Chapter 4

μ-calculus model checking

Course “Model checking”

Volker Stolz, Martin Steffen

Autumn 2019

Chapter 4

Learning Targets of Chapter "μ-calculus model checking".

The chapter covers an short intro to the (resp. one variant) of the μ-calculus and model-checking it. We focus on the most prominent version of the μ-calculus for model checking known as modal μ-calculus, with a "branching time" interpretation. The logic can be understood as the "prototypical" **logic with fixpoints**, so we'll have to talk about fixpoints as well. For model checking, we look at a bit of "game theory" (parity games).

Chapter 4

Outline of Chapter "μ-calculus model checking".

Section

Introduction

Chapter 4 “μ-calculus model checking”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Intro remarks

- rather fundamental logic
- central to μ-calculus: fixpoints
- many variations (and names)
  - propositional μ-calculus
  - modal μ-calculus
  - Hennessy-Milner logic with recursion
  - ....

For the lecture: vanilla μ-calculus

a plain, propositional modal logic + fixpoints

What's a fixpoint?

$f : A \to A$

$$f(a) = a$$

Fixpoints are everywhere

A pedestrian definition of syntax

The set Φ of *propositional formulas* is given as follows

- all propositional constants from AP are formulas
- if φ is a formula, then so is $\neg\varphi$
- if φ_1 is a formula and φ_2 is a formula, then so is $\varphi_1 \wedge \varphi_2$
- if φ_1 is a formula and φ_2 is a formula, then so is $\varphi_1 \vee \varphi_2$
- ... [more constructs if wished] ...

Is that even a definition?

Fixpoints are everywhere

A pedestrian definition of syntax (reformulated)

The set Φ of *propositional formulas* is given as follows

- $AP \subseteq \Phi$
- if $\varphi \in \Phi$, the $\neg\varphi \in \Phi$
- if $\varphi_1 \in \Phi$ and $\varphi_2 \in \Phi$, then $\varphi_1 \wedge \varphi_2 \in \Phi$
- if $\varphi_1 \in \Phi$ and $\varphi_2 \in \Phi$, then $\varphi_1 \vee \varphi_2 \in \Phi$
- . . . [more constructs if wished] . . .

Fixpoints are everywhere

A pedestrian definition of syntax (reformulated)

The set Φ of *propositional formulas* is given as follows

- $AP \subseteq \Phi$
- if $\varphi \in \Phi$, the $\neg\varphi \in \Phi$
- if $\varphi_1 \in \Phi$ and $\varphi_2 \in \Phi$, then $\varphi_1 \wedge \varphi_2 \in \Phi$
- if $\varphi_1 \in \Phi$ and $\varphi_2 \in \Phi$, then $\varphi_1 \vee \varphi_2 \in \Phi$
- ... [more constructs if wished] ...

What about that?

$$\Phi = \{p, q, ..., p \wedge p, p \wedge q, p \wedge (p \vee q) \ldots\} \cup \{5, p\#q, \neg 5, 5 \wedge (q\#q) \ldots\}$$

How to fix(-point) it?

- ... No other entities are formulas, i.e. elements of Φ
- Φ is the smallest set such that 1) $AP \subseteq \Phi$, 2) ...
- Φ is inductively given by the following conditions: 1) ...

"Mu"

$$F(S) = AP \cup \{\neg\phi \mid \phi \in S\} \cup \{\varphi_1 \wedge \varphi_1 \mid \varphi_1 \in S, \varphi_2 \in S\} \cup \ldots$$

$$\Phi = \mu F$$

Fixpoints are everywhere, indeed

- grammars (with special syntax)

$$\varphi ::= AP \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \ldots$$

- Kleene star and regular expressions:
  - finite words over Σ: written Σ^*
  - $a(b+c)^*$
- semantics of programming language
  - `while`-loop: make single steps, until termination (but not more!)
- data structures
  - the natural numbers are given (“constructed”) by 0 and $succ$ as constructor (and not more!)
- proof (and proof trees): a proof is given (“constructed”) from *axioms* and application of *rules* (but not more!)

Connection to induction

remember: one way of formulation

"Φ is inductively given as follows ...

Natural numbers

1. 0 is a natural number
2. if n is a natural number, then so is $succ(n)$ (written also $n + 1$)
3. $n + 1 = m + 1$, then $n = m$
4. there is no natural number n with $n + 1 = 0$

Connection to induction

remember: one way of formulation

"Φ is inductively given as follows ...

Natural numbers

1. 0 is a natural number
2. if n is a natural number, then so is $succ(n)$ (written also $n+1$)
3. $n+1 = m+1$, then $n = m$
4. there is no natural number n with $n+1 = 0$

Peano Nr. 5

$$\text{If a set } S \text{ contains } 0 \text{ and is closed under successor, then } \textbf{all} \text{ natural numbers are in } S. \tag{1}$$

Connection to induction

remember: one way of formulation

"Φ is inductively given as follows ...

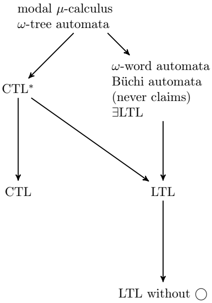
Natural numbers

1. 0 is a natural number
2. if n is a natural number, then so is $succ(n)$ (written also $n+1$)
3. $n+1 = m+1$, then $n = m$
4. there is no natural number n with $n+1 = 0$

Peano Nr. 5: natural induction

$$\forall\varphi.\varphi(0) \wedge (\forall n.\varphi(n) \rightarrow \varphi(n+1)) \rightarrow \forall n.\varphi(n)\ . \tag{2}$$

Expressivity



Section

Propositional μ-calculus: syntax and semantics

Syntax
Background: Fixpoints
Semantics

Chapter 4 “μ-calculus model checking”
Course “Model checking”
Volker Stolz, Martin Steffen
Autumn 2019

Labelled transition systems

A *transition system* is a tuple

$$(S, \to, \{P_i\}_{i \in \mathbb{N}})$$

- $AP = \{p_0, p_1, p_2, \ldots\}$
- Act: $(= \Sigma)$ *actions* $a, b', \ldots$
- $\to \subseteq S \times Act \times S$
  - $s \xrightarrow{a} s'$, a-transition, from s to s'
- $P_i \subseteq S$
- note: switch of perspective for "proposition labelling"

Syntax

$$
\begin{array}{lllll}
\varphi & ::= & p \mid \neg p & & \text{props and their negation} \\
& \mid & X & & \text{variables} \\
& \mid & \varphi \wedge \varphi & \mid \ \varphi \vee \varphi & \text{2 usual boolean connectives} \\
& \mid & [a]\varphi & \mid \ \langle a \rangle \varphi & \text{(multi)-modal operators} \\
& \mid & \mu X.\varphi & \mid \ \nu X.\varphi & \text{fix-points}
\end{array}
$$

- $true$ and $false$: $p \wedge \neg p$ resp. $p \vee \neg p$
- variables $X, Y \ldots \in Var$
- actions $a, b' \ldots \in Act$

Remarks on the syntax

- general negation: missing
  - especially $\neg X$ not part of the syntax
  - but: $\neg\varphi$ *definable*
- μ and ν (or σ when unspecific): binding operators
  - free and bound occurrences of variables
  - renaming of bound variables (α-renaming)
- some well-formedness conditions
  - don't reuse variables
  - guardedness

What about the variables?

- propositional μ-calculus
- $X, Y \ldots$ *different* from first-order logic variables
- variables here represent
  - formulas (from μ-calculus), resp.
  - semantically: *sets of states*

$\Rightarrow$ second-order flavor!

Preview on the semantics

- given transition system $\mathcal{M}$

Satisfaction relation

$$s \models_{\mathcal{M}} \varphi$$

Preview on the semantics

- given transition system $\mathcal{M}$

Satisfaction relation

$$s \models^{\mathcal{V}}_{\mathcal{M}} \varphi \tag{3}$$

Preview on the semantics

- given transition system $\mathcal{M}$

Satisfaction relation

$$s \models^{\mathcal{V}}_{\mathcal{M}} \varphi \tag{3}$$

Equivalently: semantic function

semantics of φ in transition system $\mathcal{M}$ and with valuation $\mathcal{V}$:

$$[\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}} \in 2^S \tag{4}$$

$$\mathcal{V} : \mathit{Var} \to 2^S$$

Semantics (no fix-points)

$$\varphi ::= p_i \ \mid \ \neg p_i \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \varphi_1 \vee \varphi_2 \ \mid \ [a]\varphi \ \mid \ \langle a \rangle \varphi$$

$$\begin{array}{rclrcl}
[\![true]\!]^{\mathcal{M}}_{\mathcal{V}} & = & S & [\![false]\!]^{\mathcal{M}}_{\mathcal{V}} & = & \emptyset \\
[\![p_i]\!]^{\mathcal{M}}_{\mathcal{V}} & = & P_i & [\![\neg p_i]\!]^{\mathcal{M}}_{\mathcal{V}} & = & S - P_i \\
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}} & = & [\![\varphi_1]\!]^{\mathcal{M}}_{\mathcal{V}} \cap [\![\varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}} & [\![\varphi_1 \vee \varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}} & = & [\![\varphi_1]\!]^{\mathcal{M}}_{\mathcal{V}} \cup [\![\varphi_2]\!]^{\mathcal{M}}_{\mathcal{V}}
\end{array}$$

$$\begin{array}{rcl}
[\![[a]\varphi]\!]^{\mathcal{M}}_{\mathcal{V}} & = & \{s \in S \mid \forall s'.s \xrightarrow{a} s' \Rightarrow s' \in [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}}\} \\
[\![\langle a \rangle\varphi]\!]^{\mathcal{M}}_{\mathcal{V}} & = & \{s \in S \mid \exists s'.s \xrightarrow{a} s' \ \wedge \ s' \in [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}}\}
\end{array}$$

Fixpoints in LTL

- $\Box p$?
- $\Diamond p$
- $p \ U \ q$

Reconsider for instance $\Box p$?

- fix-point equation for "always"?

$$\Box p = p \wedge \bigcirc \Box p. \tag{5}$$

Reconsider for instance $\Box p$?

- fix-point equation for "always"?

$$\Box p = p \wedge \bigcirc \Box p. \tag{5}$$

choose $\Box p = false$

$$false = p \wedge \bigcirc false \tag{6}$$

Orders, lattices, etc.

as a reminder:

- partial order $(L, \sqsubseteq)$
- *upper bound* l of $Y \subseteq L$:
- *least* upper bound (lub): $\bigsqcup Y$ (or *join*)
- dually: lower bounds and greatest lower bounds: $\bigsqcap Y$ (or *meet*
- complete lattice $L = (L, \sqsubseteq) = (L, \sqsubseteq, \bigsqcap, \bigsqcup, \bot, \top)$: a partially ordered set where meets and joins exist for *all subsets*, furthermore $\top = \bigsqcap \emptyset$ and $\bot = \bigsqcup \emptyset$.

Fixpoints

given complete lattice L and monotone $f : L \to L$.

- fixpoint: $f(l) = l$

$$Fix(f) = \{l \mid f(l) = l\}$$

- f *reductive* at l, l is a pre-fixpoint of f: $f(l) \sqsubseteq l$:

$$Red(f) = \{l \mid f(l) \sqsubseteq l\}$$

- f *extensive* at l, l is a post-fixpoint of f: $f(l) \sqsupseteq l$:

$$Ext(f) = \{l \mid f(l) \sqsupseteq l\}$$

Define "lfp" / "gfp"

$$lfp(f) \triangleq \bigsqcap Fix(f) \quad \text{and} \quad gfp(f) \triangleq \bigsqcup Fix(f)$$

Tarski's theorem

Core

Perhaps core insight of the whole lattice/fixpoint business: not only does the $\bigsqcap$ of all pre-fixpoints uniquely exist (that's what the lattice is for), but —and that's the trick— *it's a pre-fixpoint* itself (ultimately due to montonicity of f).

Theorem

L: complete lattice, $f : L \to L$ monotone.

$$\begin{array}{rcccl} lfp(f) & \triangleq & \bigsqcap Red(f) & \in & Fix(f) \\ gfp(f) & \triangleq & \bigsqcup Ext(f) & \in & Fix(f) \end{array} \tag{7}$$

- Note: lfp (despite the name) is *defined* as glb of all pre-fixpoints
- The theorem (more or less directly) implies lfp is the *least* fixpoint

Fixpoint iteration

- often: iterate, approximate least fixed point from below $(f^n(\bot))_n$:

$$\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \dots$$

- not assured that we "reach" the fixpoint ("within" ω)

$$\begin{array}{lcl} \bot \sqsubseteq f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) & \sqsubseteq & \mathit{lfp}(f) \\ & & \mathit{gfp}(f) \quad \sqsubseteq \bigsqcap_n f^n(\top) \sqsubseteq f^n(\top) \sqsubseteq (\top) \end{array}$$

- additional requirement: continuity on f for all ascending chains $(l_n)_n$

$$f(\bigsqcup_n (l_n)) = \bigsqcup (f(l_n))$$

Semantics of formulas with free variables

- apply the FP theorem (Knaster-Tarski)
- assume $\mu X.\varphi(X)$ or $\nu X.\varphi(X)$
- $\mathcal{M}$ with state set S: fixed
- consider semantics of body $\varphi(X)$,
  - assume (for simplicity), only one free variable X

$$[\![\varphi(X)]\!]^{\mathcal{M}} : 2^S \to 2^S$$

- 2 welcome facts
  1. 2^S a complete lattice
  2. the function is monotone (and also continuous, under reasonable assumptions)
- general case: φ may have more variables than just X

$$f(S') = [\![\varphi]\!]^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']} : 2^S \to 2^S$$

with $S' \subseteq S$

Semantics of the fixpoints

$$\llbracket \mu X.\varphi \rrbracket^{\mathcal{M}}_{\mathcal{V}} = \bigsqcap \{S' \subseteq S \mid \llbracket \varphi \rrbracket^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']} \subseteq S'\} \qquad \text{(lfp)}$$
$$= \bigsqcap \{S' \subseteq S \mid f(S') \subseteq S'\}$$
$$\llbracket \nu X.\varphi \rrbracket^{\mathcal{M}}_{\mathcal{V}} = \bigsqcup \{S' \subseteq S \mid S' \subseteq \llbracket \varphi \rrbracket^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']}\} \qquad \text{(gfp)}$$
$$= \bigsqcup \{S' \subseteq S \mid S' \subseteq f(S')\}$$

where $f(S') = \llbracket \varphi \rrbracket^{\mathcal{M}}_{\mathcal{V}[X \mapsto S']}$

Alternation of fixpoints

- expressivity of μ-calculus: due to fix-points
- more precisely: "nesting" of fix.points
- even more precisely: alternation-depth of nested fixpoints.
- compare: direct recursion vs. mutual recursion
- similarly: "$\mu^2 = \mu$"
- technical definition of nesting: not 100% immediate
  1. $(\mu X.\varphi) \wedge (\nu X.\varphi_2)$: no nesting
  2. $\mu X.\mu Y \varphi(X, Y)$: no alternation
  3. $\nu X.((\mu Y.p \vee \langle b \rangle Y) \wedge [a]X)$: ??
  4. ???

Section

Model checking

Chapter 4 "μ-calculus model checking"
Course "Model checking"
Volker Stolz, Martin Steffen
Autumn 2019

Game

Definition (Game)

A game is a triple $G = (V, T, Acc)$ where

1. V are *nodes* partitioned between two players, Adam and Eve: $V = V_A + V_E$.
2. $T \subseteq V \times V$ is a *transition relation* determining the possible successors of each node, and
3. $Acc \subseteq V^\omega$ is a set defining the *winning condition*

- node: aka position
- Acc: *winning condition*

Playing a game like this

- two-player game
- two kinds of nodes (Eve's and Adam's)
- game "moves" through positions
  - in one of Eve's nodes: *Eve* chooses
  - analogous for Adam
- winning:
  - winning condition: from the perspective of Eve
    - ininite path through G: if Acc satisfied, Eve wins, otherwise Adam
  - a player "stuck": looses as well
  - no draw possible
- winning *node*: $\exists$ winning strategy

strategy θ (for Eve)

Given G. For each sequence of nodes $\vec{v}$, ending in a node $v \in V_E$: *choose* $\theta(\vec{v}) = v'$, such that $v \to v'$

Games as general framework

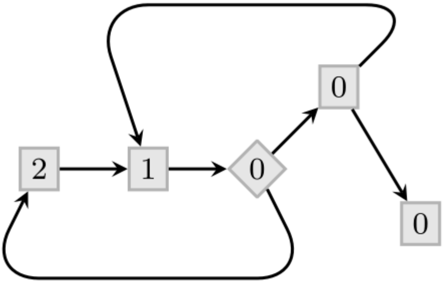
- "game theory": broad field with many applications
- here: game used for
  - semantics, logics, model-checking
- situation often: open systems
  - enviroment context $\leftrightarrow$ program
  - attacker $\leftrightarrow$ system
  - controllable $\leftrightarrow$ non-controllable parts

Game

Different, players with own "goals" (conflicting or at least different) and local "influence" or control.

- many variations
  - 2-player, multi-player
  - zero-sum games (no win/win situation in those...)
  - restricted information
  - probabilistic ("mixed") strategies (Nash-equilibrium!)
  - ...

Game example



- nodes or positions in the graph
  - Eve: “diamond”-shaped
  - Adam: “box”-shaped
- winning condition (here): Eve wins, if the game passes through “2” infinitely many times
- numbers in the nodes: “ranks” (see later)

Positional strategies

- stategy in general $\theta(\vec{v}v) = v'$
- in the example: strategy of Eve: can be dependent on the "current node" only

$\Rightarrow$ memoryless or positional

$$\theta(v) = v'$$

Parity games

given game G

Parity winning condition

ranking: $\Omega : V \to \{0, 1, \ldots, d\}$

$$Acc = \{\vec{v} \in V^\omega \mid \lim_{i\to\infty} \sup \Omega(v_i) \text{ is even}\}$$

Mostowski [2], Emerson and Jutla [1]

Theorem (PWC theorem)

- *every position is winning for one of the two players*
- *it's winnable by a positional strategy*
- *it's decidable who wins*

Model checking μ-calculus and parity games

Verification game

$$\mathcal{M}, s \models \varphi \quad \Rightarrow \quad \mathcal{G}(\mathcal{M}, \varphi)$$

$\mathcal{M}, s \models \varphi$ iff Eve wins from position corresponding to s and φ in $\mathcal{G}(\mathcal{M}, \varphi)$

- $\mathcal{V}$: valuation for free vars, i.e., game $\mathcal{G}_{\mathcal{V}}(\mathcal{M}, \varphi)$
- positions in the game

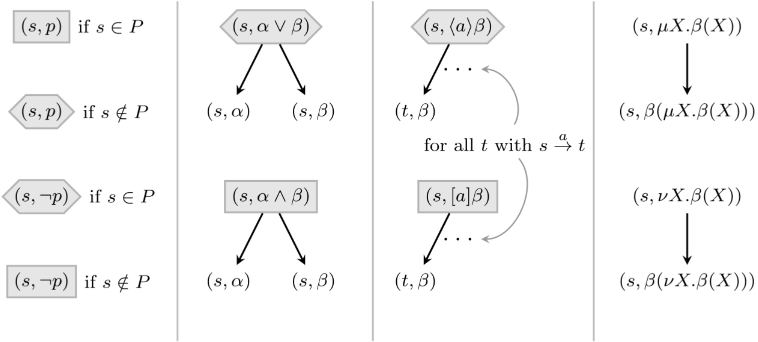
$$(s, \psi)$$

ψ: 1 formula from the closure of φ

Intention of the construction

Eve has winning strategy from (s, ψ) iff $\mathcal{M}, \mathcal{V}, s \models \psi$

Rules of the verification game



Ranking

- ranking positions with fp- formulas $\nu.\psi$ or $\mu.\psi$

μ-formula	ν-formula
odd.	even

- additionally: subformulas should must no higher ranks than the surrounding formula
- standard way: connect to alternation depth

$$
\begin{array}{rcll}
\Gamma(\psi) & = & 2\lfloor adepth(X)/2 \rfloor & \text{for } \psi = \nu X.\psi'(X) \\
\Gamma(\psi) & = & 2\lfloor adepth(X)/2 \rfloor + 1 & \text{for } \psi = \mu X.\psi'(X) \\
\Gamma(\psi) & = & 0 & \text{otherwise}
\end{array}
$$

References I

Bibliography

[1] Emerson, E. A. and Jutla, C. (1991). Tree automata, mu-calculus, and determinacy. In *IEEE Symposium on Foundations of Computer Science (FOCS'91)*.

[2] Mostowski, A. W. (1984). Regular expressions for infinite trees and a standard form of automata. In Skowron, A., editor, *Computation Theory*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer Verlag.