Generalized μ-Automata

Satisfaction and Refinement Games

Student Research Project

Jan Waller

Christian-Albrechts-University Kiel, Germany
jwa@informatik.uni-kiel.de

Abstract. Generalized μ-Automata are a new kind of model, used for the abstraction of Kripke structures. They are based upon a combination of pre-abstraction and post-abstraction, developed and sketched by Fecher and Huth. Satisfaction of these new models with the modal μ-calculus is introduced with the help of satisfaction games between generalized μ-automata and the modal μ-calculus in its alternate form of alternating tree automata. Finally a notion of refinement (abstraction) between two generalized μ-automata is introduced via refinement games.

Key words: Generalized μ-Automata, Satisfaction Games, Refinement Games, Abstraction

1 Introduction

Model checking has its origins in the early 1980s, when Clarke and Emerson [1] as well as Queille and Sifakis [7] introduced a new algorithmic approach for the verification of systems. One formally defines satisfaction as an relation $M \models \phi$, where M is a mathematical model of a system and ϕ is a property, which has to be shown, encoded within a formal language, such as the modal μ-calculus [6]. Problems arise with the size of the state space of M, which is often infinite or exponential in its description. Instead of directly calculating $M \models \phi$, one employs abstraction techniques, i.e. one constructs an abstract model A from the specification of M such that $A \models \phi$ implies $M \models \phi$.

Fecher and Huth [4] worked upon the abstraction technique named *pre-abstraction* and developed a new abstraction techniques called *post-abstraction*. Depending on the model M and proposition ϕ the calculation of pre- or post-abstraction may be preferable to each other. They sketched how a combination of both techniques may result in a more precise abstraction without increasing the cost of the abstraction synthesis too much.

Contribution. In this student research project I present the combination of both techniques mentioned above, which results in the model of generalized μ-automata. In addition a sound definition of satisfaction is given with the help of satisfaction games between this new model and alternating tree automata. Further a definition of refinement is given with the help of refinement games between two generalized μ-automata, which is reflexive, transitive and sound.

Related work. Some other abstraction techniques were developed by Graf and Saïdi [5], who introduced predicate abstraction (an abstract model A is constructed from a partition of the state space of M such that certain properties of A also hold in M), and Dams [2], who introduced the notion of an abstraction relation between states of the abstract model A and states of M together with the notion of precision.

The notion of pre-abstraction is based upon work by de Alfaro et al. [3], who defined precision for the alternating time μ-calculus for may- and must-transitions using a parity game in which the Refuter can replace a concrete state with any state in the same partition. Pre-abstraction is a transformation of this game into a satisfaction game, restricted to the modal μ-calculus and partition based abstraction.

Outline. In section 2 of this work Kripke structures and alternating tree automata are briefly introduced. Furthermore satisfaction between Kripke structure and alternating tree automata is defined with the help of their standard satisfaction game. Section 3 briefly summarizes the notion of abstraction as well as the techniques of pre-abstraction and post-abstraction and their respective models named *generalized Kripke modal transition system* and *μ-automaton* as well as their particular satisfaction games. The new model named *generalized μ-automaton* is defined within section 4. Additionally satisfaction of this model is defined with the help of satisfaction games. Finally section 5 correlates two generalized μ-automata with the notion of refinement and abstraction respectively, formalized through refinement games.

2 Preliminaries

In the following, $\mathbb{P}(S)$ denotes the power set of a set S. Let $\mathbb{N}$ denote the natural numbers including 0, $\mathbb{N}_{\geq 1}$ the natural number greater than 0. Let 0 be even. For a relation $\rho \subseteq B \times C$ with $X \subseteq B$ I will write $X.\rho$ for $\{c \in C \mid \exists b \in X : (b, c) \in \rho\}$.

Let AP be a finite set of atomic propositions, such that $true, false \in \text{AP}$ and if $p \in \text{AP}$, then $\neg p \in \text{AP}$.

Note 1. Since all games presented here are kinds of parity games, there exists a winning strategy for a player iff the Player has a history independent winning strategy.

Most of the following definitions and examples are taken from [4].

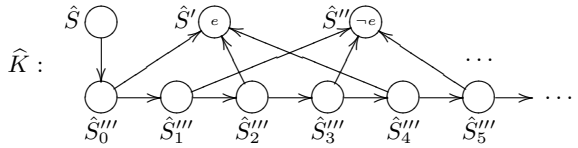

Fig. 1. A *Kripke structure*. The propositions $L(w) \subseteq \mathrm{AP}$ at a world w are depicted within world borders. Arrows $w \to w'$ denote $(w, w') \in R$. World names are depicted close to the corresponding world.

2.1 Kripke Structures

In accordance to Fecher and Huth [4] I will not consider action labels on models, thus the concrete models considered here are Kripke structures over a finite set of propositions AP. Such Kripke structures are directed graphs. Arcs represent *transitions* and vertices represent worlds. Propositions taken from AP are assigned as labels to each vertex.

Definition 1 (Kripke structure). *A* Kripke structure K *over* AP *is a tuple* $K = (W, \hat{w}, R, L)$ *where*

- W *is a (nonempty) set of worlds,*
- $\hat{w} \in W$ *is an initial world,*
- $R \subseteq W \times W$ *is its world transition relation, and*
- $L\colon W \to \mathbb{P}(\mathrm{AP})$ *is its labeling function.*

K *is finite whenever* W *is.*

Example 1. Consider a simple program to determine if a given number ($\in \mathbb{N}$) is even. The idea is to start with 0 which is even and increase by 1 switching from even to odd or from odd to even. Now one just continues to increase until one reaches the given number and checks whether the current status is even or odd.

A possible Kripke structure $\widehat{K}$ to model such a system is illustrated in Fig. 1. Here the set of worlds W consists of an initial world $\hat{S}$ which represents initiating the program, the set of worlds $\hat{S}_i'''$ for $i \in \mathbb{N}$ and the two worlds $\hat{S}'$ and $\hat{S}''$ which represent printing the result for even or odd numbers. Each world corresponds to an assignment to variables. Here only one variable is used. It is undefined in $\hat{S}$, $\hat{S}'$ and $\hat{S}''$ and set to i in $\hat{S}_i'''$ for $i \in \mathbb{N}$. Further there are two propositions e and $\neg e \in AP$, where e means even and $\neg e$ odd. Accordingly $\{e\}$ and $\{\neg e\}$ are assigned to $\hat{S}'$ and $\hat{S}''$ via L. The transitions R are depicted in the figure.

To check a given number n one starts at $\hat{S}$, takes the transition to $\hat{S}_0'''$ and continues onward until the counting variable is set to n, thus to $\hat{S}_n'''$. Then one takes the transition to $\hat{S}'$ or $\hat{S}''$ respectively and gets the result from the labeling.

Formally the Kripke Structure $\widehat{K}$ is defined as:
$\widehat{K} = (W, \hat{S}, R, L)$ with $\mathrm{AP} = \{true, false, e, \neg e\}$ and

- $W = \{\hat{S}, \hat{S}', \hat{S}''\} \cup \{\hat{S}_i''' \mid i \in \mathbb{N}\}$,
- $R = \{(\hat{S}, \hat{S}_0''')\} \cup \{(\hat{S}_{2i}''', \hat{S}') \mid {}_{i \in \mathbb{N}}\} \cup \{(\hat{S}_{2i+1}''', \hat{S}'') \mid {}_{i \in \mathbb{N}}\} \cup \{(\hat{S}_i''', \hat{S}_{i+1}''') \mid {}_{i \in \mathbb{N}}\}$,
- $L\colon W \to \mathbb{P}(\mathrm{AP})$ with $\hat{S}' \mapsto \{e\}; \hat{S}'' \mapsto \{\neg e\}$; for all $w \in W \setminus \{\hat{S}', \hat{S}''\} : w \mapsto \emptyset$.

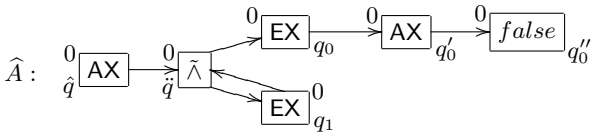

Fig. 2. An *alternating tree automaton*. State names and acceptance numbers are depicted close to corresponding states. A depiction of the transition function is within the state borders in conjunction with the transition arrows.

2.2 Alternating Tree Automata

The modal μ-calculus is a logic to express properties of Kripke structures. Here I use the modal μ-calculus in its equivalent form of alternating tree automata. This equivalence is shown by Wilke [8].

Definition 2 (Alternating tree automaton). *An* alternating tree automaton *is a tuple* $A = (Q, \hat{q}, \delta, \Theta)$ *such that*

- Q *is a finite, nonempty set of states,*
- $\hat{q} \in Q$ *is an initial state,*
- δ *is a transition function, which maps each automaton state* ($\in Q$) *to one of the following forms:* $p \mid \neg p \mid q' \mid q'\tilde{\wedge}q'' \mid q'\tilde{\vee}q'' \mid \mathsf{EX}\, q' \mid \mathsf{AX}\, q'$ *(where* $p, \neg p \in \mathrm{AP}; q', q'' \in Q$*), and*
- $\Theta\colon Q \to \mathbb{N}$ *is an acceptance condition, which assigns an acceptance number to each state.*

Example 2. Considering the Kripke structure given in Example 1 on the previous page one might be interested whether the given system is able to terminate for all input numbers ($\in \mathbb{N}$). So, at each world $\hat{S}_i'''$ there must be a transition leading to one of the two final worlds and one transition to next world $\hat{S}_{i+1}'''$. In other words: After any transition, ($*$) (i) there is a transition such that no further transition is possible; and (ii) there is a transition such that ($*$) holds again. Note that these transitions are inside the Kripke structure.

An example of an alternating tree automaton to incorporate ($*$) is depicted in Fig. 2. After any transition ($\hat{q}$), there are two transitions ($\ddot{q}$). One such that no further transitions are possible (q_0) and one transition such that the property described by $\ddot{q}$ holds again (q_1).

Note that the term transitions refers to the Kripke structure. Every time the transition function of the alternating tree automata refers to EX or AX, a transition in the Kripke structure happens. This will be clarified further by the section satisfaction games.

Formally the alternating tree automata $\widehat{A}$ is defined as:
$\widehat{A} = (Q, \hat{q}, \delta, \Theta)$ with $\mathrm{AP} = \{true, false, e, \neg e\}$ and

- $Q = \{\hat{q}, \ddot{q}, q_0, q_0', q_0'', q_1\}$,
- $\delta\colon \hat{q} \mapsto \mathsf{AX}\, \ddot{q}; \ddot{q} \mapsto q_0 \tilde{\wedge} q_1; q_0 \mapsto \mathsf{EX}\, q_0'; q_0' \mapsto \mathsf{AX}\, q_0''; q_0'' \mapsto false; q_1 \mapsto \mathsf{EX}\, \ddot{q}$,
- $\Theta\colon Q \to \mathbb{N}$ with $q \mapsto 0$ for all $q \in Q$.

2.3 Satisfaction Games

Satisfaction games are a special kind of parity game played by two players. One player is the *Verifier*, the other one is the *Refuter*. The game is played on a field consisting of two automata, one representing the model of a system (e.g. a Kripke structure $K = (W, \hat{w}, R, L)$) and one being an alternating tree automaton $A = (Q, \hat{q}, \delta, \Theta)$ representing the property to be shown. Each configuration of the game is a pair (w, q) with $w \in W$ and $q \in Q$. Initially a pawn is placed on the initial position of each automaton $(\hat{w}, \hat{q})$ (initial *configuration*). The satisfaction game is played in rounds with a set of rules detailing which player moves the pawns in which way on the field each round, resulting in a new configuration. Each player wins a play iff the other one is losing. Winning conditions for finite plays are detailed by the rules. Additionally, a play is lost if one player is forced to take a turn which is impossible to do (e.g. one pawn should be moved to an unreachable/non-existing position). Infinite plays are won by the Verifier iff the maximum of all infinitely often occurring acceptance numbers ($\Theta(q) \in \mathbb{N}$) for each configuration (w, q) of the play is even, else they are won by the Refuter.

Definition 3 (Winning Strategy). *A player is said to have a* winning strategy *iff he has a strategy to move the pawns on his turns that allows him to win the play regardless of moves the other player performs.*

The Verifier *wins a game* (opposed to a play) if he has a *winning strategy.*

Rules for satisfaction games with Kripke structures. The rules for satisfaction games between a Kripke structure $(W, \hat{w}, R, L)$ and an alternating tree automaton $(Q, \hat{q}, \delta, \Theta)$ are presented below:
If the current configuration is (w, q), choose the rule detailing winning conditions or movement of the pawns in the game according to the value of $\delta(q)$.

$\delta(q)$	Rules
p:	Verifier wins if $p \in L(w)$; Refuter wins if $p \notin L(w)$
$\neg p$:	Verifier wins if $\neg p \in L(w)$; Refuter wins if $\neg p \notin L(w)$
q':	the next configuration is (w, q')
$q_1 \tilde{\wedge} q_2$:	Refuter picks a q' from $\{q_1, q_2\}$; the next configuration is (w, q')
$q_1 \tilde{\vee} q_2$:	Verifier picks a q' from $\{q_1, q_2\}$; the next configuration is (w, q')
$\mathsf{EX}\, q'$:	Verifier picks $w' \in \{w\}.R$; the next configuration is (w', q')
$\mathsf{AX}\, q'$:	Refuter picks $w' \in \{w\}.R$; the next configuration is (w', q')

Satisfaction plays are sequences of configurations generated by these rules.

Definition 4 (Satisfaction for Kripke Structures). *A Kripke structure K* satisfies *an alternating tree automaton A, written $K \models A$ iff the Verifier has a winning strategy for the satisfaction game between K and A starting in the initial configuration $(\hat{w}, \hat{q})$.*

This definition of satisfaction games and these rules for Kripke structures correspond to the ones presented by Wilke [8] in Sec. 3 of his paper. Equivalence of the definition and the rules to the well known satisfaction of Kripke Structures with the modal μ-calculus follows from Theorem 3 in the same paper.

Example 3. Using the Kripke structure $\widehat{K}$ in Example 1 on page 3 and the alternating tree automaton $\widehat{A}$ in Example 2 on page 4, one has $\widehat{K} \models \widehat{A}$.

Any satisfaction game between $\widehat{K}$ and $\widehat{A}$ starts in the initial configuration $(\hat{S}, \hat{q})$. At each configuration (w, q) the value of $\delta(q)$ determines the rule employed. One possible play is depicted in the table below:

Round	(w,q)	$\delta(q)$	Rules, Choices and Movement
1	$(\hat{S}, \hat{q})$	$\mathsf{AX}\,\ddot{q}$	Refuter picks $w' \in \{\hat{S}\}.R = \{\hat{S}_0'''\}$ $\hookrightarrow$ Refuter picks $w' = \hat{S}_0'''$ $\hookrightarrow$ next configuration: $(\hat{S}_0''', \ddot{q})$
2	$(\hat{S}_0''', \ddot{q})$	$q_0 \tilde{\wedge} q_1$	Refuter picks a $q' \in \{q_0, q_1\}$ $\hookrightarrow$ Refuter picks $q' = q_1$ $\hookrightarrow$ next configuration: $(\hat{S}_0''', q_1)$
3	$(\hat{S}_0''', q_1)$	$\mathsf{EX}\,\ddot{q}$	Verifier picks $w' \in \{\hat{S}_0'''\}.R = \{\hat{S}', \hat{S}_1'''\}$ $\hookrightarrow$ Verifier picks $w' = \hat{S}_1'''$ $\hookrightarrow$ next configuration: $(\hat{S}_1''', \ddot{q})$
4	$(\hat{S}_1''', \ddot{q})$	$q_0 \tilde{\wedge} q_1$	Refuter picks a $q' \in \{q_0, q_1\}$ $\hookrightarrow$ Refuter picks $q' = q_1$ $\hookrightarrow$ next configuration: $(\hat{S}_1''', q_0)$
5	$(\hat{S}_1''', q_0)$	$\mathsf{EX}\,q_0'$	Verifier picks $w' \in \{\hat{S}_1'''\}.R = \{\hat{S}'', \hat{S}_2'''\}$ $\hookrightarrow$ Verifier picks $w' = \hat{S}''$ $\hookrightarrow$ next configuration: $(\hat{S}'', q_0')$
6	$(\hat{S}'', q_0')$	$\mathsf{AX}\,q_0''$	Refuter picks $w' \in \{\hat{S}''\}.R = \emptyset$ $\hookrightarrow$ Refuter can't pick a next world, thus looses the play $\hookrightarrow$ Verifier wins the play

Proof. A proof of $\widehat{K} \models \widehat{A}$ requires a winning strategy for the Verifier. As one can easily see, the only configurations the Verifier can make any decisions are configurations (w, q_0) and (w, q_1) with $w \in W$. The first choice is made at $w = \hat{S}_0'''$. Following the winning strategy presented below, one can easily see that $w \in \{\hat{S}_i''' | i \in \mathbb{N}\}$ always holds true. Thus decisions can only be made at $(\hat{S}_i''', q_0)$ or $(\hat{S}_i''', q_1)$.
A winning strategy is:

- $(\hat{S}_i''', q_0)$: Verifier chooses $\hat{S}'$ or $\hat{S}''$ as his next world,
- $(\hat{S}_i''', q_1)$: Verifier chooses $\hat{S}_{i+1}'''$ as his next world.

With this strategy, all finite games end like the example above. The Refuter can choose q_1 as often as he wants to, but the Verifier just returns to $\ddot{q}$, staying within $\{\hat{S}_i''' | i \in \mathbb{N}\}$. As soon as the Refuter chooses q_0 the play ends as above. If the refuter never chooses q_0 the play is infinite, thus the maximum over all infinitely often occurring acceptance numbers is calculated. Since in this example all acceptance numbers are 0, the maximum is 0, which is even. Thus the Verifier wins. □

3 Pre- and Post-Abstraction

This section is based upon the work of Dams [2], Fecher and Huth [4].

One problem with satisfaction games is the large number of worlds present within Kripke structures. Abstraction is used to reduce this number of worlds. An *abstraction function* α is a surjective function $\alpha : W \to I$ which maps concrete Kripke worlds ($w \in W$) to abstract ones ($\alpha(w) \in I$). Two worlds $w, w' \in W$ are said to be *compatible* iff $\alpha(w) = \alpha(w')$.

Example 4. Let $\hat{\alpha}$ be an abstraction function for the Kripke structure $\widehat{K}$ described in Example 1 on page 3 with $\hat{\alpha} : \hat{S} \mapsto \hat{i}$, $\hat{S}' \mapsto \hat{i}'$, $\hat{S}'' \mapsto \hat{i}''$, $\hat{S}'''_n \mapsto \hat{i}'''$ for $n \in \mathbb{N}$. Clearly all $\hat{S}'''_n, n \in \mathbb{N}$ are compatible worlds under $\hat{\alpha}$.

Let K^α be an abstracted Kripke structure. This means K^α is derived from a Kripke structure K by applying the abstraction function α to its world space and adjusting the rest of the structure in a manner that is irrelevant for the moment. Some possible structures of K^α will be presented in the next sections. The important things with abstraction are *soundness* and *precision*.

Definition 5 (Soundness of abstraction). *Let K be a Kripke structure, α an abstraction function and K^α an abstracted Kripke structure. The abstraction is said to be* sound *iff for all alternating tree automata A*

$$(K^\alpha \models A) \implies (K \models A)$$

holds true. Generally speaking the opposite implication does not hold.

Definition 6 (Precision of abstraction [informal]). *An abstraction is said to be* more precise than *another abstraction if it is possible to verify more properties in the first one than in the second one. An abstraction is said to be* precise *if (with the given parameters such as the abstraction function or an underlying satisfaction game) it has the most verifiable properties.*

Fecher and Huth [4] present two precise and sound abstraction techniques called *pre-* and *post-abstraction*.

3.1 Pre-Games and Pre-Abstraction

Pre-Games are special kinds of satisfaction games between a Kripke Structure K with an abstraction function α and an Alternating Tree Automaton A. While the game is played on the abstracted Kripke structure, all moves are based on the concrete Kripke structure. In other words, configurations of pre-games are pairs $(i, q) \in I \times Q$, but before applying any rule of the satisfaction game, the Refuter chooses one concrete world w with $\alpha(w) = i$. This world is used to determine the action within the selected rule, resulting in (w', q'). Then the new configuration is $(\alpha(w'), q')$. Hence the Verifier can only verify properties that hold for all compatible worlds of the current configuration.

This leads to *pre-abstraction*, which uses *generalized Kripke modal transition systems* incorporating *must-hypertransitions* to implement the abstraction given by α as well as the possible world changes inherent to pre-games.

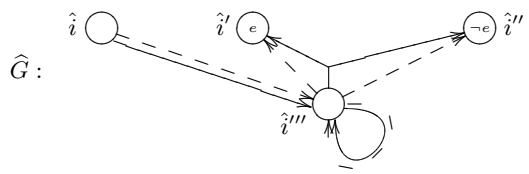

Fig. 3. A *generalized Kripke modal transition system.* The propositions $L(w) \subseteq AP$ at a world w are depicted within world borders. Dashed arrows $w \dashrightarrow w'$ denote may-transitions $(w, w') \in R^+$, while solid arrows $w \to \{w_1, \ldots, w_n\}, n \in \mathbb{N}$ represent must-hypertransitions $(w, \{w_1, \ldots, w_n\}) \in R^-$. World names are depicted close to the corresponding worlds.

Definition 7 (Generalized Kripke modal transition systems). *A* Generalized Kripke modal transition systems M *over* AP *is a tuple* $(W, \hat{w}, R^-, R^+, L)$ *where*

- W *is a (nonempty) set of worlds,*
- $\hat{w} \in W$ *is an initial world,*
- $R^- \subseteq W \times \mathbb{P}(W)$ *is its set of must-hypertransitions,*
- $R^+ \subseteq W \times W$ *is its set of may-transitions, and*
- $L \colon W \to \mathbb{P}(\mathrm{AP})$ *is its labeling function.*

M *is finite whenever* W *is.*

The complete means of constructing the generalized Kripke modal transition system via *pre-abstraction* are left to the paper of Fecher and Huth [4], Definition 7.

The definition of satisfaction and the rules for satisfaction games with generalized Kripke modal transition system are, with two exceptions, identical to the corresponding definitions and rules for Kripke structures. The changes are presented below:

$\delta(q)$	Rules
$\mathsf{EX}\, q'$:	Verifier picks $D' \in \{w\}.R^-$; Refuter picks $w' \in D$; next configuration: (w', q')
$\mathsf{AX}\, q'$:	Refuter picks $w' \in \{w\}.R^+$; next configuration: (w', q')

Precision related to pre-games and soundness of pre-abstraction are shown by Fecher and Huth [4].

Example 5. The generalized Kripke modal transition system $\widehat{G}$ from Fig. 3 is an abstraction from $\widehat{K}$ (Example 1 on page 3) using the abstraction function $\hat{\alpha}$ from Example 4 on the preceding page.

$\widehat{G} \models \widehat{A}$, with $\widehat{A}$ from Example 2 on page 4, can be easily seen with the following winning strategy:

- $(\hat{i}''', q_0)$: Verifier chooses the must-hypertransition $(\hat{i}''', \{\hat{i}', \hat{i}''\})$,
- $(\hat{i}''', q_1)$: Verifier chooses the self-loop $(\hat{i}''', \{\hat{i}'''\})$.

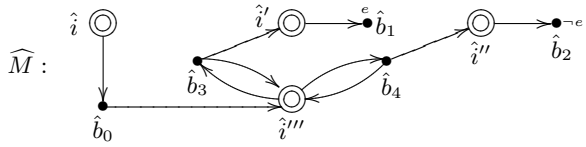

Fig. 4. A *μ-automaton.* OR-worlds ($\in O$) are depicted as double-lined circles and BRANCH-worlds ($\in B$) as solid circles. The propositions $L(b) \subseteq AP$ at a BRANCH-world b are depicted in small font next to the corresponding worlds. Arrows $o \to b$ and $b \to o$ denote OR-relations $(o, b) \in \rightrightarrows$ respectively BRANCH-relations $(b, o) \in \to$. World names are depicted close to the corresponding worlds.

3.2 Post-Games and Post-Abstraction

Post-Games are similar to pre-games, they also are special kinds of satisfaction games between a Kripke Structure K with an abstraction function α and an Alternating Tree Automaton A. Configurations off such games are, similar to satisfaction games on Kripke structures, pairs $(w, q) \in W \times Q$. After any move resulting in (w', q'), the Refuter may switch to another compatible world w'' of w', resulting in the new configuration (w'', q'). Consequently the Verifier is usually able to verify more properties than in pre-games, albeit at a (usually) higher cost.

This leads to *post-abstraction*, which uses *μ-automata* to implement the abstraction given by α as well as the possible world changes inherent to post-games.

Definition 8 (μ-automata). *A* μ-automaton M *over the set of atomic propositions* AP *is a tuple* $(O, B, \hat{w}, \rightrightarrows, \to, L)$ *such that*

- O *is a set of OR-worlds,*
- B *is a set of BRANCH-worlds (disjoint from* O*),*
- $\hat{w} \in O \cup B$ *is an initial world,*
- $\rightrightarrows \subseteq O \times B$ *is its OR-transition relation,*
- $\to \subseteq B \times O$ *is its BRANCH-transition relation, and*
- $L \colon B \to \mathbb{P}(\mathrm{AP})$ *is its labeling function.*

M *is finite if both* B *and* O *are.*

The complete means of constructing μ-automata via *post-abstraction* are left to the paper of Fecher and Huth [4], Definition 10.

The definition of satisfaction and the rules for satisfaction games with μ-automata are similar to the corresponding definitions and rules for Kripke structures. Configurations of satisfaction games between μ-automata and alternating tree automata are pairs $(w, q) \in (O \cup B) \times Q$.

If the current configuration is (o, q) with $o \in O$, the only rule is:

- Refuter picks $b \in \{o\}$. $\rightrightarrows$; the next configuration is (b, q).

Otherwise for configurations (b, q) with $b \in B$ the rules, with the following exceptions, are like the normal rules for Kripke structures:

$\delta(q)$	Rules
$\mathsf{EX}\, q'$:	Verifier picks $o \in \{b\}$. $\rightarrow$; the next configuration is (o, q')
$\mathsf{AX}\, q'$:	Refuter picks $o \in \{b\}$. $\rightarrow$; the next configuration is (o, q')

Precision related to post-games and soundness of pre-abstraction are shown by Fecher and Huth [4].

Example 6. The μ-automata $\widehat{M}$ from Fig. 4 on the preceding page is an abstraction from $\widehat{K}$ (Example 1 on page 3) using the abstraction function $\hat{\alpha}$ from Example 4 on page 7.

$\widehat{M} \models \widehat{A}$, with $\widehat{A}$ from Example 2 on page 4, can be easily seen with the following winning strategy:

- $(\hat{b}_3, q_0), (\hat{b}_4, q_0)$: Verifier chooses the transition to $\hat{i}'$ or $\hat{i}''$ respectively,
- $(\hat{b}_3, q_1), (\hat{b}_4, q_1)$: Verifier chooses the transition to $\hat{i}'''$.

3.3 Remarks

As Fecher and Huth [4] annotated, that there are cases where pre-abstraction leads to more complex models than post-abstraction and vice versa. Which one leads to the better results when verifying a given property depends heavily on that property (the alternating-tree-automata) as well as the underlying Kripke structure. The complexity of generalized Kripke modal transition systems (pre-abstraction) results from the must-hypertransitions. The complexity of μ-automata results from the blow up in brach worlds, which often is exponential.

Using the same abstraction function α post-abstraction is more precise than pre-abstraction, albeit most often more complex. Nevertheless using different abstraction functions the expressiveness of both abstractions is the same. If any property under an abstraction function α is not satisfied by the pre-abstracted automaton, but the post-abstracted one, one can construct an abstraction function α' such that the pre-abstraction under α' satisfies the property at the cost of even more increased complexity.

In the discussion of their paper, Fecher and Huth [4] sketch a combination of pre- and post-abstraction which may result in more precise abstraction without increasing the complexity too much. This leads to the generalized μ-automata detailed below.

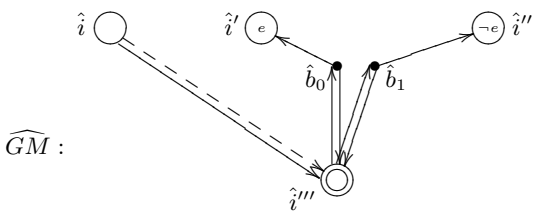

Fig. 5. A *generalized μ-automaton*. TRANS-worlds ($\in T$) are depicted as unfilled circles, OR-worlds ($\in O$) as double-lined circles and BRANCH-worlds ($\in B$) as small solid circles. The propositions $L(w) \subseteq \mathrm{AP}$ at a world w are depicted within world borders. Dashed arrows $t \dashrightarrow w$ are may-transitions $(t, w) \in R^+$, while solid arrows $t \to w$ between TRANS-states and from TRANS- to OR-states are must-transitions $(t, w) \in R^-$. Arrows $o \to b$ and $b \to w$ denote OR-relations $(o, b) \in \rightrightarrows$, respectively BRANCH-relations $(b, w) \in \to$. World names are depicted close to the corresponding worlds.

4 Generalized μ-Automata

The model of generalized μ-automata results from a combination of pre- and post-abstraction [4]. Its basic idea is employing the simpleness of pre-abstraction, as far as possible, and switching to post-abstraction if must-hypertransitions lead to greater complexity. In other words one employs the generalized Kripke modal transition systems, where all must-hypertransitions are replaced by additional worlds and transitions according to a local post-abstraction.

Definition 9 (Generalized μ-automaton). *A* generalized μ-automaton GM *over* AP *is a tuple* $(T, O, B, \hat{w}, R^-, R^+, \rightrightarrows, \to, L)$ *such that*

- T *is the set of TRANS-worlds,*
- O *is the set of OR-worlds,*
- B *is the set of BRANCH-worlds,*
- $\hat{w} \in (T \cup O \cup B)$ *is its initial world,*
- $R^-, R^+ \subseteq T \times (T \cup O)$ *is the set of* must- *and* may-*transitions (respectively),*
- $\rightrightarrows \subseteq O \times B$ *is the set of OR-transitions,*
- $\to \subseteq B \times (T \cup O)$ *is the set of BRANCH-transitions,*
- $L\colon (T \cup B) \to \mathbb{P}(\mathrm{AP})$ *is its labeling function.*

G *is finite if* T, O *and* B *are.*

The worlds $w \in T \cup O \cup B$ and the respective transitions (starting in these worlds) of the generalized μ-automaton consist of two parts. The "pre-abstraction-part" corresponds to the underlying pre-abstraction without must-hypertransitions, so one is in this part iff $w \in T$. The "post-abstraction-part" corresponds to the local post-abstraction that replaced any must-hypertransitions, so one is in this part iff either $w \in B$ or $w \in O$.

Remark 1 (Well-formedness of generalized μ-automata). A generalized μ-automata is said to be well-formed iff there is a may-transition $(v, w) \in R^+$ whenever there is a must-transition $(v, w) \in R^-$. Henceforth only well-formed generalized μ-automata are considered.

Example 7. An example of a generalized μ-automaton resulting from the Kripke structure in Example 1 on page 3 and the abstraction function $\hat{\alpha}$ from Example 4 on page 7 is illustrated in Fig. 5 on the preceding page.

As one can easily see, the worlds of the generalized μ-automaton $\widehat{GM}$ match the worlds of the generalized Kripke modal transition system $\widehat{G}$ almost perfectly. The only difference results from the world $\hat{i}'''$ where the must-hypertransition in $\widehat{G}$ starts. This world is replaced with $\hat{i}'''$ from $\widehat{M}$ along with the BRANCH-worlds $\hat{b}_0$ and $\hat{b}_1$ which correspond to the two BRANCH-worlds $\hat{b}_3$ and $\hat{b}_4$ from $\widehat{M}$. Clearly, all worlds $w \in \{\hat{i}, \hat{i}', \hat{i}''\}$ are inside the "pre-abstraction-part", while all worlds with $w \in \{\hat{i}''', \hat{b}_0, \hat{b}_1\}$ are inside the "post-abstraction-part".

For every world w from the "pre-abstraction-part" all transitions $\{w\}.R^+$ and $\{w\}.R^-$ from $\widehat{G}$ are taken over to $\widehat{GM}$. Similarly, for all worlds w from the "post-abstraction-part" all transitions $\{w\}.\rightrightarrows$ and $\{w\}.\rightarrow$ are taken over to their respective relations within $\widehat{GM}$.

Finally the labeling function of each underlying automaton is wrapped to the respective worlds.

Formally the generalized μ-automaton $\widehat{GM}$ is defined as:
$\widehat{GM} = (T, O, B, \hat{i}, R^-, R^+, \rightrightarrows, \rightarrow, L)$ with $\mathrm{AP} = \{true, false, e, \neg e\}$ and

- $T = \{\hat{i}, \hat{i}', \hat{i}''\}$,
- $O = \{\hat{i}'''\}$,
- $B = \{\hat{b}_0, \hat{b}_1\}$,
- $R^- = \{(\hat{i}, \hat{i}''')\}$,
- $R^+ = \{(\hat{i}, \hat{i}''')\}$,
- $\rightrightarrows = \{(\hat{i}''', \hat{b}_0), (\hat{i}''', \hat{b}_1)\}$,
- $\rightarrow = \{(\hat{b}_0, \hat{i}'''), (\hat{b}_1, \hat{i}'''), (\hat{b}_0, \hat{i}'), (\hat{b}_1, \hat{i}'')\}$, and
- $L\colon (T \cup B) \to \mathbb{P}(\mathrm{AP})$ with $\hat{i}' \mapsto \{e\}; \hat{i}'' \mapsto \{\neg e\}$ and $\hat{i}, \hat{b}_0, \hat{b}_1 \mapsto \emptyset$.

4.1 Satisfaction Games

Satisfaction games between a generalized μ-automaton $GM = (T, O, B, \hat{w}, R^-, R^+, \rightrightarrows, \rightarrow, L)$ and an alternating tree automaton $A = (Q, \hat{q}, \delta, \Theta)$ are similar to satisfaction games between Kripke structures and alternating tree automata. Each configuration of the game is a pair (w, q) with $w \in T \cup O \cup B$ and $q \in Q$. The initial configuration is $(\hat{w}, \hat{q})$.

The set of rules is divided in two parts. Depending on w of the current configuration (w, q), one is either in the "pre-abstraction-part" or in the "post-abstraction-part" (see above for details) of the generalized μ-automaton. Inside the "pre-abstraction-part" the rules of pre-abstraction are adapted and employed, inside the "post-abstraction-part" the rules of post-abstraction are in use.

Rules for Satisfaction Games. The rules for satisfaction games between a generalized μ-automaton $(T, O, B, \hat{w}, R^-, R^+, \rightrightarrows, \rightarrow, L)$ and an alternating tree automaton $(Q, \hat{q}, \delta, \Theta)$ are presented below:
If the current configuration is (w, q), choose the rule detailing winning conditions or movement of the pawns in the game depending on the part one is in and according to the value of $\delta(q)$.

pre-abstraction-part with $w \in T$:

$\delta(q)$	Rules
p:	Verifier wins if $p \in L(w)$; Refuter wins if $p \notin L(w)$
$\neg p$:	Verifier wins if $\neg p \in L(w)$; Refuter wins if $\neg p \notin L(w)$
q':	the next configuration is (w, q')
$q_1 \tilde{\wedge} q_2$:	Refuter picks a q' from $\{q_1, q_2\}$; the next configuration is (w, q')
$q_1 \tilde{\vee} q_2$:	Verifier picks a q' from $\{q_1, q_2\}$; the next configuration is (w, q')
$\mathsf{EX}\, q'$:	Verifier picks $w' \in \{w\}.R^-$; the next configuration is (w', q')
$\mathsf{AX}\, q'$:	Refuter picks $w' \in \{w\}.R^+$; the next configuration is (w', q')

post-abstraction-part with $w \in O$:

Rule
Refuter picks $b \in w.\rightrightarrows$; the next configuration is (b, q)

post-abstraction-part with $w \in B$:

$\delta(q)$	Rules
p:	Verifier wins if $p \in L(w)$; Refuter wins if $p \notin L(w)$
$\neg p$:	Verifier wins if $\neg p \in L(w)$; Refuter wins if $\neg p \notin L(w)$
q':	the next configuration is (w, q')
$q_1 \tilde{\wedge} q_2$:	Refuter picks a q' from $\{q_1, q_2\}$; the next configuration is (w, q')
$q_1 \tilde{\vee} q_2$:	Verifier picks a q' from $\{q_1, q_2\}$; the next configuration is (w, q')
$\mathsf{EX}\, q'$:	Verifier picks $w' \in \{w\}.\rightarrow$; the next configuration is (w', q')
$\mathsf{AX}\, q'$:	Refuter picks $w' \in \{w\}.\rightarrow$; the next configuration is (w', q')

Satisfaction plays are sequences of configurations generates by these rules.

Definition 10 (Satisfaction for Generalized μ-automata). *A generalized μ-automata GM* satisfies *an alternating tree automaton A, written $GM \models A$ iff the Verifier has a winning strategy for the satisfaction game between GM and A starting in the initial configuration $(\hat{w}, \hat{q})$.*

Remark 2. Soundness of these rules follows with the help of the notion of refinement presented in the next section, Remark 3 on page 15 and Theorem 3 on page 20. One can easily see that any generalized μ-automata GM satisfies an alternating tree automaton A iff there exists a Kripke structure K with $K \models A$.

Example 8. Using the generalized μ-automaton $\widehat{GM}$ in Example 7 on page 12 and the alternating tree automata $\widehat{A}$ in Example 2 on page 4, one has $\widehat{GM} \models \widehat{A}$.

Any satisfaction game between $\widehat{GM}$ and $\widehat{A}$ starts in the initial configuration $(\hat{i}, \hat{q})$. One possible play is depicted in the table below:

Round	(w, q)	part	$\delta(q)$	Rules, Choices and Movement
1	$(\hat{i}, \hat{q})$	pre	$\mathsf{AX}\,\ddot{q}$	Refuter picks $w' \in \{\hat{i}\}.R^+ = \{\hat{i}'''\}$ ↪ Refuter picks $w' = \hat{i}'''$ ↪ next configuration: $(\hat{i}''', \ddot{q})$
2	$(\hat{i}''', \ddot{q})$	post	–	Refuter picks $b \in \{\hat{i}'''\}.\rightrightarrows = \{\hat{b}_0, \hat{b}_1\}$ ↪ Refuter picks $b = \hat{b}_0$ ↪ next configuration: $(\hat{b}_0, \ddot{q})$
3	$(\hat{b}_0, \ddot{q})$	post	$q_0 \tilde{\wedge} q_1$	Refuter picks a $q' \in \{q_0, q_1\}$ ↪ Refuter picks $q' = q_1$ ↪ next configuration: $(\hat{b}_0, q_1)$
4	$(\hat{b}_0, q_1)$	post	$\mathsf{EX}\,\ddot{q}$	Verifier picks $w' \in \{\hat{b}_0\}.\rightarrow = \{\hat{i}', \hat{i}'''\}$ ↪ Verifier picks $w' = \hat{i}'''$ ↪ next configuration: $(\hat{i}''', \ddot{q})$
5	$(\hat{i}''', \ddot{q})$	post	–	Refuter picks $b \in \{\hat{i}'''\}.\rightrightarrows = \{\hat{b}_0, \hat{b}_1\}$ ↪ Refuter picks $b = \hat{b}_1$ ↪ next configuration: $(\hat{b}_1, \ddot{q})$
6	$(\hat{b}_1, \ddot{q})$	post	$q_0 \tilde{\wedge} q_1$	Refuter picks a $q' \in \{q_0, q_1\}$ ↪ Refuter picks $q' = q_0$ ↪ next configuration: $(\hat{b}_1, q_0)$
7	$(\hat{b}_1, q_0)$	post	$\mathsf{EX}\,q_0'$	Verifier picks $w' \in \{\hat{b}_1\}.\rightarrow = \{\hat{i}'', \hat{i}'''\}$ ↪ Verifier picks $w' = \hat{i}''$ ↪ next configuration: $(\hat{i}'', q_0')$
8	$(\hat{i}'', q_0')$	pre	$\mathsf{AX}\,q_0''$	Refuter picks $w' \in \{\hat{i}''\}.R^+ = \emptyset$ ↪ Refuter can't pick a next world, thus looses ↪ Verifier wins the play

Proof. A proof of $\widehat{GM} \models \widehat{A}$ requires a winning strategy for the Verifier. Similar to the proof of Example 3 on page 6 one can easily see that the only time the Verifier can make any decisions is at the configurations $(\hat{b}_0, q_0), (\hat{b}_1, q_0), (\hat{b}_0, q_1), (\hat{b}_1, q_1)$. A winning strategy is:

- $(\hat{b}_0, q_0), (\hat{b}_1, q_0)$: Verifier chooses the transition to $\hat{i}'$ or $\hat{i}''$ respectively,
- $(\hat{b}_0, q_1), (\hat{b}_1, q_1)$: Verifier chooses the transition to $\hat{i}'''$.

With this strategy all finite games end like the example above. The choice of $\hat{b}_0$ or $\hat{b}_1$ doesn't influence the outcome of the game. The Refuter can choose q_1 as often as he wants to, but the Verifier just returns to $\ddot{q}$, staying within $\{\hat{i}''', \hat{b}_0, \hat{b}_1\}$. As soon as the Refuter chooses q_0 the play ends as above. If the refuter never chooses q_0 the play is infinite, thus the maximum over all infinitely often occurring acceptance numbers is calculated. Since in this example all acceptance numbers are 0, the maximum is 0, which is even. Thus the Verifier wins. □

5 Refinement

Generally, refinement is the opposite technique of abstraction. If model M_1 refines Model M_2 then M_2 abstracts M_1. As mentioned before, abstraction (and so its converse refinement) is useful in the context of verification. Here one employs abstraction to yield a smaller number of worlds within a system and then uses model checking on this abstract system. The notion of refinement games presented below allows one to correlate two given generalized μ-automata and construct an abstraction (refinement) hierarchy of such automata.

Remark 3 (Kripke structures as generalized μ-automata). It can be easily seen, that Kripke structures can be seen as a subset of generalized μ-automata. Given a Kripke structure $K = (W, \hat{w}, R, L)$, you can easily construct a generalized μ-automata $GM[K] = (W, \emptyset, \emptyset, \hat{w}, R, R, \emptyset, \emptyset, L)$ which corresponds to this Kripke structure. It is obvious that for any alternating tree automata A: $K \models A$ iff $GM[K] \models A$.

5.1 Refinement Games of Generalized μ-Automata

Refinement of generalized μ-automata is defined with the help of *refinement games*, which are special kinds of parity games, similar to satisfaction games. Two players (*Player I* and *Player II*) play this game on a field consisting of two generalized μ-automata $GM_1 = (T_1, O_1, B_1, \hat{w}_1, R_1^-, R_1^+, \rightrightarrows_1, \rightarrow_1, L_1)$ and $GM_2 = (T_2, O_2, B_2, \hat{w}_2, R_2^-, R_2^+, \rightrightarrows_2, \rightarrow_2, L_2)$. To simplify the notations, define $W_1 = T_1 \cup O_1 \cup B_1$ and $W_2 = T_2 \cup O_2 \cup B_2$.

Each configuration of the game is a pair $(w_1, w_2) \in W_1 \times W_2$. Initially a pawn is placed on the initial worlds of each automaton, resulting in the initial configuration $(\hat{w}_1, \hat{w}_2)$. Refinement games are played in rounds with a set of rules detailing which player moves the pawns in which way on the field each round, resulting in a new configuration. If more than one rule is applicable a single one is chosen by Player II. Each player wins a play iff the other one is losing. Winning conditions for finite plays are detailed by the rules. Additionally, a play is lost if one player is forced to take a turn which is impossible to do (e.g. one pawn should be moved to an unreachable/non-existing position). Infinite plays are won by Player I.

Definition 11 (Winning Strategy). *A player is said to have a* winning strategy *iff he has a strategy to move the pawns on his turns that allows him to win all plays regardless of the moves the other player performs and the rules chosen.*

Definition 12 (Refinement). *Let $GM_1 = (T_1, O_1, B_1, \hat{w}_1, R_1^-, R_1^+, \rightrightarrows_1, \rightarrow_1, L_1)$ and $GM_2 = (T_2, O_2, B_2, \hat{w}_2, R_2^-, R_2^+, \rightrightarrows_2, \rightarrow_2, L_2)$ be two generalized μ-automata. GM_1* refines *GM_2 (and GM_2* abstracts *GM_2) iff Player I has a winning strategy for all refinement plays (using the rules stated below) between GM_1 and GM_2 started at $(\hat{w}_1, \hat{w}_2)$.*

Rules for Refinement Games. The rules for refinement games between two generalized μ-automata $GM_1 = (T_1, O_1, B_1, \hat{w}_1, R_1^-, R_1^+, \rightrightarrows_1, \rightarrow_1, L_1)$ and $GM_2 = (T_2, O_2, B_2, \hat{w}_2, R_2^-, R_2^+, \rightrightarrows_2, \rightarrow_2, L_2)$ with $W_1 = T_1 \cup O_1 \cup B_1$ and $W_2 = T_2 \cup O_2 \cup B_2$ at configuration (w_1, w_2) are presented below, divided by a case analysis on the considered configuration:

$w_1 \in O_1$**:**
Player II picks $w_1' \in \{w_1\}.\rightrightarrows$; next configuration: (w_1', w_2)

$w_1 \notin O_1 \wedge w_2 \in O_2$**:**
Player I picks $w_2' \in \{w_2\}.\rightrightarrows$; next configuration: (w_1, w_2')

$w_1 \notin O_1 \wedge w_2 \notin O_2$**:**
Player II can choose any of the following three alternative rules:

1. Player II picks $p \in L_2(w_2)$; Player I wins iff $p \in L_1(w_1)$
2. Player II picks $w_2' \in (\{w_2\}.R_2^-) \cup (\{w_2\}.\rightarrow_2)$;
   Player I picks $w_1' \in (\{w_1\}.R_1^-) \cup (\{w_1\}.\rightarrow_1)$;
   $\hookrightarrow$ the next configuration is (w_1', w_2')
3. Player II picks $w_1' \in (\{w_1\}.R_1^+) \cup (\{w_1\}.\rightarrow_1)$;
   Player I picks $w_2' \in (\{w_2\}.R_2^+) \cup (\{w_2\}.\rightarrow_2)$;
   $\hookrightarrow$ the next configuration is (w_1', w_2')

Refinement games are sequences of configurations generates by these rules.

Example 9. Using the generalized μ-automaton $\widehat{GM}$ from Example 7 on page 12 and the generalized μ-automaton $GM[\widehat{K}]$ from Remark 3 on the previous page corresponding to the Kripke Structure $\widehat{K}$ in Example 1 on page 3, one has $GM[\widehat{K}]$ refines $\widehat{GM}$.

Any refinement game between $GM[\widehat{K}]$ and $\widehat{GM}$ starts in the initial configuration $(\hat{S}, \hat{i})$. One possible play is depicted in the table below:

Round	(w_1, w_2)	Rules	Choices and Movement
1	$(\hat{S}, \hat{i})$	$\hat{S} \notin O_1$ $\hat{i} \notin O_2$ second rule	Player II picks $w_2' \in \{\hat{i}\}.R^- = \{\hat{i}'''\}$ $\hookrightarrow$ Player I picks $w_1' \in \{\hat{S}\}.R^- = \{\hat{S}_0'''\}$ $\hookrightarrow$ next configuration: $(\hat{S}_0''', \hat{i}''')$
2	$(\hat{S}_0''', \hat{i}''')$	$\hat{S}_0''' \notin O_1$ $\hat{i}''' \in O_2$	Player I picks $w_2' \in \{\hat{i}'''\}.\rightrightarrows = \{\hat{b}_0, \hat{b}_1\}$ Player I picks $w_2' = \hat{b}_0$ $\hookrightarrow$ next configuration: $(\hat{S}_0''', \hat{b}_0)$
3	$(\hat{S}_0''', \hat{b}_0)$	$\hat{S}_0''' \notin O_1$ $\hat{b}_0 \notin O_2$ third rule	Player II picks $w_1' \in \{\hat{S}_0'''\}.R^+ = \{\hat{S}', \hat{S}_1'''\}$ $\hookrightarrow$ Player II picks $w_1' = \hat{S}_1'''$ $\hookrightarrow$ Player I picks $w_2' \in \{\hat{b}_0\}.\rightarrow = \{\hat{i}', \hat{i}'''\}$ $\hookrightarrow$ Player I picks $w_2' = \hat{i}'''$ $\hookrightarrow$ next configuration: $(\hat{S}_1''', \hat{i}''')$
4	$(\hat{S}_1''', \hat{i}''')$	$\hat{S}_1''' \notin O_1$ $\hat{i}''' \in O_2$	Player I picks $w_2' \in \{\hat{i}'''\}.\rightrightarrows = \{\hat{b}_0, \hat{b}_1\}$ Player I picks $w_2' = \hat{b}_1$ $\hookrightarrow$ next configuration: $(\hat{S}_1''', \hat{b}_1)$

Round	(w_1, w_2)	Rules	Choices and Movement
5	$(\hat{S}_1''', \hat{b}_1)$	$\hat{S}_1''' \notin O_1$ $\hat{b}_1 \notin O_2$ second rule	Player II picks $w_2' \in \{\hat{b}_1\}.\rightarrow = \{\hat{i}'', \hat{i}'''\}$ $\hookrightarrow$ Player II picks $w_2' = \hat{i}''$ $\hookrightarrow$ Player I picks $w_1' \in \{\hat{w}_1\}.R^- = \{\hat{S}'', \hat{S}_2'''\}$ $\hookrightarrow$ Player I picks $w_1' = \hat{S}''$ $\hookrightarrow$ next configuration: $(\hat{S}'', \hat{i}'')$
6	$(\hat{S}'', \hat{i}'')$	$\hat{S}'' \notin O_1$ $\hat{i}'' \notin O_2$ first rule	Player II picks $p \in L_2(\hat{i}'') = \{\neg e\}$ $\hookrightarrow$ Player II picks $p = \neg e$ $\hookrightarrow$ since $p = \neg e \in L_1(\hat{S}'')$ $\hookrightarrow$ Player I wins the play

Proof. A proof of $GM[\widehat{K}]$ refines $\widehat{GM}$ requires a winning strategy for Player I. One can easily see that the only time Player I can make any decisions in with $w_2 \in O_2$ and the second or third alternate rule.

The winning strategy presented below makes sure that $w_2 \in O_2$ is only ever reached within configurations $(\hat{S}_i''', \hat{i}''')$ with $i \in \mathbb{N}$. Additionally the second and third alternate rules are only reached within configurations $(\hat{S}_i''', \hat{b}_j)$ with $i \in \mathbb{N}$ and $j \in \{1, 2\}$. Thus the following winning strategy is complete.
A possible *winning strategy* is:

With $w_2 \in O_2$, Player I picks $w_2' \in \{\hat{b}_0, \hat{b}_1\}$ depending on w_1. This choice can only be made at $w_1 = \hat{S}_i'''$ with $i \in \mathbb{N}$. If i is even Player I picks $w_2' = \hat{b}_0$ else Player I picks $w_2' = \hat{b}_1$.

With the second alternate rule, Player I picks according to the choice done by Player II. If Player II picks $\hat{i}'(\hat{i}'')$, Player I picks $\hat{S}'(\hat{S}'')$. If Player II picks $\hat{i}'''$, Player I picks $\hat{S}_i'''$ with $i \in \mathbb{N}$.

The third alternate rule is analog to the second one. That is Player I picks according to the choice done by Player II. If Player II picks $\hat{S}'(\hat{S}'')$, Player I picks $\hat{i}'(\hat{i}'')$. If Player II picks $\hat{S}_i'''$ with $i \in \mathbb{N}$, Player I picks $\hat{i}'''$.

With this strategy all finite plays end similarly to the example play presented above. Further according to the definition all infinite plays are won by Player I. Thus Player I wins all refinement plays between $GM[\widehat{K}]$ and $\widehat{GM}$. □

5.2 Reflexive, Transitive and Sound Refinement

The notion of refinement presented above is reflexive, transitive and sound as shown by the theorems below.

Theorem 1 (Reflexiveness). *Let GM be a generalized μ-automaton. Then GM refines GM.*

Proof. An obvious winning strategy for Player I is always selecting the same worlds as Player II. This selection is possible as long as both pawns are on the same worlds, i.e. the configuration is (w, w).

If $w \in O$ Player II must choose $w' \in \{w\}.\rightrightarrows$ resulting in configuration (w', w) with $w' \in B$. Thus Player I can pick the same $w' \in \{w\}.\rightrightarrows$ resulting in configuration (w', w').

If $w \notin O$ Player I obviously has the same possibilities to select next worlds as Player II resulting in configuration (w', w').

Thus as long as Player II is moving its pawn, Player I is able to move its pawn in the same way. If Player II selects the first of the alternate rules, thus picking $p \in L(w)$, Player I is obviously able to pick $p \in L(w)$, too.

Thus Player I wins all games and GM refines (abstracts) GM. □

Definition 13 (Reachability). *In any refinement game with GM refines GM_2 employing a winning strategy ν, a configuration (w_1, w_2) is called* reachable under ν *iff there exist a sequence of configurations in any play of this game, that is won by Player I with the help of ν, such that (w_1, w_2) is a configuration of this sequence.*

In any satisfaction game with $GM_1 \models A$ employing a winning strategy γ, a configuration (w, q) is called reachable under γ *iff there exist a sequence of configurations in any play of this game, that is won by the Verifier with the help of γ, such that (w, q) is a configuration of this sequence.*

Theorem 2 (Transitiveness). *Let GM_1, GM_2 and GM_3 be three generalized μ-automata with GM_1 refines GM_2 and GM_2 refines GM_3. Then GM_1 refines GM_3.*

Proof. Let ν_{12} and ν_{23} be history independent winning strategies for refinement games between GM_1, GM_2 and GM_2, GM_3 respectively.

Let $\mathcal{V}$ be the set of configurations $((w_1, w_3), w_2)$, such that (w_1, w_2) is reachable under ν_{12} and (w_2, w_3) is reachable under ν_{23}. In those configurations, w_2 encodes partial history information and is always modified by Player I.

Initially w_2 is set to $\hat{w}_2$, and obviously$((\hat{w}_1, \hat{w}_3), \hat{w}_2) \in \mathcal{V}$. If $\hat{w}_1 \notin O_1 \wedge \hat{w}_2 \in O_2$ Player 1 has to set w_2 to w_2' according to ν_{12} at the configuration $(\hat{w}_1, \hat{w}_2)$. The new starting configuration is $((\hat{w}_1, \hat{w}_3), w_2')$. Obviously $(\hat{w}_1, w_2')$ is reachable under ν_{12}. Additionally since $(\hat{w}_2, \hat{w}_3)$ is reachable under ν_{23}, there exists a play between GM_2 and GM_3 such that at configuration $(\hat{w}_2, \hat{w}_3)$ Player II picks w_2'. Thus $(w_2', \hat{w}_3)$ is reachable under ν_{23} and the starting configuration is in $\mathcal{V}$.

A winning strategy ν_{13} for Player I on any $((w_1, w_3), w_2) \in \mathcal{V}$ is defined by a case analysis on this configuration as follows, where also the modification of w_2 is being defined. Additionally it is shown that any play, beginning in a configuration of $\mathcal{V}$ and played by Player I according to the strategy presented below, stays within $\mathcal{V}$.

Case Analysis of $((w_1, w_3), w_2) \in \mathcal{V}$:

$w_1 \in O_1$:

Player II picked $w_1' \in \{w_1\}.\rightrightarrows$. If $w_2 \notin O_2$ the next configuration is $((w_1', w_3), w_2)$. Since (w_1, w_2) is reachable under ν_{12}, there exists a play between GM_1 and GM_2 under ν_{12} such that at configuration (w_1, w_2) Player II picks w_1'. Thus (w_1', w_2) is reachable under ν_{12}, thus the next configuration is in $\mathcal{V}$.

If $w_2 \in O_2$ Player 1 sets w_2' according to ν_{12} at the configuration (w_1', w_2). The next configuration is $((w_1', w_3), w_2')$. Obviously (w_1', w_2') is reachable under ν_{12}. Since (w_2, w_3) is reachable under ν_{23}, there exists a play between

GM_2 and GM_3 under ν_{23} such that at configuration (w_2, w_3) Player II picks w_2'. Thus (w_2', w_3) is reachable under ν_{23}, thus the next configuration is in $\mathcal{V}$.

$w_1 \notin O_1 \wedge w_3 \in O_3$**:** (and $w_2 \notin O_2$)
Player I picks w_3' according to ν_{23} at the configuration (w_2, w_3). The next configuration is $((w_1, w_3'), w_2)$. Obviously (w_2, w_3') is reachable under ν_{23}, thus the next configuration is in $\mathcal{V}$.

$w_1 \notin O_1 \wedge w_3 \notin O_3$**:** (and $w_2 \notin O_2$)

- Player II selected the second alternative rule and chose w_3'.
  Player I sets $w_2' \in (\{w_2\}.R_2^-) \cup (\{w_2\}.\rightarrow_2)$ according to ν_{23} at configuration (w_2, w_3) as well as the choice of w_3'. Then Player I picks $w_1' \in (\{w_1\}.R_1^-) \cup (\{w_1\}.\rightarrow_1)$ according to ν_{12} at configuration (w_1, w_2) as well as the choice of w_2'. There are three special cases to consider now:
  - $w_1' \notin O_1 \wedge w_2' \in O_2$**:** It is obvious that (w_1', w_2') is reachable under ν_{12}, thus Player 1 is able to set w_2'' according to ν_{12} at the configuration (w_1', w_2'). The next configuration is $((w_1', w_3'), w_2'')$. Obviously (w_1', w_2'') is reachable under ν_{12}. Additionally it is obvious that (w_2', w_3') is reachable under ν_{23}, thus there exists a play between GM_2 and GM_3 such that at configuration (w_2', w_3') Player II picks w_2''. Thus (w_2'', w_3') is reachable under ν_{23} and thus the the next configuration is in $\mathcal{V}$.
  - $w_1' \notin O_1 \wedge w_2' \notin O_2$**:** The next configuration is $((w_1', w_3'), w_2')$. Obviously (w_1', w_2') is reachable under ν_{12} and (w_2', w_3') is reachable under ν_{23}, thus the next configuration is in $\mathcal{V}$.
  - $w_1' \in O_1 \wedge w_2' \in O_2$**:** Since a selection of $w_2'' \notin O_2$ depends on $w_1' \notin O_1$, this has to wait until the next step of the play. Thus the next configuration is the same as in the second case.
- Player II selected the third alternative rule and chose w_1'.
  Player I sets $w_2' \in (\{w_2\}.R_2^+) \cup (\{w_2\}.\rightarrow_2)$ according to ν_{12} at configuration (w_1, w_2) as well as the choice of w_1'. Then Player I picks $w_3' \in (\{w_3\}.R_3^+) \cup (\{w_3\}.\rightarrow_3)$ according to ν_{23} at configuration (w_2, w_3) as well as the choice of w_2'. There are exactly the same three special cases to consider as above.

It should be obvious that the strategy ensures the invariant $w_1 \notin O_1 \implies w_2 \notin O_2$ in front of every step in the game.

Employing the strategy presented above ensures that Player I can perform a move of its pawn whenever he has to do so. Additionally since all moves are performed according to the two winning strategies ν_{12} and ν_{23}, whenever Player II selects the first alternate rule, picking $p \in L_3(w_3)$ there exists $p \in L_2(w_2)$ (due to ν_{23}) and thus $p \in L_1(w_1)$ (due to ν_{12}).

Thus the strategy presented above is a winning strategy and Player I wins all refinement games between GM_1 and GM_3. Thus GM_1 refines GM_3. □

Theorem 3 (Soundness). *Let GM_1 and GM_2 be two generalized μ-automata with GM_1 refines GM_2. This notion of refinement is sound, i.e. for all alternating tree automata A*

$$(GM_2 \models A) \implies (GM_1 \models A)$$

Proof. Let A be an alternating tree automaton. Let γ_2 be a history independent winning strategy for the Verifier with respect to the satisfaction game $GM_2 \models A$. Let ν be a history independent winning strategy for Player I with respect to the refinement game between GM_1 and GM_2.

Let $\mathcal{V}$ be the set of configurations $((w_1, q), w_2)$, such that (w_2, q) is reachable under γ_2 and (w_1, w_2) is reachable under ν. In those configurations, w_2 encodes partial history information and is always modified by the Verifier.

Initially w_2 is set to $\hat{w}_2$, and obviously $((\hat{w}_1, \hat{q}), \hat{w}_2) \in \mathcal{V}$. If $\hat{w}_1 \notin O_1 \wedge \hat{w}_2 \in O_2$ the Verifier has to set w_2 to w_2' according to ν at the configuration $(\hat{w}_1, \hat{w}_2)$ as if the Verifier would be Player I. The new starting configuration is $((\hat{w}_1, \hat{q}), w_2')$. Obviously $(\hat{w}_1, w_2')$ is reachable under ν. Additionally since $(\hat{w}_2, \hat{q})$ is reachable under γ_2, there exists a play between GM_2 and A such that at configuration $(\hat{w}_2, \hat{q})$ the Refuter picks w_2'. Thus $(w_2', \hat{q})$ is reachable under γ_2 and the starting configuration is in $\mathcal{V}$.

A winning strategy γ_1 for the Verifier on any configuration $((w_1, q), w_2) \in \mathcal{V}$ is defined by a case analysis on on this configuration, where also the modification of w_2 is being defined. Additionally it is shown that any play, beginning in a configuration of $\mathcal{V}$ and played by the Verifier according to the strategy presented below, stays within $\mathcal{V}$.

In the definition of the rules of the satisfaction game on page 13, there are three distinct parts. Here two parts are enough. If $\delta(q) = \mathsf{EX}\, q'$ the Verifier is (depending on w) able to pick from R^- or $\rightarrow$. The second alternative rule in the refinement game allows the same choices. Thus the same set of rules is able to govern the strategy for $w_1 \in B$ as well as $w_1 \in T$. Similarly for $\delta(q) = \mathsf{AX}\, q'$ with the third alterative rule.

Case Analysis of $((w_1, q), w_2) \in \mathcal{V}$**:**

$w_1 \in O_1$**:**

Refuter chose w_1'. If $w_2 \notin O_2$ the next configuration is $((w_1', q), w_2)$. Since (w_1, w_2) is reachable under ν, there exists a play between GM_1 and GM_2 such that at configuration (w_1, w_2) Player II picks w_1'. Thus (w_1', w_2) is reachable under ν and the next configuration is in $\mathcal{V}$.

If $w_2 \in O_2$ the Verifier sets w_2' according to ν at the configuration (w_1', w_2) as if he would be Player I. The next configuration is $((w_1', q), w_2')$. Obviously (w_1', w_2') is reachable under ν. Since (w_2, q) is reachable under γ_2, there exists a play between GM_2 and A such that at configuration (w_2, q) the Refuter picks w_2'. Thus (w_2', q) is reachable under γ_2 and the next configuration is in $\mathcal{V}$.

$w_1 \in (T_1 \cup B_1)$**:** (and $w_2 \notin O_2$)

$\delta(q) = q_1 \tilde{\wedge} q_2$**:**

Refuter picked $q' \in \{q_1, q_2\}$. The next configuration is $((w_1, q'), w_2)$.

Since (w_2, q) is reachable under γ_2, there exists a play between GM_2 and A such that at configuration (w_2, q) the Refuter picks q'. Thus (w_2, q') is reachable under γ_2 and the next configuration is in $\mathcal{V}$.

$\delta(q) = q_1 \tilde{\vee} q_2$**:**
Verifier picks $q' \in \{q_1, q_2\}$ according to γ_2 at the configuration (w_2, q). The next configuration is $((w_1, q'), w_2)$. Obviously (w_2, q') is reachable under γ_2. Thus the next configuration is in $\mathcal{V}$.

$\delta(q) = \mathsf{EX}\, q'$**:**
Verifier sets w_2' according to γ_2 at the configuration (w_2, q). Then the Verifier picks w_1' according to ν at the configuration (w_1, w_2) where he acts as Player I and as if Player II just chose w_2'. There are three special cases to consider now:

- $w_1' \notin O_1 \wedge w_2' \in O_2$**:** It is obvious that (w_1', w_2') is reachable under ν, thus the Verifier is able to set w_2'' according to ν at the configuration (w_1', w_2') as if he were Player I. The next configuration is $((w_1', q'), w_2'')$. Since obviously (w_2', q') is reachable under γ_2, there exists a play between GM_2 and A such that at configuration (w_2', q') the Refuter picks w_2''. Thus (w_2'', q') is reachable under γ_2. Additionally it is obvious that (w_1', w_2'') is reachable under ν and thus the the next configuration is in $\mathcal{V}$.
- $w_1' \notin O_1 \wedge w_2' \notin O_2$**:** The next configuration is $((w_1', q'), w_2')$. Obviously is (w_2', q') reachable under γ_2 and (w_1', w_2') reachable under ν. Thus the next configuration is in $\mathcal{V}$.
- $w_1' \in O_1 \wedge w_2' \in O_2$**:** Since a selection of $w_2'' \notin O_2$ depends on $w_1' \notin O_1$, this has to wait until the next step of the play. Thus the next configuration is the same as in the second case.

$\delta(q) = \mathsf{AX}\, q'$**:**
Refuter chose w_1'. Verifier sets w_2' according to ν at the configuration (w_1, w_2) where he acts as Player I and as if Player II just chose w_1'. There are three special cases to consider now:

- $w_1' \notin O_1 \wedge w_2' \in O_2$**:** It is obvious that (w_1', w_2') is reachable under ν, thus the Verifier is able to set w_2'' according to ν at the configuration (w_1', w_2') as if he were Player I. The next configuration is $((w_1', q'), w_2'')$. Obviously (w_1', w_2'') is reachable under ν. Additionally since (w_2, q) is reachable under γ_2, there exists a play between GM_2 and A such that at configuration (w_2, q) the Refuter picks w_2'. Thus (w_2', q') is reachable under γ_2. Since $w_2' \in O_2$ there exists a play between GM_2 and A such that at configuration (w_2', q) the Refuter picks w_2''. Thus (w_2'', q') is reachable under γ_2 and the next configuration is in $\mathcal{V}$.
- $w_1' \notin O_1 \wedge w_2' \notin O_2$**:** The next configuration is $((w_1', q'), w_2')$. It is obvious that (w_1', w_2') is reachable under ν. Additionally since (w_2, q) is reachable under γ_2, there exists a play between GM_2 and A such that at configuration (w_2, q) the Refuter picks w_2'. Thus (w_2', q') is reachable under γ_2. Thus the next configuration is in $\mathcal{V}$.

$w_1' \in O_1 \wedge w_2' \in O_2$**:** Since a selection of $w_2'' \notin O_2$ depends on $w_1' \notin O_1$, this has to wait until the next step of the play. Thus the next configuration is the same as in the second case.

It should be obvious that the strategy ensures the invariant $w_1 \in (T_1 \cup B_1) \implies w_2 \notin O_2$ in front of every step in the game.

Employing the strategy presented above ensures that the Verifier can perform a move of its pawn whenever he has to. Additionally whenever $\delta(q) = p$ or $\delta(q) = \neg p$, Verifier wins because of γ_2 and ν being winning strategies, i.e. γ_2 ensures that $p \in L_2(w_2)$ and the first alternate rule of the refinement game with ν ensures that $p \in L_1(w_1)$. Thus the Verifier wins all finite satisfaction plays. Infinite plays are also won, since γ_2 is a winning strategy, i.e. the maximum of all infinitely occurring acceptance numbers for all configurations (w_2, q) of the play is even, thus the maximum of all infinitely occurring acceptance numbers for all configurations (w_1, q) is even, too.

Thus the strategy presented above is a winning strategy and the Verifier wins all satisfaction games between GM_1 and A. Thus $GM_1 \models A$. □

Bibliography

[1] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981. ISBN 3-540-11212-X.

[2] Dennis Dams. *Abstract interpretation and partition refinement for model checking.* PhD thesis, Technische Universiteit Eindhoven, The Netherlands, 1996.

[3] Luca de Alfaro, Patrice Godefroid, and Radha Jagadeesan. Three-valued abstractions of games: Uncertainty, but with precision. In *LICS*, pages 170–179, 2004.

[4] Harald Fecher and Michael Huth. More precise partition abstraction. In *VMCAI*, Lecture Notes in Computer Science. Springer, 2007.

[5] Susanne Graf and Hassen Saïdi. Construction of abstract state graphs with pvs. In *CAV*, pages 72–83, 1997.

[6] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[7] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in cesar. In *Symposium on Programming*, pages 337–351, 1982.

[8] Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Belg. Math. Soc.*, 8(2):359–391, May 2001.