Model-checking Games (handout)

Seminar on Games in Verification and Synthesis
(University of Saarland, Reactive Systems Group, Klaus Dräger)
By Walid Haddad

1 Overview

[1] presents two approaches which reduce the *model checking problem* and the *satisfiability problem* of μ-calculus to the *winner problem* in *parity games*. We focus on the model checking approach; the model checking problem is reduced to the *acceptance problem* for *alternating tree automata* (ATAs) which is then reduced to the winner problem for parity games.

Some facts are decisive for this approach:

- Alternating tree automata are equivalent to the *modal μ-calculus* with respect to expressive power
- Parity games are powerful tools that can solve the acceptance problem for ATAs with an acceptable complexity

Modal μ-calculus model checking requires that models of systems be represented as Kripke structures:

1.1 Kripke structures

Formally, a Kripke structure is a tuple $\mathcal{K}$ = (W, A, κ) where:

- W is a set of *worlds*
- A $\subseteq$ W $\times$ W is an *accessibility relation*
- κ: $\mathcal{Q} \rightarrow 2^W$ is an *interpretation* of the propositional variables, which assigns to each propositional variable the set of worlds where it holds true

A *pointed Kripke structure* is a pair ($\mathcal{K}$, ω) where $\mathcal{K}$ is a Kripke structure and ω a world of it. A *Kripke query* is a class of pointed Kripke structures.

1.2 Modal μ-calculus

Modal μ-calculus is a temporal logic augmented by operators for least and greatest fixed points. It is very expressive (any formula in LTL, CTL and CTL* can be encoded in the μ-calculus) and it is as expressive as *alternating tree automata*; in other words, for any formula in modal μ-calculus there is an equivalent ATA and vice versa.

1.2.1 Syntax

Let *Var* be a set of fixed point variables, *Prop* be a set of propositional variables:

$$\varphi, \psi \in \mathcal{L}_\mu ::= \bot \mid \top \mid X \mid p \mid \neg p \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Box\varphi \mid \Diamond\varphi \mid \mu X \varphi \mid \nu X \varphi$$

where p $\in$ *Prop*, $X \in$ *Var* and μ (ν) is the least (greatest) fixed point operator.

1.2.2 Semantics

Let K be a Kripke structure, then $\varphi \in \mathcal{L}_\mu$ is evaluated to $||\varphi||_K \subseteq \mathcal{W}^K$ in K

- $||\bot||_K = \emptyset, \qquad ||\top||_K = \mathcal{W}^K$
- $||p||_K = \kappa^K(p), \quad ||\neg p||_K = \mathcal{W}^K \setminus \kappa^K(p)$
- $||\varphi \vee \psi||_K = ||\varphi||_K \cup ||\psi||_K$
- $||\varphi \wedge \psi||_K = ||\varphi||_K \cap ||\psi||_K$
- $||\Box\varphi||_K = \{w \in \mathcal{W}^k | Scs_K(w) \subseteq ||\varphi||_K\}$
- $||\Diamond\varphi||_K = \{w \in \mathcal{W}^k | Scs_K(w) \cap ||\varphi||_K \neq \emptyset\}$

where $\mathrm{Scs}_K(w)$ is the set of all successors of w in K.

Let K [q $\mapsto$ W] = (W^K, A^K, κ^K[q $\mapsto$ W]) where κ^K[q $\mapsto$ W] is given by κ^K[q $\mapsto$ W](q$'$) = W if q$'$ = q; otherwise, κ^K[q $\mapsto$ W](q$'$) = κ^K(q$'$).

- $||\mu q \varphi||_K = \bigcap \{W \subseteq \mathcal{W}^k | ||\varphi||_{K[q \mapsto W]} \subseteq W\}$
- $||\nu q \varphi||_K = \bigcup \{W \subseteq \mathcal{W}^k | ||\varphi||_{K[q \mapsto W]} \supseteq W\}$

For more details, we refer to [1] which also presents a semantics from a different view; a *query-based* semantics.

1.3 Alternating tree automata

Alternating tree automata are finite-state devices designed to accept or reject pointed Kripke structures. They can deal with arbitrary branching in a very natural way.

1.3.1 Definition

An alternating tree automaton (ATA) is a tuple $\mathcal{A}$ = (S, s$_I$, δ, Ω) where:

- S is a finite set of *states*
- s$_I$ is an *initial state*

- δ is a *transition function*
- Ω: S $\rightarrow \omega$ is a *priority function*, which assigns a *priority* to each state

The transition function δ maps every state to a transition condition over S where the set of all *transition conditions* over S contains conditions of the form:

$$0,1,\ q,\ \neg q,\ s,\ \Box s,\ \Diamond s,\ s \wedge s',\ s \vee s'$$

for every s, s' $\in$ S and for every q $\in \mathcal{Q}$.

1.3.2 Runs

A *run* of an ATA $\mathcal{A}$ on ($\mathcal{K}$, w_0) is a (W$\times$S)-vertex labeled tree $R = (V^R, E^R, \lambda^R)$ where the initial vertex is labeled by (w_0, s_0) and every vertex v with label (w, s) the following conditions are satisfied (δ(s) $\neq$ 0):

δ(s)	Condition
q	w $\in \kappa^K$(q)
$\neg$q	w $\notin \kappa^K$(q)
$\Diamond$s'	there exists v' $\in Scs_R$(v) such that s^R(v') = s' and w$^R \in$ Scs$_K$(w)
$\Box$s'	for every w' $\in Scs_K$(w) there exists v' $\in Scs_R$(v) such that $\lambda(v') = (w', s')$
s' $\vee$ s''	there exists v' $\in$ Scs$_R$(v) such that $\lambda(v') = (w, s')$ or $\lambda(v') = (w, s'')$
s' $\wedge$ s''	there exists v', v'' $\in$ Scs$_R$(v) such that $\lambda(v') = (w, s')$ and $\lambda(v'') = (w, s'')$

A run is accepting if the state labeling of every infinite branch through R satisfies the parity acceptance condition determined by Ω.

1.3.3 Translation

Constructing an alternating tree automaton for every $\mathcal{L}_\mu$ formula φ that recognizes the exact query that the formula defines is straightforward (proof is more complicated).

Let A(φ) be the ATA of φ. The subformulas of φ build the states of A(φ); φ itself is the initial state. The transition function reflects the structure of the formula and the priority function reflects the alternation structure of the formula.

We define a normal form for $\mathcal{L}_\mu$ formulas. An $\mathcal{L}_\mu$ formula is in *normal form* if every propositional variable q is only quantified at most once and if in this case all occurrences of q are in the scope of this quantification. Every formula is equivalent to a formula in normal form of the same size and alternation depth.

Given an $\mathcal{L}_\mu$ formula φ in normal form and a propositional variable q occurring in φ. Either every occurrence of q in φ is free or every occurrence of q in φ is quantified by the same fixed point operator (it is bound in the same subformula denoted by φ_q). Let φ be an $\mathcal{L}_\mu$ formula in normal form. A(φ) is defined by A(φ) = ($\mathcal{S}$, s_I, δ, Ω) where:

- $\mathcal{S}$ the set which contains for each subformula ψ in φ a corresponding state
- $s_I = \langle\varphi\rangle$ is the initial state
- the transition function is defined by:
  - $\delta(\langle\bot\rangle) = 0$, $\delta(\langle\top\rangle) = 1$,
  - $\delta(\langle q\rangle) =$ q if q $\in$ free(φ), otherwise $\delta(\langle q\rangle) = \langle\varphi_q\rangle$; $\delta(\langle\neg q\rangle) = \neg$ q,
  - $\delta(\langle\psi \wedge \chi\rangle) = \langle\psi\rangle \wedge \langle\chi\rangle$, $\delta(\langle\psi \vee \chi\rangle) = \langle\psi\rangle \vee \langle\chi\rangle$,
  - $\delta(\langle\Diamond\psi\rangle) = \Diamond\,\langle\psi\rangle$, $\delta(\langle\Box\psi\rangle) = \Box\,\langle\psi\rangle$,
  - $\delta(\langle\mu q\psi\rangle) = \langle\psi\rangle$, $\delta(\langle\nu q\psi\rangle) = \langle\psi\rangle$.
- for every $\psi \in F_\mu$ with alternation depth > 0, $\Omega(\langle\psi\rangle) = 2\lceil\alpha(\psi)\rceil - 1$,
- for every $\psi \in F_\nu$ with alternation depth > 0, $\Omega(\langle\psi\rangle) = 2\lfloor\alpha(\psi)\rfloor$.

More details and a correctness proof are presented in [1].

2 Reduction to the acceptance problem for ATAs

The model checking problem can be reduced to the acceptance problem for alternating tree automata:

Model Checking: given a finite pointed Kripke structure ($\mathcal{K}$, w) and an $\mathcal{L}_\mu$ formula φ, determine whether or not ($\mathcal{K}$, w) $\models \varphi$.

Accepts: given a finite pointed Kripke structure ($\mathcal{K}$, w) and an alternating tree automaton $\mathcal{A}$, determine whether $\mathcal{A}$ accepts ($\mathcal{K}$, w).

3 Parity games

Formally, a *parity game* is a tuple $\mathcal{P} = (L_0, L_1, l_I, M, \Omega)$ where:

- L_0 and L_1 are disjoint sets, the sets of Player 0's and Player 1's locations, resp.
- $l_I \in L_0 \cup L_1$ is an *initial location*
- $M \subseteq (L_0 \cup L_1) \times (L_0 \cup L_1)$ is a set of *moves*, and

- Ω: $(L_0 \cup L_1) \rightarrow \omega$ is a *priority function* with a finite range.

$\mathcal{G}(\mathcal{P})$ is a directed graph called the *game graph* of $\mathcal{P}$. A partial play of $\mathcal{P}$ is a path through $\mathcal{G}(\mathcal{P})$ starting with l_I. A *play* of $\mathcal{P}$ is a maximum path through $\mathcal{G}(\mathcal{P})$ starting with l_I.

A play p is *winning* for Player 0 if it is infinite and sup(pΩ) is even or it is finite and p(*) $\in$ L_1. A play p is *winning* for Player 1 if it is infinite and sup(pΩ) is odd or it is finite and p(*) $\in$ L_0. A *winning strategy* for Player 0 makes sure that whatever Player 1 does in a play, it will be a win for Player 0. A *strategy tree* for Player 0 in $\mathcal{P}$ is a tree $\mathcal{T}$ satisfying the following conditions:

- The root of $\mathcal{T}$ is labeled l_I
- Every v $\in$ V^T with λ^T(v) $\in$ L_0 has a successor in $\mathcal{T}$ labeled with a successor of λ^T(v) in $\mathcal{G}(\mathcal{P})$ (Player 0 must move when it is his turn)
- Every v $\in$ V^T with λ^T(v) $\in$ L_1 has, for every successor l of λ^T(v) in $\mathcal{G}(\mathcal{P})$ a successor in $\mathcal{T}$ labeled l (all options of player 1 have to be taken into account)

A branch v of $\mathcal{T}$ is *winning* if its labeling, which is a play is winning. A strategy tree $\mathcal{T}$ for Player 0 is *winning* if every branch through $\mathcal{T}$ is winning. Player 0 wins a game $\mathcal{P}$ if there exists a winning strategy tree for him.

4 Reduction to the winner problem for parity games

The acceptance problem for ATAs can be reduced to the winner problem for parity games:

WINS: given a finite parity game $\mathcal{P}$, determine whether or not Player 0 wins the game $\mathcal{P}$.

- Construct a game $\mathcal{P} = (\mathcal{A}, \mathcal{K}, w_I)$ such that Player 0 wins if and only if $\mathcal{A}$ accepts $(\mathcal{K}, w_I)$
- Choices of Player 0: correspond to the choices $\mathcal{A}$ has to make when in a transition condition it has to satisfy a disjunction or a $\Diamond$ requirement
- Choices of Player 1: correspond to the choices $\mathcal{A}$ has to make when in a transition condition it has to satisfy a conjunctions or $\Box$ requirements

Formally, $\mathcal{P}(\mathcal{A}, \mathcal{K}, w_I)$ = (L_0, L_1, (w_I^K, s_I^A), M, Ω) where L_0 is the set of all pairs (w, s) where δ(s) is of the form 0, q with q $\notin$ κ^K(w), $\neg$ q with q $\in$ κ^K(w), s$'$ $\vee$ s'', or $\Diamond$s; this also determines L_1. The successors of a location (w,s) are determined by the rules in Table 2. The priority function Ω maps (w, s) to Ω^A(s).

Theorem 1: [1] *Let ($\mathcal{K}$, w) be a pointed Kripke structure and $\mathcal{A}$ an alternating tree automaton. $\mathcal{A}$ accepts ($\mathcal{K}$, w) if and only if Player 0 has a winning strategy.*

- Table 2 -

δ(s)	Condition
0, 1, q or $\neg$q	(w,s) has no successors
s$'$	(w,s) has one successor (w, s$'$)
s$'$ $\vee$ s$''$(s$'$ $\wedge$ s$''$)	(w,s) has two successors (w, s$'$) and (w, s$''$)
$\Diamond$s$'$ ($\Box$s$'$)	(w, s) has a successor (w$'$, s$'$) for every w$'$ $\in$ Scs_K(w)

Proof. Just observe that accepting runs of $\mathcal{A}$ on ($\mathcal{K}$, w) and winning strategy trees for player 0 in $\mathcal{P}$($\mathcal{A}$, $\mathcal{K}$, w) are identical.

Theorem 2: [1] WINS is in **UP** $\cup$ **co-UP**

5 References

[1] *T. Wilke, Alternating tree automata, parity games, and modal mu-calculus, Bull. Belg. Math. Soc., vol. 8, iss. 2, pp. 359391, 2002.*