOF.** This is the dual to Lemmas 8 and 9. Hence, it is proved in the same way exchanging $\top$ and $\bot$, "improve" and "worsen", $\nu$ and $\mu$, Player $\exists$ and $\forall$. $\Box$

**Theorem 11** *Given a KMTS $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{x} \mid x \in \mathcal{A}! \cup \mathcal{A}?\}, L)$, an $s \in \mathcal{S}$, and a closed $\varphi \in 3\text{–}\mathcal{L}_\mu$, we have:*

*(a)* $[\![\varphi]\!]^{\mathcal{T}}(s) = \top$ *iff Player $\exists$ has a winning strategy for $\Gamma_{\mathcal{T}}(s, \varphi)$,*
*(b)* $[\![\varphi]\!]^{\mathcal{T}}(s) = \bot$ *iff Player $\forall$ has a winning strategy for $\Gamma_{\mathcal{T}}(s, \varphi)$,*
*(c)* $[\![\varphi]\!]^{\mathcal{T}}(s) = ?$ *iff neither Player $\exists$ nor Player $\forall$ has a winning strategy for $\Gamma_{\mathcal{T}}(s, \varphi)$.*

**PROOF.** Parts (a) and (b) are proved in Lemmas 8, 9 and 10. For part (c) suppose that $[\![\varphi]\!]^{\mathcal{T}}(s) = ?$. Then none of the players can have a winning strategy because using parts (a) and (b) one would immediately contradict the assumption. Conversely, suppose that none of them has a winning strategy

but $[\![\varphi]\!]^{\mathcal{T}}(s) \neq ?$. Again, using (a) or (b) one obtains an immediate contradiction. □

**Theorem 12** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{x} \mid x \in \mathcal{A}\}, L)$ be a Kripke structure with $s \in \mathcal{S}$ and $\mathcal{T}' = (\mathcal{S}', \{\xrightarrow{x} \mid x \in \mathcal{A}! \cup \mathcal{A}?\}, L')$ be an abstraction of $\mathcal{T}$ with concretization function $\gamma$. Let $s' \in \mathcal{S}'$ with $s \in \gamma(s')$.*

*(a) If Player $\exists$ has a winning strategy for $\Gamma_{\mathcal{T}'}(s', \varphi)$ then $\mathcal{T}, s \models \varphi$.*
*(b) If Player $\forall$ has a winning strategy for $\Gamma_{\mathcal{T}'}(s', \varphi)$ then $\mathcal{T}, s \not\models \varphi$.*

**PROOF.** Suppose Player $\exists$ wins $\Gamma_{\mathcal{T}'}(s', \varphi)$. According to Theorem 11, we have $[\![\varphi]\!]^{\mathcal{T}'}(s') = \top$. Applying Theorem 3 we get $[\![\varphi]\!]^{\mathcal{T}}(s) = \top$, i.e. $\mathcal{T}, s \models \varphi$. Part (b) is proved analogously. □

## 5 Deciding the Model Checking Problem for 3–$\mathcal{L}_\mu$

The previous section relates model checking games with the semantics of 3–$\mathcal{L}_\mu$. An algorithm estimating the winner of the game and a winning strategy is yet to be given. Note that the result of the previous section also holds for infinite-state systems. From now on, however, we restrict to finite KMTS.

For the sake of readability we will deal with parity games. Instead of Player $\exists$ and $\forall$, we talk of Player 0 and Player 1, resp., and use $\sigma$ to denote Player 0 or 1 and $\bar{\sigma} = 1 - \sigma$ for the opponent. [3]

Parity games are traditionally used to describe the model checking game for $\mu$-calculus [7]. For simplicity, we consider parity games with dead-end vertices (see Remark 14). In order to describe our game for 3–$\mathcal{L}_\mu$, we need to generalize them in the following ways: (1) we have two types of edges: must edges and may edges, where every must edge is also a may edge, (2) terminal configurations (dead-end) are classified as either winning for one player, or as tie-configurations, and (3) an eagerness requirement is added to the winning conditions.

### *5.1 Three-valued Parity Games*

A *three-valued parity game* $G = (A, \chi)$ has an *arena* $A = (V_0, V_1, V_{tie}, \xrightarrow{must}, \xrightarrow{may})$ s.t. $V_0$, $V_1$ and $V_{tie}$ are disjoint sets of vertices. Let $V := V_0 \cup V_1 \cup V_{tie}$. Then

[3] The numbers 0 and 1 have parities and this is more intuitive for this notion of game.

$\xrightarrow{must} \subseteq \xrightarrow{may} \subseteq (V \setminus V_{tie}) \times V$, meaning that every $v \in V_{tie}$ is a dead-end. $\chi : V \to \mathbb{N}$ is a *priority function* that maps each vertex $v \in V$ to a *priority*.

A play from $v_0 \in V$ is a maximal sequence of vertices $v_0, \ldots$, where Player $\sigma$ moves from $v_i$ to $v_{i+1}$ when $v_i \in V_\sigma$ and $(v_i, v_{i+1}) \in \xrightarrow{may}$. It is called $\sigma$-*eager* iff Player $\sigma$ chooses only moves that are (also) in $\xrightarrow{must}$. A $\sigma$-eager play is *winning* for Player $\sigma$ if

- it is finite and ends in $V_{\bar{\sigma}}$, or
- it is infinite and the minimal priority occurring infinitely often is even when $\sigma = 0$ or odd when $\sigma = 1$.

All other plays are a *tie*.

(2)(3)A strategy for player $\sigma$ in the 3-valued parity game $G = (A, \chi)$ is a function $\zeta : V_\sigma \to V$ such that for all $v \in V_\sigma$ and $v' \in V$: $\zeta(v) = v'$ implies that Player $\sigma$ can move from $v$ to $v'$. A play $v_0 v_1 \ldots$ is said to *conform* to $\zeta$ if for all $k \in \mathbb{N}$, s.t. $v_k \in V_\sigma$: $v_{k+1} = \zeta(v_k)$.

A strategy $\zeta$ for player $\sigma$ is a *winning strategy* from $V' \subseteq V$ if every play that starts from a vertex in $V'$ and conforms to $\zeta$ is won by player $\sigma$. It is called a *non-losing* strategy from $V'$ if every play from $v \in V'$ conforming to $\zeta$ is either won by player $\sigma$ or a tie.

Note that we restrict ourselves to so-called *memoryless* strategies. For ordinary parity games, it is well-known that the existence of winning strategies for one of the players does not depend on considering histories or not [23]. In the appendix, we will show that the result carries over to three-valued parity games: The existence of winning strategies or non-loosing strategies for the players does not depend on considering histories of plays or not. (4) Because of this, we restrict ourselves to memoryless strategies for both three-valued and ordinary parity games in the following without mentioning this explicitly every time.

(2) TODO: ML: Removed history. Please check and kill

(3) TODO: ML: add viability requirement. Please check and kill

(4) TODO: ML: Reword? Add reference to appendix

### *5.2 Model Checking Games as Parity Games*

Just as the model checking games for the modal $\mu$-calculus can be seen as ordinary parity games [7], the model checking games of the previous section can be transformed into three-valued parity games.

Let $\mathcal{T} = \{\mathcal{S}, \{\xrightarrow{x} \mid x \in \mathcal{A}! \cup \mathcal{A}?\}, L)$ be a KMTS with starting state $s_0 \in \mathcal{S}$ and let $\varphi_0 \in 3\text{–}\mathcal{L}_\mu$. We associate with these a three-valued parity game whose vertices are the configurations of the model checking game, and whose edges are applications of the model checking game rules. The set $V_0$ consists of

all configurations in which Player $\exists$ nominally makes a choice together with configurations in which the play terminates and $\forall$ wins. Similarly, the set $V_1$ consists of all configurations in which Player $\forall$ nominally makes a choice together with configurations in which the play terminates and $\exists$ wins. The remaining configurations, i.e. the ones of the form $t \vdash q$ or $t \vdash \neg q$ with $L(t)(q) = ?$ are set to $V_{tie}$. An edge is a genuine may-edge if in its corresponding model checking game rule, the player at hand chooses a transition $s \xrightarrow{a?} t$ rather than $s \xrightarrow{a!} t$. All other game rule applications lead to must-edges.

Let $X_1, \ldots, X_n$ be all the variables occurring in $\varphi_0$. They are partially ordered by the relation $\leq_{\varphi_0}$. Note that it is possible to assign to each variable a number $\chi(X_i)$ s.t. for all $i, j = 1, \ldots, n$:

- $\chi(X_i)$ is even iff $X_i$ is of type $\nu$;
- $\chi(X_i) \leq \chi(X_j)$ whenever $X_i \leq_{\varphi_0} X_j$.

Let $m := \max\{\chi(X_1), \ldots, \chi(X_n)\}$. The priorities on the parity game vertices are assigned as follows:

$$\chi(s \vdash \psi) \;:=\; \begin{cases} \chi(X) & \text{if } \psi = X \\ m & \text{o.w.} \end{cases}$$

The following theorem then follows from the fact that the three-valued parity game is simply a different view onto the model checking game.

**Proposition 13** *Player $\exists$, resp. $\forall$ wins the model checking game $\Gamma_{\mathcal{T}}(s, \varphi)$ iff Player 0, resp. 1 wins the associated three-valued parity game from the vertex $s \vdash \varphi$. Moreover, the strategies used in both games are the same.*

**Remark 14** Since the graph of a model checking game need not be total, the corresponding three-valued parity game might have dead-end vertices. These can be eliminated by applying the following simple transformations.

(1) For a dead-end vertex in $V_\sigma$ (in which $V_\sigma$ loses), set the priority to $\bar{\sigma}$ and add a must-edge back to itself.
(2) For a (dead-end) vertex $v \in V_{tie}$, arbitrarily choose $\sigma \in \{0, 1\}$ and add $v$ to $V_\sigma$, setting its priority to $\sigma$ and adding a may-edge back to itself.

Note that these changes make the parity game total, i.e. for every $v \in V$ there is a $w \in V$ s.t. $v \xrightarrow{may} w$, and in particular there are no vertices in $V_{tie}$. Moreover, a play looping in the additional edges is won by player $\sigma$ iff the corresponding play in the original game is won by the same player. In case (1) this is because of the assigned priority; in case (2) this is because the assigned priority which repeats infinitely often is associated with a player who is forced to move along a may-edge. Hence, the play is going to be a tie.

Moreover, this construction preserves the set of vertices of the game. It also preserves the winner (if any) from each vertex, and the same strategies can be used in both games (with the exception that to get a strategy for Player $\sigma$ in the total game from a strategy in the original game, one has to add to the strategy the self loops that were added to dead-end vertices of Player $\sigma$).

### *5.3 Model Checking by Solving Three-Valued Parity Games*

Proposition 13, along with Theorem 11, implies that the model checking problem for a state $s$ of a KMTS and a formula $\varphi \in 3\text{–}\mathcal{L}_\mu$ reduces to determining the winner (if any) in the corresponding three-valued parity game from $s \vdash \varphi$.

In the remainder of this section we discuss solving three-valued parity games, which means determining the winner (if any) from every vertex. Note that there are three different outcomes for every vertex: either Player 0 or Player 1 or none of them has a winning strategy. By the definition of a winning strategy it is obviously not possible for both players to have a winning strategy.

Therefore, solving the game amounts to partitioning its set of vertices into three winning sets: $W_0, W_1, W_{tie}$, where for $\sigma \in \{0,1\}$, the set $W_\sigma$ consists of all the vertices from which Player $\sigma$ has a winning strategy and the set $W_{tie}$ consists of all the vertices from which none of the players has a winning strategy.

It is not hard to see that solving a three-valued parity game can be reduced to solving two ordinary parity games: first try to find a winning strategy for Player 0 disregarding her genuine may-edges and treating dead-end vertices in $V_{tie}$ as losing for her. If the result is negative then try to find a winning strategy for Player 1 disregarding his genuine may-edges and treating tie dead-ends as losing for him. This is reminiscent of the approach of [2], where a 3-valued interpretation of a formula in a partial model is computed by considering a pessimistic and an optimistic interpretations.

More precisely, in order to find the winning set of Player $\sigma$, we reduce $G$ into an ordinary parity game denoted $G_\sigma$ by (1) removing all the outgoing genuine may-edges of vertices of Player $\sigma$, (2) ignoring the distinction between may and must edges in the remaining edges, and (3) adding $V_{tie}$ to $V_\sigma$ (meaning that Player $\bar{\sigma}$ wins in these vertices). Note that we do not change the set of vertices, nor the priority function. Formally, $G_\sigma$ is defined as follows.

**Definition 15** *Let $G = (A, \chi)$ be a three-valued parity game with arena $A = (V_0, V_1, V_{tie}, \xrightarrow{must}, \xrightarrow{may})$. For $\sigma \in \{0,1\}$, the* $\sigma$-reduced game *is an ordinary parity game $G_\sigma = (A_\sigma, \chi)$ with arena $A_\sigma = (V_0^\sigma, V_1^\sigma, \longrightarrow)$, where $V_\sigma^\sigma = V_\sigma \cup V_{tie}$, $V_{\bar{\sigma}}^\sigma = V_{\bar{\sigma}}$, and $\longrightarrow = \xrightarrow{may} \setminus \{(v, v') \mid v \in V_\sigma \text{ and } v \xrightarrow{must} v'\}$.*

$G_\sigma$ might contain dead-end vertices, some of which result from dead-end vertices in $G$ and some result from vertices of $V_\sigma$ that had only genuine may-edges in $G$. This means that they become dead-ends in $G_\sigma$. However, $G_\sigma$ can be transformed into a game whose underlying graph is total as described in Remark 14.

**Proposition 16** *Let $G$ be a three-valued parity game. Then Player $\sigma$ has a winning strategy from set $V' \subseteq V$ in $G$ iff she has a winning strategy from $V'$ in $G_\sigma$. Moreover, a winning strategy for Player $\sigma$ from $V'$ in $G_\sigma$ is also a winning strategy for her in $G$.*

We conclude that for $\sigma \in \{0, 1\}$, the winning set of Player $\sigma$ in $G$, $W_\sigma$, is exactly the winning set of Player $\sigma$ in $G_\sigma$ and $W_{tie} = V \setminus (W_0 \cup W_1)$. Therefore, solving the three-valued parity game reduces to solving the two ordinary parity games $G_0$ and $G_1$:

**Algorithm** `SolveThreeValuedGame` $(G)$

(1) $(W_0^0, W_1^0) :=$ `SolveOrdinaryGame` $(G_0)$;
(2) $(W_0^1, W_1^1) :=$ `SolveOrdinaryGame` $(G_1)$;
(3) $(W_0, W_1, W_{tie}) := (W_0^0, W_1^1, V \setminus (W_0^0 \cup W_1^1))$;
(4) **return** $(W_0, W_1, W_{tie})$;

Solving $G_0$ and $G_1$ can be done using any of the existing algorithms for solving ordinary parity games, maintaining their complexity. Moreover, winning strategies in the three-valued game can be easily obtained from winning strategies in the ordinary games, since a winning strategy for Player $\sigma$ in $G_\sigma$ is also a winning strategy for Player $\sigma$ in $G$.

**Remark 17** Note that even if the graph of the original game $G$ is connected, the underlying graph of $G_\sigma$ might not be connected. Thus, depending on the algorithm for solving ordinary parity games, if we wish to classify *all* the vertices of $G$, it might be necessary to invoke the algorithm for every connected component in $G_\sigma$ separately (for example, if the algorithm has an on-the-fly nature and it considers only reachable vertices).

When applied to model checking whether $s \models \varphi$, then after solving the corresponding three-valued game we check whether $v = s \vdash \varphi$ is in $W_0$, $W_1$, or $W_{tie}$ and conclude *true*, *false*, or *indefinite*, respectively.

**Example 18** *In terms of a parity game, the model checking game shown in Figure 3 can be visualized as shown in Figure 4. Round vertices denote vertices of Player 0, whereas square vertices are of Player 1. Vertex $v_{18}$ is shaped as a rotated square to denote that it is a (dead-end) tie vertex. The numbers labelling the vertices denote their priorities.*

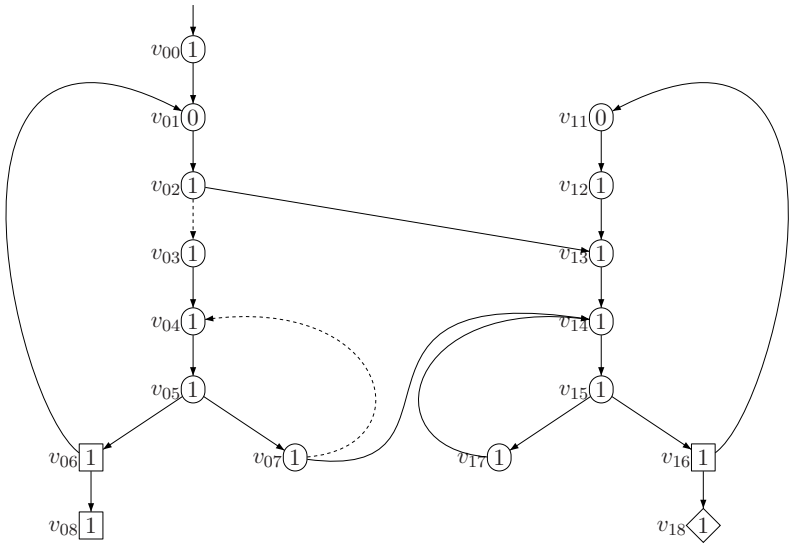
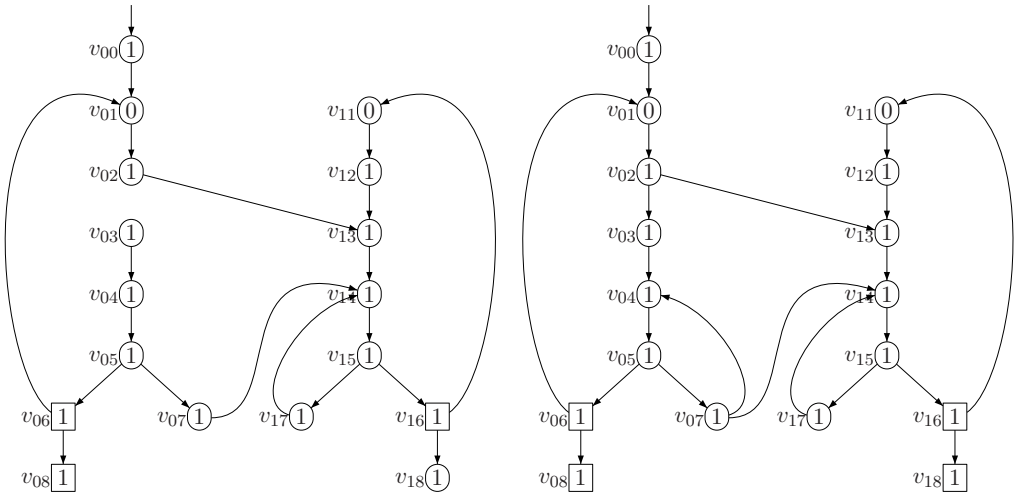



Fig. 4. Game graph of the parity game corresponding to the game from Figure 3




(a) $G_0$ (b) $G_1$

Fig. 5. Game graphs of the reduced games of the parity game from Figure 4

*The reduction to two games, $G_0$ and $G_1$ is shown in Figure 5. Note that genuine may edges of Player 0 vertices are removed in $G_0$ and that $v_{18}$ is declared as a Player 0 vertex in $G_0$ (meaning that Player 1 wins in it) and a Player 1 vertex in $G_1$ (meaning that Player 0 wins in it).*

*We see that the only vertex from which Player 0 has a winning strategy in $G_0$ is $v_{08}$. In $G_1$, Player 1 has no winning strategy regardless in which vertex the game starts. We conclude that $W_0 = \{v_{08}\}$, $W_1 = \emptyset$, and that all remaining vertices form the set $W_{tie}$.*

## 6 Refinement

Assume we are interested in knowing whether a concrete state $s_c$, described by an abstract state $s_a$, satisfies a given formula $\varphi$. Let $(W_0, W_1, W_{tie})$ be the winning sets computed for the three-valued parity game obtained by the model checking game $\Gamma_{\mathcal{T}}(s_a, \varphi)$ for $s_a$ and $\varphi$. By Theorems 12 and 13 if the vertex $v = s_a \vdash \varphi$ is in $W_0$ or $W_1$ then the answer to the model checking problem is clear: $s_c \models \varphi$ if $v \in W_0$ (meaning that Player 0 has a winning strategy from $v$ in the parity game, hence, Player $\exists$ is the winner of the underlying model checking game). Similarly, $s_c \not\models \varphi$ if $v \in W_1$. However, if $v \in W_{tie}$, the result is *don't know* and we have to refine the abstraction to get the answer.

This means that a refinement step is required if none of the players has a winning strategy from $v$. Based on this property, it was stated in [5,6] that rather than calling a two-valued parity games solver twice it is more helpful for refinement purposes to combine the two runs. This is because combining both runs carries more information about the cause for the lack of winning strategies.

But even in a combined fashion of two ordinary runs, the above approach in which the algorithm looks for winning strategies bears a significant disadvantage: if the algorithm looks for *winning strategies* it produces witnesses for its answer only in those cases in which no refinement is needed. Some notion of witness to its answer is, however, needed if the answer is that none of the players has a winning strategy. This is why we suggest to consider *non-losing strategies* instead.

### *6.1 Using Non-Losing Strategies to Solve the Game*

**Lemma 19** *Let $G$ be a three-valued parity game. Player $\sigma$ has a non-losing strategy from $v$ in $G$ iff player $\bar{\sigma}$ does not have a winning strategy from $v$ in $G$.*

The first direction of the lemma is quite clear as it is impossible that Player $\sigma$ has a non-losing strategy and at the same time Player $\bar{\sigma}$ has a winning strategy. For the other direction we use the following proposition that also provides a construction of a non-losing strategy for Player $\bar{\sigma}$ in case Player $\sigma$ does not have a winning strategy. Recall that in order to compute winning sets and strategies in a three valued parity game $G$ we considered in Section 5.3 the reduced (ordinary) games $G_0$ and $G_1$. We now note that the same approach can also be used to compute non-losing strategies for the players. This is formalized by the following proposition, which is in a sense the dual of proposition 16.

**Proposition 20** *Let $G$ be a three-valued parity game. Then Player $\sigma$ has a*

*non-losing strategy from $V' \subseteq V$ in $G$ iff she has a winning strategy from $V'$ in $G_{\bar{\sigma}}$. Moreover, a winning strategy for Player $\sigma$ from $V'$ in $G_{\bar{\sigma}}$ is in itself a non-losing strategy for her in $G$.*

We now return to the proof of Lemma 19.

**PROOF.** [Lemma 19] Proposition 20 states that Player $\sigma$ has a non-losing strategy from $v$ in $G$ iff she has a winning strategy from $v$ in $G_{\bar{\sigma}}$. By determinacy of ordinary parity games this happens iff Player $\bar{\sigma}$ does not have a winning strategy from $v$ in $G_{\bar{\sigma}}$ and by Proposition 16 this is iff Player $\bar{\sigma}$ does not have a winning strategy from $v$ in $G$. This concludes the proof of Lemma 19. □

Lemma 19 implies that a non-losing strategy for a player can be used as a witness to explain why the opponent does not win. Moreover, unlike winning strategies, where it is possible that no player has one, the above lemma implies that at least one player has a non-losing strategy, thus such an explanatory information always exists (at least for one player). This is formalized in the following lemma.

**Lemma 21** *Let $G$ be a three-valued parity game and $v$ a vertex in the game. At least one of the players has a non-losing strategy from $v$ in $G$.*

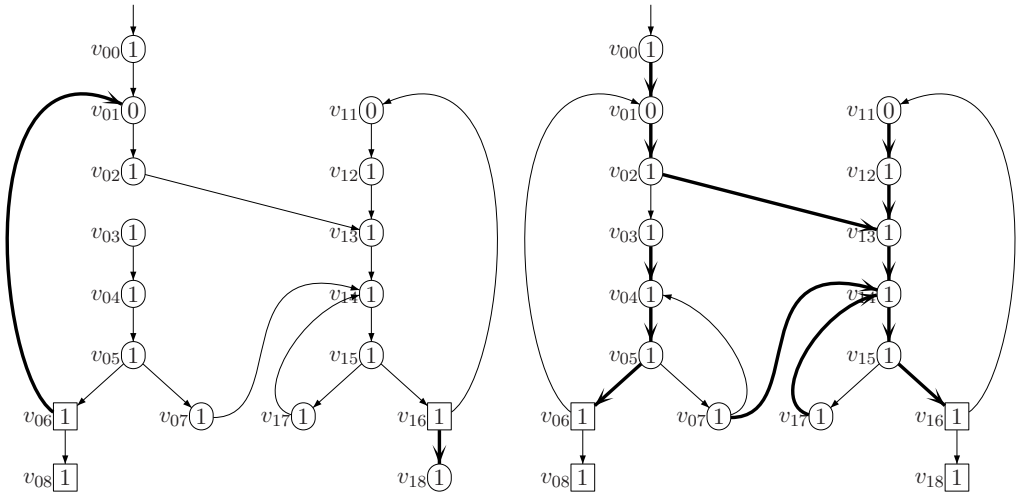
**PROOF.** Suppose none of the players has a non-losing strategy from $v$ in $G$. According to Lemma 19 both players would have to have a winning strategy in the game $G$ which is clearly impossible. □

Lemma 21 holds the key for refinement: if we use an algorithm that computes non-losing strategies then we will always have a witness.

Furthermore, Lemmas 19 and 21 also provide an alternative approach for solving the three-valued game by considering non-losing strategies rather than winning strategies, as they imply that for a three-valued parity game:

(1) $v \in W_\sigma$ iff only Player $\sigma$ has a non-losing strategy from $v$.
(2) $v \in W_{tie}$ iff both players have non-losing strategies from $v$.

In particular, Proposition 20 used in the proof of Lemma 19 provides a way to compute non-losing strategies using the reduction approach: to compute a strategy that is non-losing for Player $\sigma$ from $V \setminus W_{\bar{\sigma}}$ in $G$ we compute a strategy that is winning for Player $\sigma$ from $V \setminus W_{\bar{\sigma}}$ in the ordinary game $G_{\bar{\sigma}}$.




(a) $G_0$ (b) $G_1$

Fig. 6. Winning strategies of the reduced games of the parity game from Figure 4

Thus, in comparison to the previous reduction approach (from Section 5.3), we use here the same reduced ordinary parity games, but now in the reduced game $G_\sigma$, we are interested in a winning strategy of Player $\bar{\sigma}$ rather than of $\sigma$. As stated earlier, this approach is particularly helpful when refinement is needed. Here again it might be necessary to invoke the solver of each ordinary parity game several times in case the resulting game graph is not connected (see Remark 17).

**Example 22** *Let us reconsider the games shown in Figure 5, where we are now interested in non-losing (rather than winning) strategies of the players in the original game. In $G_0$, Player 1 has a winning strategy from all the vertices except $v_{08}$, which is shown in Figure 6(a) by bold edges. This strategy constitutes a non-losing strategy for him in the original game $G$ from the same vertices. Similarly, Player 0 has a strategy to win every play in $G_1$, regardless of where the play starts (see Figure 6(b)). Consequently, she also has a non-losing strategy from every vertex in the original game $G$. We conclude that $v_{08}$, for which only Player 0 has a non-losing strategy in $G$, is in $W_0$, whereas the rest of the vertices are in $W_{tie}$ since both players have non-losing strategies from them. As can be expected, this is consistent with Example 18, where winning strategies were considered.*

### *6.2 Refinement with Non-Losing Strategies*

When using an algorithm that solves the three-valued parity game by computing non-losing strategies, refinement is needed in the case where both players have non-losing strategies from $v = s_a \vdash \varphi$ (meaning that $v \in W_{tie}$).

As in most cases, our refinement consists of two parts. First, we choose a criterion that tells us how to split the abstract states. We then construct the refined abstract model, using the refined abstract state space. In the rest of this section we refer to the first part.

Given that $v \in W_{tie}$, our goal in the refinement is to find and eliminate at least one of the causes of the indefinite result. Thus, the criterion for splitting the abstract states is obtained from a *failure vertex*. Intuitively, this is a vertex $v_f = s'_a \vdash \varphi'$ such that (1) $v_f \in W_{tie}$; (2) the classification of $v_f$ to $W_{tie}$ *affects* the indefinite result of $v$; and (3) the indefinite classification of $v_f$ can be *changed* by splitting it. The latter requirement means that the vertex $v_f$ *itself* is responsible for introducing (some) uncertainty. The other requirements demand that this uncertainty is relevant to the result in $v$.

Recall that from each vertex in $W_{tie}$ both players have non-losing strategies. (5) They can be combined into one strategy for each player. Thus, each player $\sigma \in \{0, 1\}$ has a strategy that is non-losing for him from each vertex in $V \setminus W_{\bar{\sigma}}$ and in particular from $W_{tie}$. Let $\zeta_0$ and $\zeta_1$ be the corresponding strategies of Player 0 and Player 1 respectively. These strategies can be computed using the reduction approach, as explained in Section 6.1.

(5) TODO: ML: Removed comment on 'memoryless'

We use the non-losing strategies $\zeta_0$ and $\zeta_1$ for the failure search. Our failure search basically follows the unique play obtained by letting the players play against each other using their non-losing strategies until it identifies a failure vertex $v'$ and a *cause* for the failure. More specifically, the failure search proceeds from one tie-vertex (i.e. vertex in $W_{tie}$) to the next along this play, guided by the non-losing strategies: from $v \in V_\sigma$ it proceeds to $\zeta_\sigma(v)$. This continues until one of three possibilities occurs:

(1) The search reaches a (dead-end) vertex in $V_{tie}$.
(2) The search reaches a vertex in $W_\sigma$ for $\sigma \in \{0, 1\}$.
(3) The search reaches a vertex that was already visited.

Note the following facts regarding the play:

- The play is uniquely determined by $\zeta_0$ and $\zeta_1$.
- The play is a tie, as it is non-losing for both players (conforms to a non-losing strategy of each of them).
- (6) The play is a simple regular path if $|G| = n < \infty$ is finite, and one of the three possibilities occurs after at most $n$ steps in this play.

(6) TODO: ML: Removed reference to 'memoryless'

Now, in the first possibility $v \in V_{tie}$ is considered a failure vertex, since changing its classification to $V_0$ or $V_1$ (by splitting it) would make one of the players closer to winning.

As for the second and third possibilities, in each of them there exists one player

that is "closer" to winning the play. In the second possibility this is Player $\sigma$, for which the play reached a vertex in $W_\sigma$. In the third possibility this is the player $\sigma$ that corresponds to the parity of the minimal priority that appears in the loop that results from the two occurrences of the same vertex. [7] Note that having identified a loop in the play means that the rest of the play will be an infinite unwinding of the loop. Thus, the minimal priority that will occur infinitely often in the play will be the minimal priority that appears on the loop, whose parity corresponds to $\sigma$.

[7] TODO: ML: Did some reowording

Having that Player $\sigma$ is "closer" to winning the play, and yet knowing that the play is a tie, implies that there has to exist a genuine may-edge used by Player $\sigma$ in the prefix of the play (otherwise Player $\sigma$ would win). All of these genuine may-edges of player $\sigma$ are candidates to be considered a failure cause with their source vertex being the failure vertex. This is because changing the may-edge into a must-edge (by splitting the source vertex) would make Player $\sigma$ closer to winning and on the other hand removing the edge altogether would make Player $\bar{\sigma}$ closer to winning. The choice of one failure vertex from this set of candidates is a matter of heuristics.

To sum up, given a partition of the vertices of $G$ to $(W_0, W_1, W_{tie})$ and given non-losing strategies $\zeta_0$ and $\zeta_1$ for Player 0 and 1 resp., the algorithm `FindFailure`$(v)$ returns a failure vertex $v_f$ and cause for $v \in W_{tie}$.

**Algorithm** `FindFailure` $(v)$

(1) **if** $v \in V_{tie}$ **then return** $(v, tie)$;
(2) **else if** $v \in W_\sigma$ **then return** `choose`$(visited \cdot v, \sigma)$;
(3) **else if** $v \in visited$ **then return** `choose`$(visited \cdot v,$ `parity`$(visited, v))$;
(4) **else** // continue with the search
add $v$ to $visited$;
let $\sigma$ be such that $v \in V_\sigma$;
let $w := \zeta_\sigma(v)$;
`FindFailure`$(w)$;

where the function `parity`$(sequence, v)$ returns 0 if the minimal priority that appears in the sequence starting from the vertex $v$ is even, and 1 if it is odd. The function `choose`$(sequence, \sigma)$ chooses a vertex from $V_\sigma$ that appears in the sequence and has a genuine may-edge to its successor in the sequence. It returns the chosen vertex and the corresponding may-edge.

This concludes the description of how `FindFailure` looks for a failure vertex and cause. A simple case analysis shows the following.

**Theorem 23** *Let $v_f$ be a vertex that is returned by* `FindFailure`*$(v)$ as a failure vertex. The failure cause can either be the fact that $v_f \in V_{tie}$, or it can be a genuine may-edge $(v_f, v') \in \xrightarrow{may} \setminus \xrightarrow{must}$.*

Once we are given a failure vertex $v_f = s'_a \vdash \varphi'$ and a corresponding reason for failure, we guide the refinement to discard the cause for failure in the hope for changing the model checking result to a definite one. This is done as in [5], where the failure information is used to determine how the set of concrete states represented by $s'_a$ should be split in order to eliminate the failure cause. A criterion for splitting *all* abstract states can then be found by known techniques, depending on the abstraction used (e.g. [24,25]).

After refinement, one has to re-run the model checking algorithm on the game graph based on the refined KMTS to get a definite value for $s_c$ and $\varphi$. However, we can restrict this process to the previous $W_{tie}$. When constructing the game graph based on the refined KMTS, every vertex $s^2_a \vdash \psi$ for which a vertex $s_a \vdash \psi$ (where $s^2_a$ results from splitting $s_a$) exists in $W_0$ or $W_1$ in the previous game graph can be considered a dead-end winning for Player 0 or Player 1, respectively. This way we avoid unnecessary refinement.

**Example 24** *Reconsider the game shown in Figure 4, where $v_{00} \in W_{tie}$ (see Examples 18 and 22), and the non-losing strategies of the players discussed in Example 22 (see Figure 6). Following both non-losing strategies in $G$ will guide the play starting in $v_{00}$ to vertex $v_{18}$ through non-winning vertices for both players. Consequently, the state underlying $v_{18}$ should be refined, which is $s_1$ (see Figure 3).*

*Now, assume that $v_{18}$ is a Player 0 vertex (for example after refinement). This means that $v_{18}$ is now winning for Player 1 in $G$. As before, Player 1 has a winning strategy in $G_0$, and thus a non-losing strategy in $G$, by forcing the play to $v_{18}$. But also Player 0 can still win in $G_1$, and thus not lose in $G$, by choosing in $v_{02}$ the edge leading to $v_{03}$ (instead of the edge leading to $v_{13}$). Now, the combined non-losing strategies would give a loop $v_{00}v_{01}v_{02}v_{03}v_{04}v_{05}v_{06}v_{01}$, which asks for refining the may edge from $v_{02}$ to $v_{03}$.*

*This also demonstrates the importance, in terms of the refinement, of not limiting the computation of winning strategies in the reduced graphs $G_0$ and $G_1$ to vertices that are reachable from the vertex of interest. Namely, $v_{03}$ is unreachable in $G_0$ from $v_{00}$. Yet, the non-losing strategy of Player 0 takes the play to $v_{03}$. Thus, the information about a non-losing strategy of Player 1 from $v_{03}$ is essential in order to follow the tie play that guides the refinement. This information is only available provided that $v_{03}$ was considered during the computation of a winning strategy for Player 1 in $G_0$.*

## 7 Conclusion

In this work we present a game-based model checking for abstract models with respect to specifications in $\mu$-calculus, interpreted over a 3-valued semantics. We also suggest an automatic refinement, in case the model checking result is indefinite.

In contrast to [6], model checking is determined by solving two model checking games for $\mu$-calculus with respect to the 2-valued semantics. However, these games are based on the full board for the 3-valued game. This is particularly important for refinement, for which the board for the 3-valued game holds more information than the two boards of the 2-valued games.

The refinement is based on the novel notion of a *non-losing* strategy. In case the model checking result is indefinite, both players have non-losing strategies. Combining these strategies of the two players comprises a play, resulting with a tie. From this play, a failure node and a cause are derived and exploited for refinement.

A non-losing strategy for Player $\sigma$ can easily be extracted by computing a winning strategy for Player $\sigma$ on the 2-valued game $G_{\bar{\sigma}}$. This can be done using any algorithm for solving 2-valued model checking games. Thus, our approach can take advantage of efficient algorithms for this problem, such as Jurdzinski's algorithm for parity games [8].

Recently, there has been an active research on completeness and precision of abstractions for branching time logics (e.g. [26–30]). Various abstract models which are more expressive than KMTSs were suggested. These models add some kind of disjunctiveness to the model: for example, [28] introduces focus operations, and [27,29] uses hyper-transitions (first introduced by [31]) to model the abstract transitions. Some of these models (e.g. [28,30]) also consider fairness conditions. While fairness requires different techniques (e.g. in order to determine how to refine the fairness conditions), disjunctiveness can be handled by the approach suggested in this paper. This simply requires to define a 3-valued model checking game for such models (see for example [29] for hyper-transitions) and to encode the game as a 3-valued parity game (e.g. [28,29] use 2-valued parity games).

## References

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

[2] G. Bruns, P. Godefroid, Model checking partial state spaces with 3-valued

temporal logics., in: N. Halbwachs, D. Peled (Eds.), Computer Aided Verification, 11th International Conference, CAV '99, Vol. 1633 of Lecture Notes in Computer Science, Springer, 1999, pp. 274–287.

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

[4] P. Godefroid, R. Jagadeesan, On the expressiveness of 3-valued models, in: Verification, Model Checking and Abstract Interpretation (VMCAI), Vol. 2575 of Lecture Notes in Computer Science, 2003, pp. 206–222.

[5] S. Shoham, O. Grumberg, A game-based framework for CTL counterexamples and 3-valued abstraction-refinemnet, in: Computer Aided Verification (CAV), Vol. 2725 of Lecture Notes in Computer Science, 2003, pp. 275–287.

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

[7] E. A. Emerson, C. S. Jutla, A. P. Sistla, On model-checking for fragments of mu-calculus, in: Computer-Aided Verification, Vol. 697 of Lecture Notes in Computer Science, 1993, pp. 385–396.

[8] M. Jurdzinski, Small progress for solving parity games, in: STACS, Vol. 1770 of Lecture Notes in Computer Science, 2000, pp. 290–301.

[9] D. Kozen, Results on the propositional $\mu$-calculus, Theoretical Computer Science 27 (1983) 333–354.

[10] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, L. J. Hwang, Symbolic model checking: $10^{20}$ states and beyond, Information and Computation 98 (2) (1992) 142–170.

[11] E. Emerson, C. Lei, Efficient model checking in fragments of the propositional $\mu$-calculus, in: Symposion on Logic in Computer Science (LICS), IEEE Computer Society Press, Washington, D.C., USA, 1986, pp. 267–278.

[12] C. Stirling, D. J. Walker, Local model checking in the modal mu-calculus, in: TAPSOFT'89: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Vol. 351 of Lecture Notes in Computer Science, 1989, pp. 369–383.

[13] G. Winskel, Model checking in the modal $\nu$-calculus, in: International Colloquium on Automata, Languages, and Programming (ICALP), Vol. 372 of Lecture Notes in Computer Science, 1989, pp. 761–772.

[14] R. Cleaveland, Tableau-based model checking in the propositional mu-calculus, Acta Inf. 27 (1990) 725–747.

[15] D. Long, A. Browne, E. Clark, S. Jha, W. Marrero, An improved algorithm for the evaluation of fixpoint expressions, in: Computer Aided Verification, (CAV), Vol. 818 of Lecture Notes in Computer Science, 1994, pp. 338–350.

[16] C. Stirling, Modal and Temporal Properties of Processes, Springer, 2001.

[17] P. Godefroid, R. Jagadeesan, Automatic abstraction using generalized model checking, in: Computer-Aided Verification (CAV), Vol. 2404 of Lecture Notes in Computer Science, 2002, pp. 137–150.

[18] P. Cousot, R. Cousot, Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximation of fixed points, in: ACM Symposium on Principles of Programming Languages, 1977, pp. 238–252.

[19] D. Dams, R. Gerth, O. Grumberg, Abstract interpretation of reactive systems, ACM Transactions on Programming Languages and Systems (TOPLAS) 19 (2).

[20] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, S. Bensalem, Property preserving abstractions for the verification of concurrent systems, Formal Methods in System Design 6 (1995) 11–45.

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

[22] C. Stirling, Local model checking games, in: Concurrency Theory (CONCUR), Vol. 962 of Lecture Notes in Computer Science, 1995, pp. 1–11.

[23] E. Emerson, C. Jutla, Tree automata, mu-calculus and determinacy, in: Proc. 32th Symp. on Foundations of Computer Science (FOCS), IEEE Computer Society Press, 1991, pp. 368–377.

[24] E. Clarke, A. Gupta, J. Kukula, O. Strichman, SAT based abstraction-refinement using ILP and machine learning techniques, in: Computer-Aided Verification (CAV), Vol. 2404 of Lecture Notes in Computer Science, 2002, pp. 265–279.

[25] E. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, Counterexample-guided abstraction refinement, in: Computer Aided Verification (CAV), Vol. 1855 of Lecture Notes in Computer Science, 2000, pp. 154–169.

[26] K. Namjoshi, Abstraction for branching time properties, in: Computer Aided Verification (CAV'03), Vol. 2725 of LNCS, 2003, pp. 288–300.

[27] S. Shoham, O. Grumberg, Monotonic abstraction-refinement for CTL, in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Vol. 2988 of LNCS, 2004, pp. 546–560.

[28] D. Dams, K. Namjoshi, The existence of finite abstractions for branching time model checking, in: IEEE Symposium on Logic in Computer Science (LICS), 2004, pp. 335–344.

[29] L. de Alfaro, P. Godefroid, R. Jagadeesan, Three-valued abstractions of games: Uncertainty, but with precision, in: IEEE Symposium on Logic in Computer Science (LICS), 2004, pp. 170–179.

[30] D. Dams, K. S. Namjoshi, Automata as abstractions, in: Verification, Model Checking and Abstract Interpretation (VMCAI'05), Vol. 3385 of Lecture Notes in Computer Science, 2005, pp. 216–232.

[31] K. Larsen, L. Xinxin, Equation solving using modal transition systems, in: IEEE Symp. on Logic in Computer Science (LICS), 1990, pp. 108–117.

## A History-dependent versus Memoryless Strategies

We obtained all results in this paper using only so-called *memoryless* strategies, for model-checking games as well as for three-valued and ordinary two-valued parity games.

While in the setting of model-checking games and ordinary parity games the notion of *history-dependent strategies* does not change results on existence of winning strategies, the same result is a priori not clear for three-valued games. Although this result can be obtained as a generalization from the according result for ordinary parity games [23], we prefer to give a simple proof based on [23] and the results of the previous sections.

[8] TODO: ML: add viability requirement. Please check and kill

[8] A history-dependent strategy for Player $\sigma$ in the 3-valued parity game $G = (A, \chi)$ is a function $\zeta : V^*V_\sigma \to V$ such that for all $w \in V^*$, $v \in V_\sigma$ and $v' \in V$: $\zeta(wv) = v'$ implies that Player $\sigma$ can move from $v$ to $v'$. It is called *memoryless* if for all $w, w' \in V^*$, $v \in V_\sigma$: $\zeta(wv) = \zeta(w'v)$. A play $v_0v_1\ldots$ is said to *conform* to $\zeta$ if for all $k \in \mathbb{N}$, s.t. $v_k \in V_\sigma$: $v_{k+1} = \zeta(v_0 \ldots v_k)$.

**Theorem 25** *Player $\sigma$ has a (history dependent) winning, resp. non-losing strategy in a three-valued parity game $G$ iff she has a memoryless winning, resp. non-losing strategy.*

**PROOF.** Let us first consider winning strategies. As a memoryless strategy is also a history-dependent one, the implication from right to left is trivial.

Assume that Player $\sigma$ has a (history-dependent) winning strategy $\zeta : V^*V_\sigma \to V$ in $G$, which is not necessarily a memoryless one. As every play conforming to $\zeta$ is winning for Player $\sigma$, it is necessarily $\sigma$-eager. In other words, for all $k \in \mathbb{N}$, s.t. $v_k \in V_\sigma$ and $v_{k+1} = \zeta(v_0 \ldots v_k)$, there is [9] a must-edge from $v_k$ to $v_{k+1}$. Thus, $\zeta$ is a strategy for $G_\sigma$. Now observe that every play in $G_\sigma$ conforming to $\zeta$ is winning, meaning that $\zeta$ is a (history-dependent) winning strategy for $G_\sigma$.

[9] TODO: sharon: removed "only"

Thus, Player $\sigma$ has a winning strategy for $G_\sigma$. By [23], this implies that Player $\sigma$ has a memoryless winning strategy for $G_\sigma$. Due to Proposition 16, this shows the existence of a memoryless winning strategy for Player $\sigma$ in $G$.

Similarly, for non-losing strategies, the implication from right to left is trivial.

Assume that Player $\sigma$ has a history-dependent non-losing strategy $\zeta : V^*V_\sigma \to V$ in $G$, which is not necessarily a memoryless one.

We show that $\zeta$ is a (history-dependent) winning strategy for Player $\sigma$ in $G_{\bar{\sigma}}$. By [23], this implies that Player $\sigma$ has a memoryless winning strategy for $G_{\bar{\sigma}}$. Due to Proposition 20, this shows the existence of a memoryless non-losing strategy for Player $\sigma$ in $G$.

As every play conforming to $\zeta$ is non-losing for Player $\sigma$, it is either (1) winning for Player $\sigma$, (2) ending in a vertex from $V_{tie}$, [10] or (3) not $\bar{\sigma}$-eager. This implies that every play of $G_{\bar{\sigma}}$ conforming to $\zeta$ is either a play in $G$, in which case it is one of (1) or (2) and thus winning for Player $\sigma$, or, a finite play ending in a vertex of Player $\bar{\sigma}$ and thus also winning for Player $\sigma$. Hence, Player $\sigma$ has a winning strategy in $G_{\bar{\sigma}}$.

[10] TODO: sharon:added case 2