Model-checking Games

Seminar on Games in Verification and Synthesis
(University of Saarland, Reactive Systems Group, Klaus Draeger)

Walid Haddad

May 29, 2008

Outline

Overview

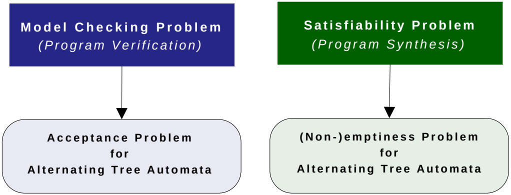
A model checking/synthesis approach:

Model Checking Problem
(Program Verification)

Satisfiability Problem
(Program Synthesis)

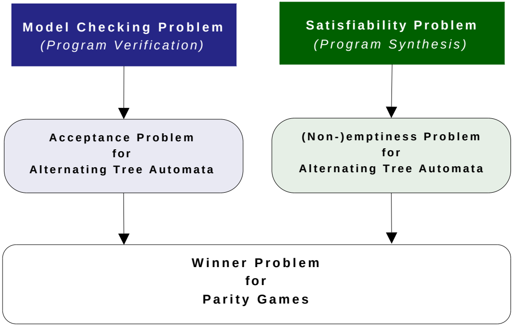
Overview

A model checking/synthesis approach:



Overview

A model checking/synthesis approach:



Model checking approach

For a system $\mathcal{S}$ and a specification $\mathcal{P}$, decide whether $\mathcal{S}$ satisfies $\mathcal{P}$, where:

- models of systems are represented as **Kripke structures**, and
- specifications are described in **modal μ-calculus**

Kripke structures

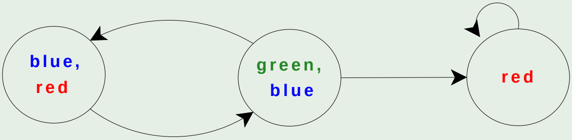
Definition

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

- W is a set of *worlds*
- $A \subseteq W \times W$ is an *accessibility relation*
- κ: $\mathcal{Q} \to 2^W$ 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 $(\mathcal{K}, \omega)$ where $\mathcal{K}$ is a Kripke structure and ω a world of it; a *Kripke query* is a class of pointed Kripke structures

Example



Modal μ-calculus

Modal μ-calculus is a temporal logic augmented by operators for least and greatest fixed points

- Used to express properties of *Kripke structures*
- Very expressive
  - LTL, CTL and CTL* can be encoded in the μ-calculus
  - as expressive as *alternating tree automata* (later)

Modal μ-calculus

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

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

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

Atomic formulas:

- $||\bot||_K = \emptyset$, $\qquad ||\top||_K = \mathcal{W}^K$

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

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

Atomic formulas:

- $||\bot||_K = \emptyset$, $\qquad ||\top||_K = \mathcal{W}^K$
- $||p||_K = \kappa^K(p)$, $\qquad ||\neg p||_K = \mathcal{W}^K \setminus \kappa^K(p)$

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

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

Disjunction and conjunction:

- $||\varphi \vee \psi||_K = ||\varphi||_K \cup ||\psi||_K$

Modal μ-calculus

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

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

Disjunction and conjunction:

- $||\varphi \vee \psi||_K = ||\varphi||_K \cup ||\psi||_K$
- $||\varphi \wedge \psi||_K = ||\varphi||_K \cap ||\psi||_K$

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

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

Modal operators:

- $||\Box\varphi||_K = \{\ w \in \mathcal{W}^k \mid Scs_K(w) \subseteq ||\varphi||_K\ \}$

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

Modal μ-calculus

Syntax

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

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

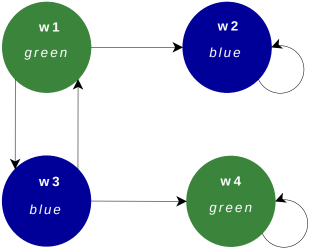
where p $\in$ *Prop*, $X \in$ *Var* and μ (ν) is the least (greatest) fixed point operator

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

Modal operators:

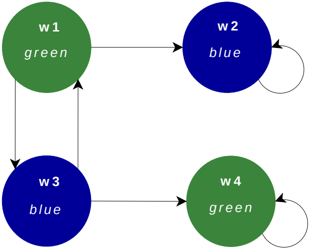
- $||\Box\varphi||_K = \{ w \in \mathcal{W}^k \mid Scs_K(w) \subseteq ||\varphi||_K \}$
- $||\Diamond\varphi||_K = \{ w \in \mathcal{W}^k \mid Scs_K(w) \cap ||\varphi||_K \neq \emptyset \}$

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



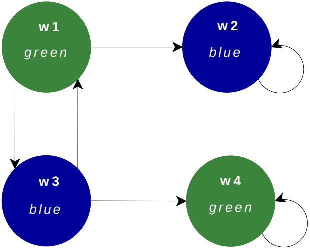
- $\varphi_0 = \mu x(\Box x)$

Modal μ-calculus - example

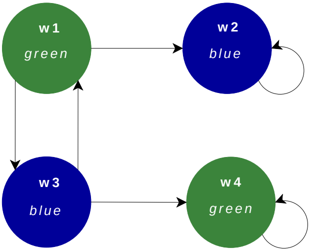


- $\varphi_0 = \mu x(\Box x)$
  - $||\varphi_0||_K = \emptyset$

Modal μ-calculus - example

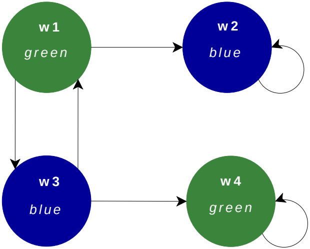


- $\varphi_0 = \mu x(\Box x)$
  - $||\varphi_0||_K = \emptyset$
- $\varphi_1 = \nu y(\textbf{green} \wedge \Box y)$



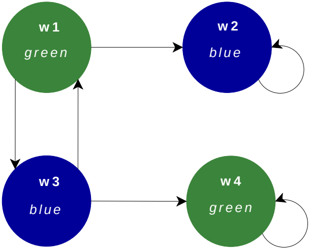
- $\varphi_0 = \mu x(\Box x)$
  - $||\varphi_0||_K = \emptyset$
- $\varphi_1 = \nu y(\textbf{green} \wedge \Box y)$
  - $||\varphi_1||_K = \{w_4\}$, (CTL: $\forall\Box$ **green**)

MODAL μ-CALCULUS - EXAMPLE



- $\varphi_0 = \mu x(\Box x)$
  - $||\varphi_0||_K = \emptyset$
- $\varphi_1 = \nu y(\textbf{green} \wedge \Box y)$
  - $||\varphi_1||_K = \{w_4\}$, (CTL: $\forall\Box$ **green**)
- $\varphi_2 = \mu x(\nu y(\textbf{green} \wedge \Box y) \vee \Diamond x)$

MODAL μ-CALCULUS - EXAMPLE



- $\varphi_0 = \mu x(\Box x)$
  - $||\varphi_0||_K = \emptyset$
- $\varphi_1 = \nu y(\textbf{green} \wedge \Box y)$
  - $||\varphi_1||_K = \{w_4\}$, (CTL: $\forall\Box$ **green**)
- $\varphi_2 = \mu x(\nu y(\textbf{green} \wedge \Box y) \vee \Diamond x)$
  - $||\varphi_2||_K = \{w_1, w_3, w_4\}$, (CTL: $\exists\Diamond\forall\Box$ **green**)

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

Definition

An alternating tree automaton (ATA) is a tuple $\mathcal{A} = (S, s_I, \delta, \Omega)$ 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 $q \in \mathcal{Q}$

Runs

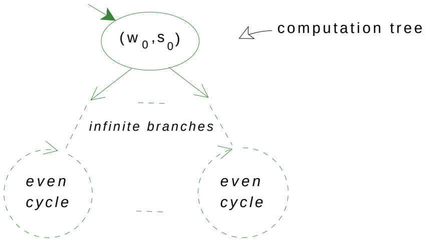
A *run* of an ATA $\mathcal{A}$ on $(\mathcal{K}, w_0)$ is a (W×S)-vertex labeled tree $R = (V^R, E^R, \lambda^R)$ where the initial vertex is labeled by (w_0, s_0) and every vertex v with label (w, s) the following conditions are satisfied ($\delta(\mathrm{s}) \neq 0$):

$\delta(\mathrm{s})$	Condition
q	$\mathrm{w} \in \kappa^K(\mathrm{q})$
$\neg$q	$\mathrm{w} \notin \kappa^K(\mathrm{q})$
$\Diamond \mathrm{s}'$	there exists $\mathrm{v}' \in Scs_R(\mathrm{v})$ such that $s^R(\mathrm{v}') = \mathrm{s}'$ and $\mathrm{w}^R(v') \in \mathrm{Scs}_K(\mathrm{w})$
$\Box \mathrm{s}'$	for every $\mathrm{w}' \in Scs_K(\mathrm{w})$ there exists $\mathrm{v}' \in Scs_R(\mathrm{v})$ such that $\lambda(v') = (w', s')$

Runs (contd.)

$\delta(s)$	Condition
$s' \vee s''$	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



TRANSLATION: FROM μ-CALCULUS TO ATAS

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

Example

Let $\varphi = \mu q_1(q_0 \vee \Diamond q_1)$. Construct the corresponding automaton $\mathcal{A}$.

- We construct a state $\langle\psi\rangle$ for every subformula ψ of φ:

  $\langle\mu q_1(q_0 \vee \Diamond q_1)\rangle$, $\langle q_0 \vee \Diamond q_1\rangle$, $\langle q_0\rangle$, $\langle\Diamond q_1\rangle$, $\langle q_1\rangle$

Translation: from μ-calculus to ATAs

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

Example

Let $\varphi = \mu q_1(q_0 \vee \Diamond q_1)$. Construct the corresponding automaton $\mathcal{A}$.

- We construct a state $\langle \psi \rangle$ for every subformula ψ of φ:

  $$\langle \mu q_1(q_0 \vee \Diamond q_1)\rangle, \langle q_0 \vee \Diamond q_1\rangle, \langle q_0\rangle, \langle \Diamond q_1\rangle, \langle q_1\rangle$$

- The transition function is given by:

$$\begin{aligned}
\delta(\langle \mu q_1(q_0 \vee \Diamond q_1)\rangle) &= \langle q_0 \vee \Diamond q_1\rangle, \\
\delta(\langle q_0 \vee \Diamond q_1\rangle) &= \langle q_0\rangle \vee \langle \Diamond q_1\rangle, \\
\delta(\langle q_0\rangle) &= q_0, \\
\delta(\langle \Diamond q_1\rangle) &= \Diamond \langle q_1\rangle, \\
\delta(\langle q_1\rangle) &= \langle \mu q_1(q_0 \vee \Diamond q_1)\rangle
\end{aligned}$$

Example (contd.)

- The definition of the transition function can be shortened to:

$$\delta(\langle \mu q_1(q_0 \vee \Diamond q_1)\rangle) = \langle q_0\rangle \vee \langle \Diamond q_1\rangle,$$
$$\delta(\langle q_0\rangle) = q_0,$$
$$\delta(\langle \Diamond q_1\rangle) = \Diamond \langle \mu q_1(q_0 \vee \Diamond q_1)\rangle$$

- $\langle \mu q_1(q_0 \vee \Diamond q_1)\rangle$ is the initial state; it gets priority 1 (all other states get priority 0)

Reduction to the acceptance problem (ATAs)

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

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

Reduction to the acceptance problem (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) $\models \varphi$

$$\downarrow$$

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)

Definition

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

- L_0 and L_1 are disjoint sets, the sets of Player 0's and Player 1's locations, resp.
- $\mathsf{l}_I \in L_0 \cup L_1$ is an *initial location*
- $\mathsf{M} \subseteq (L_0 \cup L_1) \times (L_0 \cup L_1)$ 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

Parity games

- A play p is *winning* for Player 0 if it is infinite and $\sup(p\Omega)$ is even or it is finite and $p(*) \in L_1$
- A play p is *winning* for Player 1 if it is infinite and $\sup(p\Omega)$ 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

Parity games

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

Winning conditions:

- 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

Reduction of the acceptance problem

- 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

Reduction of the acceptance problem

Formally, $\mathcal{P}(\mathcal{A}, \mathcal{K}, w_I) = (L_0, L_1, (w_I^K, s_I^A), M, \Omega)$ where:

- L_0 is the set of all pairs (w, s) where $\delta(s)$ is of the form 0, q with $q \notin \kappa^K(w)$, $\neg$ q with $q \in \kappa^K(w)$, $s' \vee s''$, or $\Diamond s$; this also determines L_1
- The successors of a location (w,s) are determined by the following rules:

$\delta(s)$	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 $w' \in Scs_K(w)$

- The priority function Ω maps (w, s) to $\Omega^A(s)$

Reduction of the acceptance problem

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

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)

Reduction of the acceptance problem

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

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)

$$\downarrow$$

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

Conclusion

- Model checking modal μ-calculus can be reduced to the winner problem for parity games
- Wins (for finite parity games) is solvable in $\mathcal{O}(m(2n/b)^{\lfloor b/2 \rfloor})$ (P-hard)

References

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

THANK YOU!