# Certification for $\mu$-Calculus with Winning Strategies

Martin Hofmann[1], Christian Neukirchen[1⋆], and Harald Rueß[2]

[1] Department of Informatics, Ludwig-Maximilians-Universität, München, Germany
[2] fortiss, An-Institut Technische Universität München, 80805 München, Guerickestr. 25, Germany

**Abstract.** We define memory-efficient certificates for $\mu$-calculus model checking problems based on the well-known correspondence between $\mu$-calculus model checking and winning certain parity games. Winning strategies can be independently checked, in low polynomial time, by observing that there is no reachable strongly connected component in the graph of the parity game whose largest priority is odd. Winning strategies are computed by fixpoint iteration following the naive semantics of $\mu$-calculus. We instrument the usual fixpoint iteration of $\mu$-calculus model checking so that it produces evidence in the form of a winning strategy; for a formula $\phi$ with fixed alternation depth, these winning strategies can be computed in polynomial time in $|S|$ and in space $O(|S|^2|\phi|^2)$, where $|S|$ is the size of the state space and $|\phi|$ the length of the formula $\phi$. On the technical level our work yields a new, simpler, and immediate constructive proof of the correspondence between $\mu$-calculus and parity games. A prototypical implementation of a $\mu$-calculus model checker generating these certificates has been developed.

## 1 Introduction

We address the problems (1) of constructing concise certificates for $\mu$-calculus model checking problems, and (2) for efficiently and independently checking these certificates by means of a trustworthy checker. Our main result here is an effective and low overhead instrumentation of the usual fixpoint iteration of $\mu$-calculus model checking [3] for generating certificates that are independently checkable in low polynomial time.

There are a number of results and algorithms for constructing witnesses and counterexamples of various forms for different sublogics, including *LTL*, *ACTL*, *CTL*, *CTL*$^*$, or the $\mu$-calculus [7, 1, 24, 33, 6, 27, 14, 34]. For example, for linear temporal logic (*LTL*) restricted to the temporal operators F, U and X, a positive certificate can be given by a finite *path*. Model checkers for *CTL*$^*$ (for example, SMV) are capable of generating *counterexamples*[7] in the form of a *lasso*; that is, infinite sequences of states $s_0, \ldots, s_i, (s_{i+1}, \ldots, s_k)^\omega$ which end up repeating periodically after some prefix of length $i$. Whereas lasso-shaped sequences

⋆ The author was supported by DFG Graduiertenkolleg 1480 (PUMA).

refute properties assumed for *all* possible paths, they fail, for example, in falsifying conjectured *existence* of certain paths. *Witnesses* for full *CTL* have been proposed by Shankar and Sorea [27, 29]. These results are based on a symbolic representation of witnesses that enables the extraction of explicit witnesses (and counterexamples) for full *CTL* model checking.

Local model checking procedures for determining whether finite-state systems have properties expressible in the $\mu$-calculus incrementally construct tableau proofs [35, 31, 8]. These tableaux can be proof-checked independently, but the size of the constructed tableaux may grow exponentially in the number of states of the underlying transition system. Based on the tableau method of local $\mu$-calculus model checking, Kick [18] proposes an optimized construction by identifying isomorphic subproofs. Namjoshi [20] introduces the notion of a *certifying model checker* that can generate independently checkable witnesses for properties verified by a model checker. He defines witnesses for properties of labelled transition systems expressed in the $\mu$-calculus based on parity games over alternating tree automata. These developments rely on $\mu$-calculus signatures [32] for termination, and are also based on the correspondence between $\mu$-calculus model checking with winning parity games [12].

The developments in this paper can certify full $\mu$-calculus model checking problems. Moreover, in order to certify that a given formula does *not* hold for some state, the result of checking the *dual formula* (cf. Lemma 1) is certified instead. In this way, certificates of the dual formula may be regarded as *generalized counterexamples* of the original formula.

Our approach of instrumenting $\mu$-calculus model checking fixpoint iteration with the computation of witnesses, including the underlying notion and algebra of *partial winning strategies*, is novel. Moreover, in contrast to the previous work on local $\mu$-calculus model checking, the witnesses generated by our global model checking algorithm are rather space-efficient, as they can be represented in space in $O(|S|^2|\phi|^2)$, where $|S|$ is the size of the state space and $|\phi|$ is the length of the formula $\phi$.

Our constructions build on the well-known equivalence of model checking for the $\mu$-calculus with winning corresponding parity games [12, 34, 13]. Because of the determinacy of parity games (see [19]), players of these games may restrict themselves to considering memoryless strategies only. In particular, there are no draws and exactly one of the players has a winning strategy for each vertex of the game graph. Algorithms for generating witnesses for players of parity games and their complexity are described by Jurdziński [17].

On the technical level our work can be seen as a new, simpler, and immediately constructive proof of the correspondence between $\mu$-calculus and parity games. Winning strategies are computed by fixpoint iteration following the naive semantics of $\mu$-calculus. No complex auxiliary devices such as signatures [32] or alternating automata [12] are needed. It should be possible to instrument existing implementations (such as the one integrated in the PVS theorem prover [23]) of $\mu$-calculus based on fixpoint iteration to generate these certificates.

**Roadmap.** This paper is structured as follows. In Sections 2 and 3 we summarize some standard developments for the $\mu$-calculus in order to keep the paper self-contained. Section 3 also contains a low polynomial-time checker for certificates which is inspired by the standard algorithm for checking for nonemptiness of Streett automata. Section 4 elaborates the correspondence between $\mu$-calculus and winning parity games and in particular contains a new constructive proof of the correspondence (Theorem 2). Section 5 provides the technical details, first, of the central notion of *partial winning strategies*, and, second, for instrumenting the usual $\mu$-calculus fixpoint iteration with the computation of partial winning strategies. For ease of exposition of this algorithm, we choose *systems of equations* as an alternative representation of $\mu$-calculus formulas. The corresponding implementation of a witness-generating $\mu$-calculus model checker is presented in Section 6, and the feasibility of our approach is demonstrated by means of selected benchmark examples. Concluding remarks, including further applications of our technical results on witness-generation and -checking, are presented in Section 7.

An earlier version of this paper, without implementation and the use of equation systems, has been presented at the VeriSure 2013 workshop (associated with CAV 2013) [16].

## 2 Syntax and Semantics

We assume variables $X \in \mathcal{X}$, propositions $p \in \mathcal{P}$, and actions $a \in \mathcal{A}$.

### 2.1 $\mu$-Calculus Formulas

**Definition 1.** *The set of $\mu$-calculus formulas is given by the grammar*

$$\phi \ ::= \ X \mid p \mid \neg p \mid \langle a \rangle \phi \mid [a]\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \mu X.\, \phi \mid \nu X.\, \phi$$

The set of free variables $FV(\phi) \subseteq \mathcal{X}$ , the size $|\phi|$ of a formula, and the substitution $\phi[Z := \psi]$ of formula $\psi$ for any free occurrence $Z \in FV(\phi)$ are defined in the usual way. Note that negation is allowed for propositions only, hence all syntactically valid formulas are monotonic in their free variables and no considerations of polarity need to be taken into account.

The notations $Q \in \{\mu, \nu\}$, $M \in \{[a], \langle a \rangle \mid a \in \mathcal{A}\}$, $* \in \{\wedge, \vee\}$ are used to simplify inductive definitions.

The semantics of $\mu$-calculus formulas is given in terms of labelled transition systems (LTS), consisting of a nonempty set of states $S$, and a family of left-total[3] relations $\xrightarrow{a} \in S \times S$ for each action $a \in \mathcal{A}$ and, finally, an assignment $T \in S \to 2^{\mathcal{P}}$ which tells for each state $s$ which atomic propositions $p \in \mathcal{P}$ are true in that state. If $T$ is an LTS, we use $\mathcal{S}(T)$ for its set of states; $\xrightarrow{a}_T$ or simply $\xrightarrow{a}$ for its transition relation and $T$ itself for its interpretation of atomic propositions.

[3] left-total means for all $s \in S$ there exists $s' \in S$ with $s \to s'$.

$$\begin{aligned}
[\![X]\!]\eta &= \eta(X) & & \\
[\![p]\!]\eta &= \{s \mid p \in T(s)\} & [\![\neg p]\!]\eta &= \{s \mid p \notin T(s)\} \\
[\![\phi_1 \vee \phi_2]\!]\eta &= [\![\phi_1]\!]\eta \cup [\![\phi_2]\!]\eta & [\![\phi_1 \wedge \phi_2]\!]\eta &= [\![\phi_1]\!]\eta \cap [\![\phi_2]\!]\eta \\
[\![\langle a \rangle \phi]\!]\eta &= \mathit{pre}(\xrightarrow{a})([\![\phi]\!]\eta) & [\![[a]\phi]\!]\eta &= \widetilde{\mathit{pre}}(\xrightarrow{a})([\![\phi]\!]\eta) \\
[\![\mu X.\phi]\!]\eta &= \mathit{lfp}(U \mapsto [\![\phi]\!]\eta[X := U]) & [\![\nu X.\phi]\!]\eta &= \mathit{gfp}(U \mapsto [\![\phi]\!]\eta[X := U])
\end{aligned}$$

**Fig. 1:** Set semantics of $\mu$-calculus formulas.

Fix a transition system $T$ and put $S = \mathcal{S}(T)$. For $\eta$ a finite partial function from $\mathcal{X}$ to $2^S$ with $FV(\phi) \subseteq \mathit{dom}(\eta)$ we define $[\![\phi]\!]\eta \subseteq S$ as in Figure 1.

The sets $\mathit{pre}(\xrightarrow{a})([\![\phi]\!]\eta)$ and $\widetilde{\mathit{pre}}(\xrightarrow{a})([\![\phi]\!]\eta)$ respectively denote the *preimage* and the *weakest precondition* of the set $[\![\phi]\!]\eta$ with respect to the binary relation $\xrightarrow{a}$; formally:

$$\begin{aligned}
&s \in \mathit{pre}(\xrightarrow{a})([\![\phi]\!]\eta) \textit{ iff } \exists t \in S.\ s \xrightarrow{a} t \textit{ and } t \in [\![\phi]\!]\eta \\
&s \in \widetilde{\mathit{pre}}(\xrightarrow{a})([\![\phi]\!]\eta) \textit{ iff } \forall t \in S.\ s \xrightarrow{a} t \textit{ implies } t \in [\![\phi]\!]\eta
\end{aligned}$$

Given the functional $F(U) = [\![\phi]\!]\eta[X := U]$, $\mathit{lfp}(F)$ and $\mathit{gfp}(F)$ respectively denote the least and the greatest fixpoints of $F$, with respect to the subset ordering on $2^S$. By Knaster-Tarski, these fixpoints exist, since $F$ is monotone.

**Proposition 1.** *$[\![QX.\phi]\!]\eta = [\![\phi[X := QX.\phi]]\!]\eta$.*

By the monotonicity of $F$, $\emptyset \subseteq F(\emptyset) \subseteq F^2(\emptyset) \subseteq \ldots$ and $S \supseteq F(S) \supseteq F^2(S) \supseteq \ldots$. Moreover, if $S$ is finite then we have

$$\begin{aligned}
[\![\mu X.\phi]\!]\eta &= \{s \in S \mid \exists t \leq |S|.\, s \in F^t(\emptyset)\}, \\
[\![\nu X.\phi]\!]\eta &= \{s \in S \mid \forall t \leq |S|.\, s \in F^t(S)\}.
\end{aligned}$$

Therefore, in case $S$ is finite, the iterative algorithm in Figure 2 computes $[\![\phi]\!]\eta$.

**Proposition 2.** *$[\![\phi]\!]\eta = \mathit{sem}(\phi, \eta)$.*

**Lemma 1.** *$s \notin [\![\phi]\!]\eta$ iff $s \in [\![\phi^*]\!]\eta'$, where $\eta'(X) = S \backslash \eta(X)$ and $\phi^*$ is the dual of $\phi$ given by*

$$\begin{aligned}
(X)^* &= X & & \\
(p)^* &= \neg p & (\neg p)^* &= p \\
(\phi_1 \wedge \phi_2)^* &= \phi_1^* \vee \phi_2^* & (\phi_1 \vee \phi_2)^* &= \phi_1^* \wedge \phi_2^* \\
([a]\phi)^* &= \langle a \rangle \phi^* & (\langle a \rangle \phi)^* &= [a]\phi^* \\
(\mu X.\phi)^* &= \nu X.\phi^* & (\nu X.\phi)^* &= \mu X.\phi^*
\end{aligned}$$

$$
\begin{aligned}
sem(X,\eta) &= \eta(X)\\
sem(p,\eta) &= T(p)\\
sem(\neg p,\eta) &= S \setminus T(p)\\
sem(\mu X.\phi,\eta) &= \textsc{iter}_X(\phi,\eta,\emptyset)\\
sem(\nu X.\phi,\eta) &= \textsc{iter}_X(\phi,\eta,S)\\
sem(\phi_1 \wedge \phi_2,\eta) &= sem(\phi_1,\eta) \cap sem(\phi_2,\eta)\\
sem(\phi_1 \vee \phi_2,\eta) &= sem(\phi_1,\eta) \cup sem(\phi_2,\eta)\\
sem([a]\phi,\eta) &= \widetilde{pre}(\xrightarrow{a})(sem(\phi,\eta))\\
sem(\langle a\rangle\phi,\eta) &= pre(\xrightarrow{a})(sem(\phi,\eta))
\end{aligned}
$$

$$
\begin{aligned}
\textsc{iter}_X(\phi,\eta,U) = &\ \textit{if } U = U' \textit{ then } U \textit{ else } \textsc{iter}_X(\phi,\eta,U')\\
&\quad \textit{where } U' := sem(\phi,\eta[X := U])
\end{aligned}
$$

**Fig. 2:** Fixpoint iteration for computing the semantics of $\mu$-calculus formulas.

## 2.2 Ordered Systems of Equations

We now use an alternate representation of $\mu$-calculus formulas considering them to be an *ordered system of equations* [26]. Under this point of view, a formula $\phi$ is represented by a set of equations $(X = \phi_X)_{X\in\mathcal{X}}$, with one formula $\phi_X$ for each variable $X$ occurring in $\phi$, together with a strict partial order of variables $\succ \subseteq \mathcal{X} \times \mathcal{X}$.

To do this, we assume that every fixpoint quantifier binds a different variable; if needed this can be ensured by $\alpha$-renaming. For example, we replace $\mu X.X \wedge \nu X.X$ with $\mu X.X \wedge \nu Y.Y$. For each variable $X$ we denote $q_X \in \{\mu,\nu\}$ the kind of quantifier that it stems from.

We then replace each fixpoint formula by the (unique) variable it introduces and thereafter give for each fixpoint formula a defining equation. Formally, for any subformula $\psi$ of $\phi$ let $\hat{\psi}$ denote the formula obtained by replacing each fixpoint subformula by the variable it binds. The equation system then contains one equation $X = \hat{\psi}$ for each fixpoint subformula $QX.\psi$, with $q_X = Q$.

In parallel, we build the strict partial order $\succ$ of variables. For each variable $X$ bound by a fixpoint subformula $QX.\psi$ and each variable $Y$ bound by a fixpoint subformula of $\psi$, i.e. all $Y$ bound below $X$, we set $X \succ Y$.

For example let

$$\phi := \nu Z.(b \vee (\mu X.X \vee [a]Z)) \wedge [a]Z$$

We have $\hat{\phi} = (b \vee (\mu X.X \vee [a]Z)) \wedge [a]Z = (b \vee X) \wedge [a]Z$ so the equations are

$$Z = (b \vee X) \wedge [a]Z \tag{i}$$
$$X = X \vee [a]Z \tag{ii}$$

Moreover, $q_Z = \nu$, $q_X = \mu$, and $Z \succ X$.

The order $\succ$ is relevant for the restoration of the original formula: had we instead set $X \succ Z$, we would retrieve

$$\mu X.X \vee [a](\nu Z.(b \vee X) \wedge [a]Z).$$

It is now clear that such systems of equations together with $\succ$ are in 1-1 correspondence with formulas.

In case the formula does not start with a quantifier, a fresh variable needs to be introduced and bound to the formula in the first place. Since this variable is not used anywhere else, either $\mu$ or $\nu$ can be chosen.

This representation is advantageous for our implementation as it avoids the need for syntactic substitution of fixpoint formulas for their variables.

We extend $[\![-]\!]\eta$ to formulas $\psi$ appearing in a given equation system according to the following clause, under the condition that $\{ X \mid X \succ Y, Y \in FV(\psi) \} \subseteq \mathrm{dom}(\eta)$, i.e. all variables of higher priority reachable from the right hand side of the equation are already bound in $\eta$.

$$[\![X]\!]\eta = \begin{cases} \mathit{lfp}(U \mapsto [\![\phi_X]\!]\eta[X{:=}U]) & \text{if } X \notin \mathrm{dom}(\eta) \text{ and } q_X = \mu \\ \mathit{gfp}(U \mapsto [\![\phi_X]\!]\eta[X{:=}U]) & \text{if } X \notin \mathrm{dom}(\eta) \text{ and } q_X = \nu \end{cases}$$

In particular, the following is then obvious, starting from an empty environment:

**Lemma 2.** *Let $\phi$ be a formula and let $X$ be the toplevel variable of the representation of $\phi$ as an equation system. Then $[\![X]\!] = [\![\phi]\!]$.*

*Remark* It is possible to extend the semantics to the case where the relations $\xrightarrow{a}$ are not necessarily total: The semantics carries over without changes.

The restriction to total relations is a standard one and it is vindicated by the following translation from the general case to the one treated here:

Given a LTS $T$ with a not necessarily total $\xrightarrow{a}$ we build a new LTS $T'$ with an additional distinguished state, $\mathcal{S}(T') = \mathcal{S}(T) \cup \{\mho\}$, then extend $\xrightarrow{a}$ with extra edges from any state to $\mho$ ($\xrightarrow{a}_{T'} = \xrightarrow{a}_T \cup \{ (s, \mho) \mid s \in \mathcal{S}(T') \}$) so that $\xrightarrow{a}_{T'}$ now is total. We also add a proposition $p_\mho$ which is true at state $\mho$ and nowhere else.

We can now define a translation $\widehat{\phi}$ for formulas by setting

$$\widehat{\langle a \rangle \phi} = \langle a \rangle \neg p_\mho \wedge \widehat{\phi} \qquad \widehat{[a]\phi} = [a] p_\mho \vee \widehat{\phi}.$$

The translation is homomorphically extended to all other connectives. It is then easy to see that $\forall s \in \mathcal{S}(T) : s \in [\![\phi]\!]_T \iff s \in [\![\widehat{\phi}]\!]_{T'}$.

## 3 Parity Games

A *parity game* is given by the following data:

- a (finite or infinite) set of positions $Pos$ partitioned into proponent's (Player 0) and opponent's (Player 1) positions: $Pos = Pos_0 + Pos_1$;
- a left-total edge relation $\rightarrow \subseteq Pos \times Pos$;
- a function $\Omega \in Pos \rightarrow \mathbb{N}$ with a finite range; we call $\Omega(p)$ the priority of position $p$.

The players move a token along the edge relation $\rightarrow$. When the token is on a position in $Pos_0$ then proponent decides where to move next and likewise for opponent.

In order to formalize the notion of "to decide" we must introduce strategies. Formally, a strategy for a player $i \in \{0,1\}$ is a function $\sigma$ that for any nonempty string $\vec{p} = p(0)\dots p(n)$ over $Pos$ such that $p(k) \rightarrow p(k+1)$ for $k = 0\dots n-1$ and $p(n) \in Pos_i$ associates a position $\sigma(\vec{p}) \in Pos$ such that $p(n) \rightarrow \sigma(\vec{p})$.

Given a starting position $p$ and strategies $\sigma_0$ and $\sigma_1$ for the two players one then obtains an infinite sequence of positions (a "play") $p(0), p(1), p(2), \dots$ by

$$\begin{aligned} p(0) &= p \\ p(n+1) &= \sigma_i(p(0)\dots p(n)) \quad \text{where } p(n) \in Pos_i \end{aligned}$$

We denote this sequence by $play(p, \sigma_0, \sigma_1)$.

The play is won by proponent (Player 0) if the largest number that occurs infinitely often in the sequence $\Omega(play(p, \sigma_0, \sigma_1))$ is even and it is won by opponent if that number is odd. Note that $\Omega(-)$ is applied component-wise and that a largest priority indeed exists since $\Omega$ has finite range.

Player $i$ wins from position $p$ if there exists a strategy $\sigma_i$ for Player $i$ such that for all strategies $\sigma_{1-i}$ of the other player (Player $1-i$) Player $i$ wins $play(p, \sigma_0, \sigma_1)$. We write $W_i$ for the set of positions from which Player $i$ wins.

A strategy $\sigma$ is *positional* if $\sigma(p(0)\dots p(n))$ only depends on $p(n)$. Player $i$ *wins positionally* from $p$ when the above strategy $\sigma_i$ can be chosen to be positional.

The following is a standard result [19].

**Theorem 1.** *Every position $p$ is either in $W_0$ or in $W_1$ and Player $i$ wins positionally from every position in $W_i$.*

*Example 1.* Fig. 3 contains a graphical display of a parity game. Positions in $Pos_0$ and $Pos_1$ are represented as circles and boxes, respectively, and labelled with their priorities. Formally, $Pos = \{a, b, c, d, e, f, g, h, i\}$; $Pos_0 = \{b, d, f, h\}$; $Pos_1 = \{a, c, e, g, i\}$; $\Omega(a) = 3, \dots$, and $\rightarrow = \{(a, b), (b, f), \dots\}$.

In the right half of Fig. 3 the winning sets are indicated and corresponding positional winning strategies are given as fat arrows. The moves from positions that are not in the respective winning set are omitted but can of course be filled-in in an arbitrary fashion.

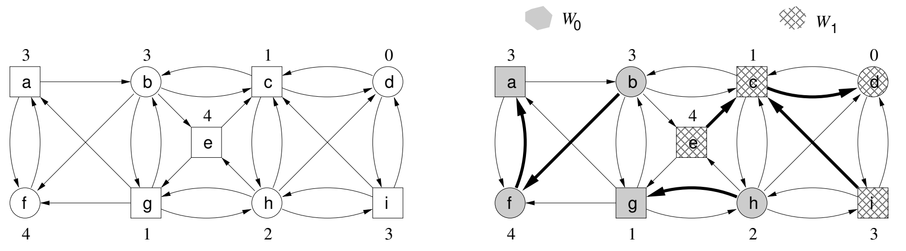


**Fig. 3:** A parity game and its decomposition into winning sets.

## 3.1 Certification of Winning Strategies

Given a parity game with finitely many positions, presented explicitly as a finite labelled graph, and a partition of $Pos$ into $V_0$ and $V_1$ we are now looking for an easy-to-verify certificate as to the fact that $V_0 = W_0$ and $V_1 = W_1$.

In essence, such a certificate will consist of a positional strategy $\sigma_i$ for each Player $i$ such that $i$ wins using $\sigma_i$ from every position $p$ in $V_i$. Clearly, this implies $V_i = W_i$ and the above theorem asserts that in principle such certificates always exist when $V_i = W_i$. However, it remains to explain how we can check that a given positional strategy $\sigma_i$ wins from a given position $p$.

We first note that for this it is enough that it wins against any adversarial positional strategy because the "optimal" counterstrategy, i.e., the one that wins from all adversarial winning positions is positional (by Theorem 1). Thus, given a positional strategy $\sigma_i$ for Player $i$ we can remove all edges from positions $p' \in Pos_i$ that are not chosen by the strategy and in the remaining game graph look for a cycle whose largest priority has parity $1 - i$ and is reachable from $p$. If there is such a cycle then the strategy was not good and otherwise it is indeed a winning strategy for Player $i$.

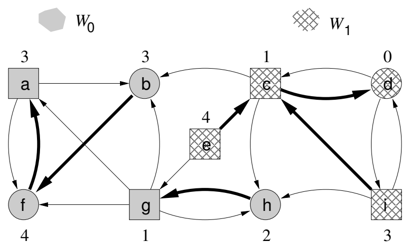
Naive enumeration of all cycles in the graph will result in having to check exponentially many cycles in the worst-case. However, the check can be performed in polynomial time [17], using the standard algorithm for nonemptiness of Streett automata [2] of which the problem at hand is actually an instance. This algorithm uses a decomposition of the graph into nontrivial strongly connected components (SCC).

If every reachable SCC only has positions whose priority has parity $i$ then obviously the strategy is good for Player $i$. Next, if there is a reachable SCC where the *highest* priority has parity $1-i$, the strategy is bad, since this particular position can be reached infinitely often.

Otherwise, the highest priority in each SCC has parity $i$ and of course player $1-i$ can win only if it is possible for them to avoid those nodes. Thus, we remove those nodes and decompose the resulting graph in SCCs again and start over.

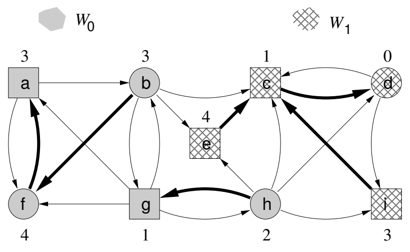
For our implementation, we use a variant of this algorithm based on Dijkstra's algorithm for SCC as presented by [10, 9]. In contrast to other efficient algorithms for this problem (such as [15]), it has the benefit of being *on-the-fly* and does not require precomputation of the parity game graph. The checking algorithm is described in more detail in Section 6.

*Example 2.* After removing the edges not taken by Player 0 according to their purported winning strategy we obtain the following graph:



We see that the two reachable SCC from $W_0$ are $\{a, b, f\}$ and $\{g, h\}$. The first one contains the cycles $a, f$ and $a, b, f$ which both have largest priority 4. The other one is itself a cycle with largest priority 2.

Likewise, adopting the viewpoint of Player 1, after removing the edges not taken by their strategy we obtain



and find the reachable (from $W_1$) SCCs to be $\{c, d, i\}$. The only cycles therein are $d, e$ and $d, e, i$. Both are good for Player 1.

## 4 Game-theoretic Characterization of $\mu$-Calculus

Fix an LTS $T$ and a $\mu$-calculus formula $\phi$. We first translate $\phi$ into an equation system (as explained in Section 2.2) over the variables $\mathcal{X}$ written as $X = \phi_X$ where $X \in \mathcal{X}$ and $q_X \in \{\mu, \nu\}$.

We also fix a function $\Omega : \mathcal{X} \to \mathbb{N}$ such that

- $q_X = \mu \Rightarrow \Omega(X)$ odd;
- $q_X = \nu \Rightarrow \Omega(X)$ even;
- $X \succ Y \Rightarrow \Omega(X) > \Omega(Y)$

If in addition $\eta$ is an environment with $\mathrm{dom}(\eta) \subseteq \mathcal{X}$ we define the game $G(T, \phi, \eta)$ as follows:

Positions are pairs $(s, \psi)$ where $s \in S$ and $\psi$ is a subformula of the right-hand sides of the equation system and $FV(\psi) \subseteq dom(\eta)$. In positions of the form $(s, \psi)$ where $\psi$ starts with $\vee$ or $\langle a \rangle$, it is proponent's (Player 0) turn. The possible moves for proponent to choose from are:

$$\begin{aligned} (s, \psi_1 \vee \psi_2) &\rightsquigarrow (s, \psi_1) \\ (s, \psi_1 \vee \psi_2) &\rightsquigarrow (s, \psi_2) \\ (s, \langle a \rangle \psi) &\rightsquigarrow (t, \psi) \qquad \text{where } s \xrightarrow{a}_T t. \end{aligned}$$

In positions of the form $(s, \psi)$ where $\psi$ starts with $\wedge$ or $[a]$ it is the opponent's turn. The possible moves for opponent to choose from are:

$$\begin{aligned} (s, \psi_1 \wedge \psi_2) &\rightsquigarrow (s, \psi_1) \\ (s, \psi_1 \wedge \psi_2) &\rightsquigarrow (s, \psi_2) \\ (s, [a] \psi) &\rightsquigarrow (t, \psi) \qquad \text{where } s \xrightarrow{a}_T t. \end{aligned}$$

From all other positions there is exactly one move so it does not matter to which player they belong. We fix them to be proponent's positions for definiteness. These unique moves are:

$$\begin{aligned} (s, X) &\rightsquigarrow (s, \phi_X) && \text{when } X \notin \mathrm{dom}(\eta) \\ (s, X) &\rightsquigarrow (s, X) && \text{when } X \in \mathrm{dom}(\eta) \\ (s, p) &\rightsquigarrow (s, p) \\ (s, \neg p) &\rightsquigarrow (s, \neg p) \end{aligned}$$

The priorities $\Omega(s, \phi)$ on these positions are defined as follows:

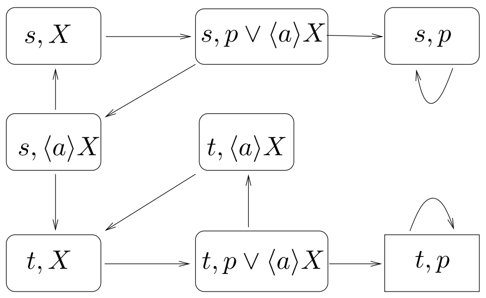
$$\begin{aligned} \Omega(s, p) &= \begin{cases} 0 & \text{if } p \in T(s) \\ 1 & \text{if } p \notin T(s) \end{cases} \\ \Omega(s, \neg p) &= \begin{cases} 1 & \text{if } p \in T(s) \\ 0 & \text{if } p \notin T(s) \end{cases} \\ \Omega(s, X) &= \begin{cases} \Omega(X) & \text{if } X \notin \mathrm{dom}(\eta) \\ 0 & \text{if } s \in \eta(X) \\ 1 & \text{if } s \notin \eta(X) \end{cases} \\ \Omega(s, \phi) &= 0 \quad \text{otherwise} \end{aligned}$$

The cases for predicates $p$, $\neg p$ and concrete sets $X$, i.e., where $X \in \mathrm{dom}(\eta)$ are clear. They are winning positions iff the associated state $s$ satisfies the corresponding predicate.

The variables $X \notin \mathrm{dom}(\eta)$ on the other hand are understood as abbreviations of the fixpoint formula they represent. Upon reaching such a position the fixpoint is unrolled and such unrolling is signalled by the priority $\Omega(X)$.

*Example 3.* Let $\phi = \mu X.\, p \vee \langle a \rangle X$ which asserts that a state where $p$ is true can be reached.

Define the transition system $T$ by $\mathcal{S}(T) = \{s, t\}$ and $T(s) = \emptyset$ and $T(t) = \{p\}$ and $\xrightarrow{a}_T = \{\, (s, s), (s, t), (t, t) \,\}$. The associated game graph is as follows:



The priorities of the positions labelled $(s, X), (t, X), (s, p)$ are 1; the priorities of the four other positions are 0.

Player 0 wins from every position except $(s, p)$. The winning strategy moves to $(s, \langle a \rangle X)$ and then $(t, X)$ and then $(t, p)$. Note that a strategy that moves from $(s, \langle a \rangle X)$ to $(s, X)$ loses even though it never leaves the winning set $W_0$. Thus, in order to compute winning strategies it is not enough to choose any move that remains in the winning set.

**Theorem 2.** *Fix a formula $\phi_0$ and an environment $\eta$.*
*If $s \in [\![\phi_0]\!]\eta$ then proponent wins $G(T, \eta)$ from $(s, \phi_0)$.*

Before proving this, we note that the converse is in this case actually a relatively simple consequence.

**Corollary 1.** *If proponent wins $G(T, \eta)$ from $(s, \phi)$ then $s \in [\![\phi]\!]\eta$.*

*Proof.* Suppose that proponent wins $G(T, \eta)$ from $(s, \phi)$ and $s \notin [\![\phi]\!]\eta$. We then have $s \in [\![\phi^*]\!]\eta'$ using Lemma 1 for the formal dualisation for formulas and complementation for environments. Thus, by the theorem, proponent wins $G(T, \eta')$ from $(s, \phi^*)$. However, it is easy to see that a winning strategy for proponent in $G(T, \eta')$ from $(s, \phi^*)$ is tantamount to a winning strategy for opponent in $G(T, \eta)$ from $(s, \phi)$; so we get a contradiction using Theorem 1. $\square$

*Proof (of Theorem 2).* The proof of Theorem 2 now works by structural induction on the equation system generated by $\phi_0$. We note that this induction preserves the invariant $\{\, X \mid X \succ Y, Y \in FV(\phi) \,\} \subseteq \mathrm{dom}(\eta)$, such that $[\![\phi]\!]\eta$ is always well-defined.

For a variable $X$, there are three cases, the latter ones are the interesting ones, as $X$ denotes a fixpoint there:

(i) $X \in \mathrm{dom}(\eta)$, then obviously $G(T, \eta)$ agrees with $[\![X]\!]\eta$.

(ii) $X \notin \mathrm{dom}(\eta)$ and $q_X = \mu$. Then we define

$$U := \{\, t \mid \textit{proponent wins } G(T,\eta) \textit{ from } (t, X) \,\}.$$

We must show that $[\![\phi_X]\!]\eta \subseteq U$. By definition of $[\![\phi_X]\!]\eta$ it suffices to show that $[\![\phi_X]\!]\eta[X \mapsto U] \subseteq U$. Thus, suppose that $t \in [\![\phi_X]\!]\eta[X \mapsto U]$. By the induction hypothesis this means that proponent wins $G(T, \eta[X \mapsto U])$ from $(t, \phi_X)$. (Now $\eta(X)$ is bound while recursing the subformulas of lower priority, preserving the condition on $\eta$.)

Call the corresponding winning strategy $\sigma$. We should prove that proponent also wins from $(t, X)$. We move to $(t, \phi_X)$ and then play according to $\sigma$. If we never reach a position $(t', X)$, then by the definition of $G(T, \eta[X \mapsto U])$ we actually have won $G(T, \eta)$.

The first time, if ever, that we reach a position $(t', X)$, we know by the definition of $U$ that $t' \in U$ and therefore we win $G(T, \eta)$ from $(t', X)$, so we abandon $\sigma$ and continue play according to the strategy embodied in the latter statement. This then ensures winning from $(t, X)$ since finite prefixes do not affect the winning condition.

(iii) $X \notin \mathrm{dom}(\eta)$ and $q_X = \nu$. Let $U := [\![X]\!]\eta$ $(= [\![\nu X.\phi_X]\!]\eta)$. We define a winning strategy for positions of the form $(t, X)$ where $t \in U$ as follows. First, we move (forcedly) to $(t, \phi_X)$. We know that $t \in [\![\phi_X]\!]\eta[X \mapsto U]$ by unwinding so that, inductively, we have a strategy that allows us to either win right away, or move to another position $(t', X)$ where $t' \in U$ and all priorities encountered on the way are smaller than the one of $X$ due to the definition of priorities, and since all higher occurring priorities are bound in $\eta$, thus not resulting in a loop.

We start over and unless we eventually do win right away at some point we would have seen the priority of $X$ itself infinitely often which is the largest and even. $\square$

We remark that while the previous result is well-known the proof presented here is quite different from the ones in the standard literature, e.g. [4], which use the order-theoretic concept of signatures, also known as rankings. Those proofs are less compositional than ours, in the sense that they do not proceed directly by structural induction on formulas but rather on the global development of all the fixpoints.

It is essentially this new compositional proof which allows us to instrument the usual fixpoint iteration so as to compute winning strategies alongside as we now detail.

## 5 Computing Winning Strategies via Fixpoint Iteration

### 5.1 Fixpoint Iteration

It is well-known that the fixpoint iteration in Figure 2 computes $[\![\phi]\!]\eta$ in the finite case. Our goal is to somehow instrument this algorithm so that it produces evidence in the form of a winning strategy. In instrumenting this algorithm to produce evidence in the form of a winning strategy it is not enough to simply compute the winning sets using SEM$(-,-)$ and then simply choose moves that do not leave the winning set. This is because of examples like 3 which show that a strategy that never leaves the winning set may nonetheless be losing.

Instead we will use the construction from the proof of Theorem 2. Some care needs to be taken with the exact setup of the input and output data formats; in particular, our algorithm will return partial winning strategies (that win on a subset of the whole winning set) but only require sets of states (rather than partial winning strategies) as the values of free variables.

## 5.2 Partial Winning Strategies

A *partial winning strategy* is a partial function $\Sigma$ mapping positions of the game $G(T,\eta)$ to elements of $S$ extended with $\{1,2,*\}$; it must satisfy the following conditions:

**STAR** If $\Sigma(\phi,s)=*$ then all immediate successors of $(\phi,s)$ are in $dom(\Sigma)$;
**OR** If $\Sigma(\phi,s)=i\in\{1,2\}$ then $\phi$ is of the form $\phi_1\vee\phi_2$ and $(\phi_i,s)\in dom(\Sigma)$;
**DIA** If $\Sigma(\phi,s)=s'\in S$ then $\phi$ is of the form $\langle a\rangle\psi$ and $s\xrightarrow{a}s'$ and $(\psi,s')\in dom(\Sigma)$.
**WIN** Player 0 wins from all the positions in $dom(\Sigma)$ and the obvious strategy induced by $\Sigma$ is a winning strategy for Player 0 from those positions.

Note that the empty function (denoted $\{\}$) is in particular a partial winning strategy. To illustrate the notation we describe a (partial) winning strategy for the entire winning set for Example 3:

$$\begin{array}{rlr}
\Sigma(\phi,s)=* & \quad\Sigma(\phi,t)=* & \\
\Sigma(P\vee\langle a\rangle\phi,s)=2 & \quad\Sigma(P\vee\langle a\rangle\phi,t)=1 & \\
\Sigma(\langle a\rangle\phi,s)=t & \quad\Sigma(P,t)=*\,, & \quad\text{and undefined elsewhere.}
\end{array}$$

So, $dom(\Sigma)=\{(\phi,s),\ldots,(P,t)\}$ and, indeed, Player 0 wins from all these positions by following the advice given by $\Sigma$. Of course, $\Sigma'(P,t)=*$ and undefined elsewhere is also a partial winning strategy albeit with smaller domain of definition.

*Updating of winning strategies.* Suppose that $\Sigma$ and $\Sigma'$ are partial winning strategies. A new partial winning strategy $\Sigma+\Sigma'$ with $dom(\Sigma+\Sigma')$ is defined by

$$(\Sigma+\Sigma')(\phi,s)=\mathit{if}\ (\phi,s)\in dom(\Sigma)\ \mathit{then}\ \Sigma(\phi,s)\ \mathit{else}\ \Sigma'(\phi,s).$$

**Lemma 3.** *$\Sigma+\Sigma'$ is a partial winning strategy and $dom(\Sigma+\Sigma')=dom(\Sigma)\cup dom(\Sigma')$*

*Proof.* A play following $\Sigma+\Sigma'$ will eventually remain in one of $\Sigma$ or $\Sigma'$; this, together with the fact that initial segments do not affect the outcome of a game implies the claim. □

## 5.3 Computing Winning Strategies by Fixpoint Iteration

For any LTS $T$, formula $\phi$ and environment $\eta$ with $dom(\eta) \supseteq FV(\phi)$ we define a partial winning strategy $\textsc{sem}(\phi)_\eta$ by the following clauses:

$$
\begin{aligned}
\textsc{sem}(X)_\eta &= \{\, (X,s) \mapsto * \mid s \in \eta(X) \,\} && \text{if } X \in \mathrm{dom}(\eta)\\
\textsc{sem}(p)_\eta &= \{\, (p,s) \mapsto * \mid p \in T(s) \,\}\\
\textsc{sem}(\neg p)_\eta &= \{\, (p,s) \mapsto * \mid p \notin T(s) \,\}\\
\textsc{sem}(\phi \wedge \psi)_\eta &= \textsc{sem}(\phi)_\eta + \textsc{sem}(\psi)_\eta\\
&\quad + \{\, (\phi \wedge \psi, s) \mapsto * \mid (\phi,s) \in \mathrm{dom}(\textsc{sem}(\phi)_\eta)\\
&\qquad\qquad \wedge (\psi,s) \in \mathrm{dom}(\textsc{sem}(\psi)_\eta) \,\}\\
\textsc{sem}(\phi \vee \psi)_\eta &= \textsc{sem}(\phi)_\eta + \textsc{sem}(\psi)_\eta\\
&\quad + \{\, (\phi \vee \psi, s) \mapsto 1 \mid (\phi,s) \in \mathrm{dom}(\textsc{sem}(\phi)_\eta) \,\}\\
&\quad + \{\, (\phi \vee \psi, s) \mapsto 2 \mid (\psi,s) \in \mathrm{dom}(\textsc{sem}(\psi)_\eta) \,\}\\
\textsc{sem}([a]\phi)_\eta &= \textsc{sem}(\phi)_\eta\\
&\quad + \{\, ([a]\phi, s) \mapsto * \mid (\phi,s) \in \mathrm{dom}(\textsc{sem}(\phi)_\eta) \,\}\\
\textsc{sem}(\langle a\rangle\phi)_\eta &= \textsc{sem}(\phi)_\eta\\
&\quad + \{\, (\langle a\rangle\phi, s) \mapsto s' \mid s \xrightarrow{a} s' \wedge (\phi,s') \in \mathrm{dom}(\textsc{sem}(\phi)_\eta) \,\}
\end{aligned}
$$

$$
\begin{aligned}
\textsc{sem}(X = \phi_X)_\eta &= \textsc{shift}(\nu X.\phi_X, \textsc{sem}(\phi_X)_{\eta[X:=sem(\phi_X,\eta)]}) && \text{if } q_X = \nu\\
\textsc{sem}(X = \phi_X)_\eta &= \textsc{shift}(\mu X.\phi_X, \textsc{iter}_X(\phi_X, \eta, \{\})) && \text{if } q_X = \mu
\end{aligned}
$$

$$
\begin{aligned}
\textsc{iter}_X(\phi,\eta,\Sigma) = {} & \text{let } \Sigma' := \textsc{sem}(\phi)_{\eta[X:=\{s \mid (\phi,s)\in\mathrm{dom}(\Sigma)\}]} \text{ in}\\
& \quad \text{if } \mathrm{dom}(\Sigma) = \mathrm{dom}(\Sigma') \text{ then } \Sigma \text{ else } \textsc{iter}_X(\phi,\eta,\Sigma')
\end{aligned}
$$

$$
\textsc{shift}(QX.\phi, \Sigma) = \Sigma + \{\, (QX.\phi, s) \mapsto * \mid (\phi,s) \in \mathrm{dom}(\Sigma) \,\}
$$

Of particular interest is the $\textsc{shift}$ function: since the only possible moves for $QX.\phi$ formulas are to move to the subformula $\phi$, we need to adjust the domain of the winning strategy under construction to only allow this move when the strategy will win for the subformula already.

Note how the fixpoint iteration in $\textsc{iter}_X$ stops when the domain of the partial winning strategy does not change anymore. Since the greatest fixpoint for $\nu X.\phi$ cannot be calculated from above in terms of winning strategies (which can only grow according to our definitions), the winning set (and thus, domain of the winning strategy) is computed using the set semantics $sem(-,-)$ instead.

The following Lemma and Theorem are now immediate from these definitions and Lemma 3.

**Lemma 4.** $\{\, s \mid (\phi,s) \in dom(\textsc{sem}(\phi)_\eta) \,\} \;=\; [\![\phi]\!]\eta$

**Theorem 3.** *$\textsc{sem}(\phi)_\eta$ is a winning strategy for $G(T,\phi,\eta)$.*

**Proposition 3.** *Given a formula $\phi$ with fixed alternation depth,* $\textsc{sem}(\phi)_\eta$ *can be computed in polynomial time in $|S|$ and in space $O(|S|^2|\phi|^2)$, where $|S|$ is the size of the state space and $|\phi|$ the length of the formula $\phi$.*

*Proof.* The computation of $\textsc{sem}(\phi)_\eta$ follows the one of $[\![\phi]\!]\eta$ hence the time bound. Just like in the usual implementations of fixpoint iteration one only needs to remember the result of the last iteration. Storing a single partial winning strategy requires space $O(|S|^2|\phi|)$ (associating at most one state to each pair of state and subformula) and the recursion stack during traversal of subformulas is limited by $|\phi|$ thus requiring us to store $O(\phi)$ partial winning strategies at any one time. This yields the announced space bound. $\square$

## 6 Implementation and Evaluation

We have developed an implementation [22] of both computation and checking of certificates in OCaml. Winning strategies are kept abstract, only exposing an `assoc` function to look up a possible move given a model and particular state. Computation of winning strategies happens by fixpoint iteration using a recursive function, just like presented in section 5.3.

Two algorithms for checking certificates are implemented: "Check", a naive, recursive one with worst-case exponential time, and "Check SCC", a more intricate one using strongly-connected components to detect cycles. Both algorithms operate *on-the-fly* and do not need to pre-compute or even keep the parity game graph in memory.

The algorithm "Check SCC" is a variant of Dijkstra's algorithm for detecting strongly connected components and can be found in [10, 9].

This algorithm works as follows [25]: During a depth-first search of the graph, we keep a stack of strongly connected components that have been found. Upon finding an edge back into a SCC that closes a cycle, we merge all SCC that are part of the cycle, since using the cycle we can now move from every SCC into any other, i.e. their union is actually one SCC.

We provide three benchmarks that give insight into the algorithms at work.

The "Flower" benchmark is a parity game (from [5]) translated into a $\mu$-calculus formula, which shows the exponential runtime of fixpoint iteration for $\mu$-calculus. However, the certificates can be checked in polynomial time.

The "Circle" benchmark measures the overhead of the algorithms. It consists of a single cycle that needs to be traversed to check for a reachability property. In this case, runtime is linear, and checking is very fast.

The "Braid" benchmark focuses on checking complexity. This family of graphs has exponentially many cycles, thus the simple checker requires exponential time. The SCC algorithm is not affected and checks these strategies in linear time.

## 7 Conclusion

Our main result is an effective and low overhead instrumentation of the usual fixpoint iteration of $\mu$-calculus model checking [3] for generating certificates or

**Table 1:** Runtimes on a AMD Phenom II X4 920 (2.80 GHz)

| Problem | States | sem [s] | SEM [s] | Check [s] | Check SCC [s] |
|---|---|---|---|---|---|
| Flower 8 | 16 | 0.179 | 0.203 | 0.009 | 0.040 |
| Flower 10 | 20 | 3.166 | 1.960 | 0.071 | 0.419 |
| Flower 12 | 24 | 32.269 | 11.688 | 0.287 | 2.061 |
| Flower 14 | 28 | 320.931 | 61.733 | 1.298 | 10.829 |
| Flower 16 | 32 | 3196.043 | 326.666 | 6.131 | 58.871 |
| Circle 100 | 100 | 0.003 | 0.001 | 0.001 | 0.001 |
| Circle 1000 | 1000 | 0.109 | 0.018 | 0.005 | 0.006 |
| Circle 10000 | 10000 | 15.763 | 3.398 | 0.054 | 0.057 |
| Circle 100000 | 100000 | 2027.584 | 811.041 | 0.581 | 0.582 |
| Braid 6 | 12 | 0.001 | 0.005 | 1.282 | 0.009 |
| Braid 8 | 16 | 0.002 | 0.003 | 31.062 | 0.013 |
| Braid 10 | 20 | 0.002 | 0.006 | 711.002 | 0.020 |
| Braid 100 | 200 | 0.663 | 0.993 | — | 3.674 |

counterexamples that are independently checkable in low polynomial time. The notion of *partial winning strategies* is central to our developments and also seems to be novel.

We have implemented our witness-generating algorithms and demonstrated the feasibility of our approach by means of a collection of benchmark examples. For simple formulas, manual inspection of the generated certificates yields counterexamples similar to those generated by SMV, but algorithmic approaches for extracting explicit counterexamples in general needs further investigation.

There are numerous applications for our certifying $\mu$-calculus model checker. In particular, it should be possible to generate checkable certificates for the bisimulation between programs and for model checking problems for both linear time temporal logics and computation tree logics [11] as the basis for assurance cases and certification arguments for safety-critical systems. Moreover, certificates for $\mu$-calculus model checking might also be used as the basis of symmetric abstraction-refinement-based model checking engines for the full $\mu$-calculus based on refining over-approximations using *spurious counterexamples* and relaxing under-approximations using *dubious witnesses* along the lines of [29, 30], for sending code together with proofs of arbitrary safety and liveness properties properties, which are then checked by code consumers according to the *proof-carrying code* paradigm of [21], and for synthesizing *correct-by-construction* controllers from these certificates [30].

Our developments may also form the underpinning for a sound integration of $\mu$-calculus model checking into other verification systems such as PVS [23]. Using Shankar's kernel of truth [28] approach, which is based on checking the verification and on verifying the checker, certificates are generated using an untrusted implementation of our $\mu$-calculus model checking algorithms, and certificates are then checked by means of an executable PVS function, which itself is verified in a trusted kernel of PVS.

## References

1. Biere, A., Zhu, Y., Clarke, E.: Multiple state and single state tableaux for combining local and global nodel checking. In: Olderog, E.R., Steffen, B. (eds.) Correct System Design, Lecture Notes in Computer Science, vol. 1710, pp. 163–179. Springer (1999)
2. Bloem, R., Gabow, H.N., Somenzi, F.: An algorithm for strongly connected component analysis in $n \log n$ symbolic steps. Formal Methods in System Design 28(1), 37–56 (2006)
3. Bradfield, J., Stirling, C.: Modal mu-calculi. Studies in Logic and Practical Reasoning 3, 721–756 (2007)
4. Bradfield, J., Stirling, C.: Modal logics and mu-calculi: an introduction. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebra, pp. 293–330. Elsevier (2001)
5. Buhrke, N., Lescow, H., Vöge, J.: Strategy construction in infinite games with streett and rabin chain winning conditions. In: Margaria, T., Steffen, B. (eds.) TACAS. Lecture Notes in Computer Science, vol. 1055, pp. 207–225. Springer (1996)
6. Clarke, E., Jha, S., Lu, Y., Veith, H.: Tree-like counterexamples in model checking. In: Logic in Computer Science, 2002. Proceedings. 17th Annual IEEE Symposium on. pp. 19–29. IEEE (2002)
7. Clarke, E., Grumberg, O., McMillan, K., Zhao, X.: Efficient generation of counterexamples and witnesses in symbolic model checking. In: Proceedings of the 32nd annual ACM/IEEE Design Automation Conference. pp. 427–432. ACM (1995)
8. Cleaveland, R.: Tableau-based model checking in the propositional mu-calculus. Acta Informatica 27(8), 725–747 (1990)
9. Duret-Lutz, A.: Contributions à l'approche automate pour la vérification de propriétés de systèmes concurrents. Phd thesis, Université Pierre et Marie Curie (Paris 6) (Jul 2007), `https://www.lrde.epita.fr/~adl/th.html`
10. Duret-Lutz, A., Poitrenaud, D., Couvreur, J.M.: On-the-fly emptiness check of transition-based Streett automata. In: Liu, Z., Ravn, A.P. (eds.) ATVA'09. Lecture Notes in Computer Science, vol. 5799, pp. 213–227. Springer (2009)
11. Emerson, E., Jutla, C., Sistla, A.: On model-checking for fragments of $\mu$-calculus. In: Courcoubetis, C. (ed.) Computer Aided Verification, Lecture Notes in Computer Science, vol. 697, pp. 385–396. Springer (1993)
12. Emerson, E., Jutla, C.: Tree automata, mu-calculus and determinacy. In: Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS'91). pp. 368–377. IEEE (1991)
13. Grädel, E.: Back and forth between logic and games. In: Apt, K., Grädel, E. (eds.) Lectures in Game Theory for Computer Scientists, pp. 99–138. Cambridge University Press (2011)
14. Gurfinkel, A., Chechik, M.: Proof-like counter-examples. In: Proceedings of the 9th international conference on Tools and algorithms for the construction and analysis of systems. pp. 160–175. Springer (2003)
15. Henzinger, M.R., Telle, J.A.: Faster algorithms for the nonemptiness of streett automata and for communication protocol pruning. In: in Scandinavian Workshop on Algorithm Theory. pp. 16–27. Springer (1996)
16. Hofmann, M., Rueß, H.: Certification for $\mu$-calculus with winning strategies. ArXiv e-prints (Jan 2014)

17. Jurdziński, M.: Algorithms for solving parity games. In: Apt, K., Grädel, E. (eds.) Lectures in Game Theory for Computer Scientists, pp. 74–98. Cambridge University Press (2011)
18. Kick, A.: Generation of counterexamples for the $\mu$-calculus. Tech. Rep. ira-tr-1995-37, Universität Karlsruhe, Germany (1995)
19. Martin, D.A.: Borel determinacy. The annals of Mathematics 102(2), 363–371 (1975)
20. Namjoshi, K.: Certifying model checkers. In: Computer Aided Verification. pp. 2–13. Springer (2001)
21. Necula, G.: Proof-carrying code. In: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 106–119. ACM (1997)
22. Neukirchen, C.: Computation of winning strategies for $\mu$-calculus by fixpoint iteration. Master's thesis, Ludwig-Maximilians-Universität München (Nov 2014)
23. Owre, S., Rushby, J.M., Shankar, N.: PVS: a prototype verification system. In: Automated Deduction—CADE-11, pp. 748–752. Springer (1992)
24. Peled, D., Pnueli, A., Zuck, L.: From falsification to verification. FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science pp. 292–304 (2001)
25. Renault, E., Duret-Lutz, A., Kordon, F., Poitrenaud, D.: Three SCC-based emptiness checks for generalized Büchi automata. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) Proceedings of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'13). Lecture Notes in Computer Science, vol. 8312, pp. 668–682. Springer (Dec 2013)
26. Seidl, H.: Fast and Simple Nested Fixpoints. Universität Trier, Mathematik/Informatik, Forschungsbericht 96-05 (1996)
27. Shankar, N., Sorea, M.: Counterexample-driven model checking (revisited version). Tech. Rep. SRI-CSL-03-04, SRI International (2003)
28. Shankar, N.: Rewriting, inference, and proof. In: Rewriting Logic and Its Applications, pp. 1–14. Springer (2010)
29. Sorea, M.: Dubious witnesses and spurious counterexamples (2005), `http://www.cs.man.ac.uk/~msorea/talks/york.pdf`, uK Model Checking Days, York
30. Sorea, M.: Verification of real-time systems through lazy approximations. Ph.D. thesis, University of Ulm, Germany (2004)
31. Stirling, C., Walker, D.: Local model checking in the modal mu-calculus. In: TAPSOFT'89. pp. 369–383. Springer (1989)
32. Streett, R., Emerson, E.: The propositional mu-calculus is elementary. In: Paredaens, J. (ed.) Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 172, pp. 465–472. Springer (1984)
33. Tan, L., Cleaveland, R.: Evidence-based model checking. In: Brinksma, E., Larsen, K. (eds.) Computer Aided Verification, Lecture Notes in Computer Science, vol. 2404, pp. 641–680. Springer (2002)
34. Vardi, M., Wilke, T.: Automata: From logics to algorithms. In: WAL. pp. 645–753 (December 2007)
35. Winskel, G.: A note on model checking the modal $\nu$-calculus. Theoretical Computer Science 83(1), 157–167 (1991)