Logic, Automata, and Games

Sophie Pinchinat

IRISA, university of Rennes 1, France

M2RI 2011-2012

The Model-Checking Problem

The Model-checking Problem: A system *Sys* and a specification *Spec*, decide whether *Sys* satisfies *Spec*, or not.

Example: Mutual exclusion protocol

Process 0: repeat
00: non-critical section 1
01: wait unless turn $= 0$
10: critical section 1
11: turn $:= 1$

Process 1: repeat
00: non-critical section 2
01: wait unless turn $= 1$
10: critical section 2
11: turn $:= 0$

- A state is a bit vector of the form (line no. of process 1,line no. of process 2, value of turn)
- The initial state is (00000).
- *Spec* = "some state of the form (1010x) is never reached", and "always when a state of the form (01xyz) is reached, then later a state of the form (10x'y'z') is reached" (and similarly for Process 2, i.e. states (xy01z) and (x'y'10z'))

Kripke Structures

Assume given $Prop = \{p_1, \ldots, p_n\}$ a set of atomic propositions.

Definition

A Kripke structure over *Prop* is $\mathcal{S} = (S, R, \lambda)$

- S is a set of states
- $R \subseteq S \times S$ is a transition relation
- $\lambda : S \rightarrow 2^{Prop}$ associates those p_i which are assumed true in s.

A rooted Kripke structure is a pair $(\mathcal{S}, s)$ where s is a distinguished initial state

Mutual Exclusion Protocol Example

Let us use

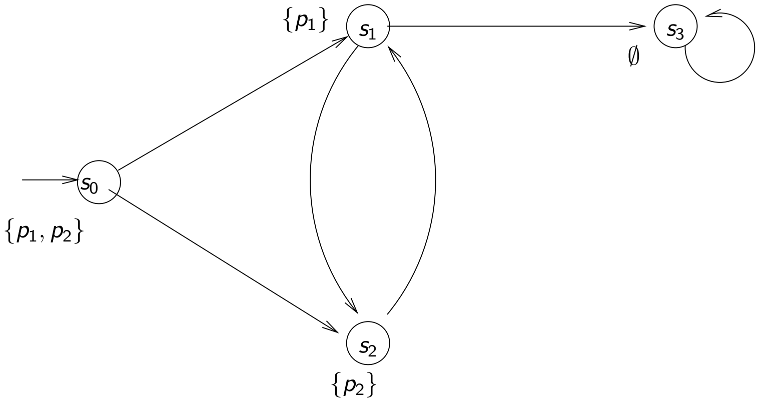
- Use p_1 and p_2 for “being in wait instruction before critical section” for Process 0 and Process 1 respectively
- Use p_3 and p_4 for “being in critical section” for Process 0 and Process 1 respectively

The label function looks like $\lambda(01101) = \{p_1, p_4\}$; remember states are (line no. of process 1,line no. of process 2, value of turn)

EXERCISE: Define the KS corresponding to the Mutual Exclusion Protocol

A Toy System

Over $Prop = \{p_1, p_2\}$.



$\lambda(s_2) = \{p_2\}$

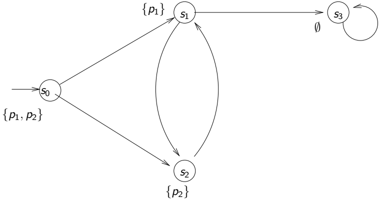
Paths and Words

Let $\mathcal{S} = (S, R, \lambda)$ be a Kripke structure over $Prop = \{p_1, p_2, \ldots, p_n\}$.

- A path through $(\mathcal{S}, s)$ is a sequence $s_0, s_1, s_2, \ldots$ where $s_0 = s$ and $(s_i, s_{i+1}) \in R$ for $i \geq 0$
- Its corresponding word $(\in (2^{Prop})^\omega)$ is $\lambda(s_0), \lambda(s_1), \lambda(s_2), \ldots$.

  For example,

  $$\alpha = \{p_1, p_2\}\{p_1\}\{p_2\}\{p_1\}\emptyset\emptyset\emptyset\ldots$$



- If $\alpha = \alpha(0)\alpha(1)\ldots \in (2^{Prop})^\omega$, write α^i for $\alpha(i)\alpha(i+1)\ldots$.
  So $\alpha = \alpha^0$.

Linear Time Logic for Properties of Words

[Eme90] We use modalities

G	denotes	"*Always*"
F	denotes	"*Eventually*"
X	denotes	"*Next*"
U	denotes	"*Until*"

The syntax of the logic LTL is:

$$\varphi_1, \varphi_2 (\ni LTL) ::= a \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \mathbf{X}\,\varphi_1 \mid \varphi_1 \,\mathbf{U}\, \varphi_2$$

where $a \in \Sigma$. LTL formulas are interpreted over words $\alpha \in \Sigma^\omega$.

Note that the words may arise from a Kripke structure $(\mathcal{S}, s)$ over *Prop* so that $\Sigma = 2^{Prop}$.

Semantics of LTL

Let $\alpha \in \Sigma^\omega$. Define $\alpha^i \models \varphi$ by induction over φ.

- $\alpha^i \models a$ iff $\alpha(i) = a$
- $\alpha^i \models \varphi_1 \vee \varphi_2$ iff ...
- $\alpha^i \models \neg\varphi_1$ iff
- $\alpha^i \models \mathbf{X}\,\varphi_1$ iff $\alpha^{i+1} \models \varphi_1$
- $\alpha^i \models \varphi_1 \,\mathbf{U}\, \varphi_2$ iff for some $j \geq i$, $\alpha^j \models \varphi_2$, and
  for all $k = i, \ldots, j-1$, $\alpha^k \models \varphi_1$

Let $\begin{cases} \mathbf{F}\varphi \overset{\text{def}}{=} \texttt{true}\,\mathbf{U}\,\varphi, \text{ hence } \alpha^i \models \mathbf{F}\varphi \text{ iff } \alpha^j \models \varphi \text{ for some } j \geq i. \\ \mathbf{G}\varphi \overset{\text{def}}{=} \neg\mathbf{F}\neg\varphi, \text{ hence } \alpha^i \models \mathbf{G}\varphi_1 \text{ iff } \alpha^j \models \varphi_1 \text{ for every } j \geq i. \end{cases}$

Examples of formulas

1. $\alpha \models \mathbf{GF}a$ iff "in α, a occurs infinitely often".
2. $\alpha \models \mathbf{X}\,\mathbf{X}\,(b \;\Rightarrow\; \mathbf{F}c)$ iff "If $\alpha(2) = b$, then $\alpha(j) = c$ for some $j \geq 2$".
3. $\alpha \models \mathbf{F}(a \wedge \mathbf{X}\,(b\,\mathbf{U}\,a))$ iff "... " (EXERCISE)

Augmenting LTL: the logic CTL^*

We want to specify that every word of $(\mathcal{S}, s)$ satisfies an LTL specification φ, or that there exists a word in the Kripke structure such that something holds. We use CTL^* [EH83] which extends LTL with quantfications over words:

$$\psi_1, \psi_2 (\ni CTL^*) ::= \mathbf{E}\,\psi \mid a \mid \psi_1 \vee \psi_2 \mid \neg\psi_1 \mid \mathbf{X}\,\psi_1 \mid \psi_1\,\mathbf{U}\,\psi_2$$

Semantics: for a word α, a position i, and a rooted Kripke structure $(\mathcal{S}, s)$:

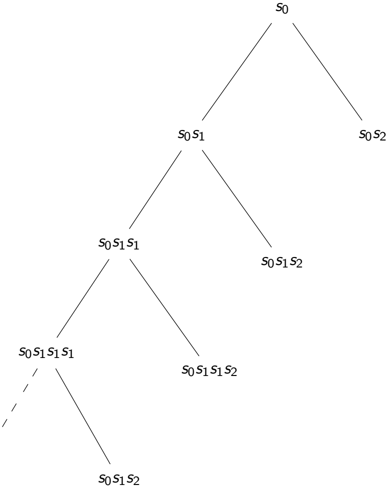
$$\alpha^i \models_{(\mathcal{S},s)} \mathbf{E}\,\psi \quad \text{iff} \quad \alpha'^i \models_{(\mathcal{S},s)} \psi \text{ for some } \alpha' \text{ in } (\mathcal{S}, s) \text{ st. } \alpha[0,\ldots,i] = \alpha'[0,\ldots,i]$$

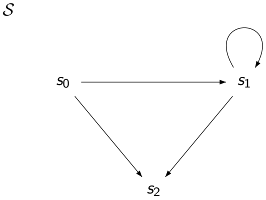
Let $\mathbf{A}\,\psi \stackrel{\text{def}}{=} \neg\mathbf{E}\,\neg\psi$

CTL* is more expressive than LTL: $\mathbf{A}\,[\mathbf{G}\text{life} \;\Rightarrow\; \mathbf{GEX}\,\text{death}]$

Interpretation over Trees

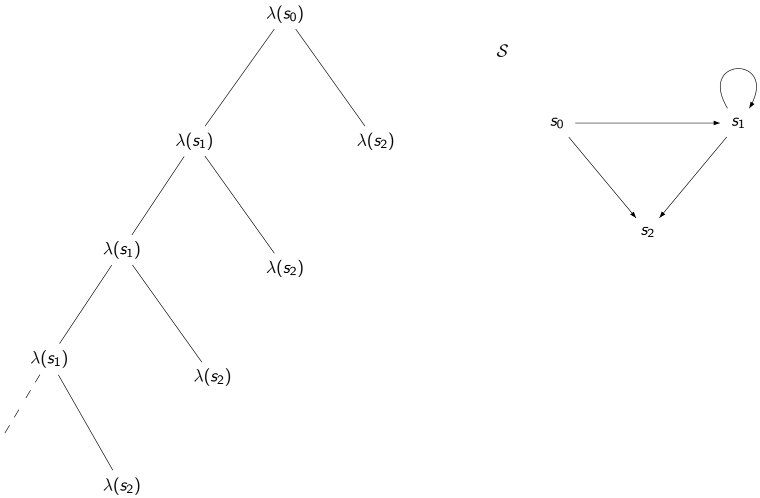
- We unravel $\mathcal{S} = (S, R, \lambda)$ from s as a tree
- Paths of $\mathcal{S}$ are retrieved in the tree as branches.





Interpretation over Trees

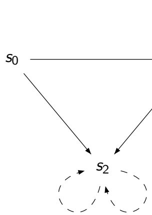
- In the tree, we keep only the information about propositions in the current state along the path.



Interpretation over Trees

- We keep from the unraveling information about propositions
- We assume that states have exactly two successors (ordered)

$\mathcal{S}$



EXERCISE draw the corresponding tree

We make a huge simplification:

we consider only **Kripke structures which unravel as full binary trees**

but the theory generalizes to arbitray structures.

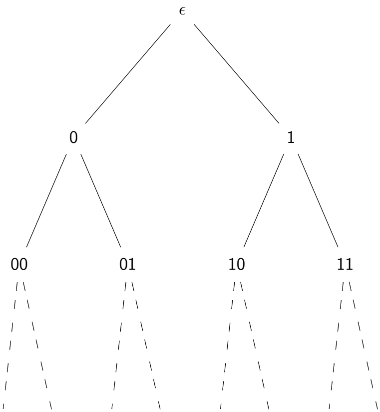
Σ-Labeled Full Binary Trees

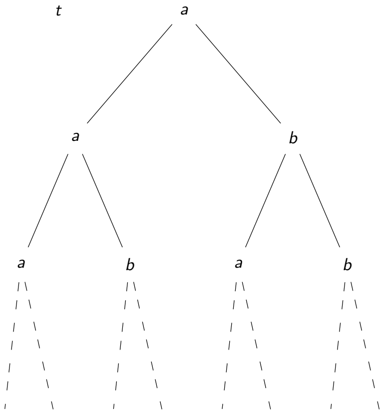
- The full binary tree is the set $\{0,1\}^*$ of finite words over a two element alphabet.
- The root is the empty word ϵ.
- A node is some $w \in \{0,1\}^*$.
- Every $w \in \{0,1\}^*$ has two children: a left son $w0$ and a right son $w1$.

Definition

A Σ-labeled (full binary) tree is a function $t : \{0,1\}^* \to \Sigma$.
Trees(Σ) is the set of Σ-labeled full binary trees.

The full binary tree and a $\{a, b\}$-labeled tree





Obviously, we will take $\Sigma = 2^{Prop}$.
In the example, $Prop = \{p\}$, and say $a = \{p\}$,$b = \emptyset$.

The (propositional) Mu-calculus

The Mu-calculus

- invented by Dana Scott and Jaco de Bakker, and further developed by Dexter Kozen

- D. Kozen.
  Results on the propositional μ-calculus. Theoretical Computer Science, 27(3):333-354, 1983.
- A. Arnold and D. Niwinski.
  Rudiments of mu-calculus. North-Holland, 2001.
- E. A. Emerson and C. S. Jutla.
  Tree automata, mu-calculus and determinacy. In Proceedings 32nd Annual IEEE Symp. on Foundations of Computer Science, FOCS'91, San Jose, Puerto Rico, 1-4 Oct 1991, pages 368-377. IEEE Computer Society Press, Los Alamitos, California, 1991.

The Mu-calculus

Fundamental importance for several reasons, all related to its expressiveness:

- Uniform logical framework with great raw expressive power. It subsumes most modal and temporal logic of programs (e.g. LTL, CTL, CTL*).
- the Mu-calculus over binary trees coincide in expressive power with alternating tree automata.
- the semantic of the Mu-calculus is anchored in the Tarski-Knaster theorem, giving a means to do iteration-based model-checking in an efficient manner.

Smooth Introduction

- Consider the CTL formula $\mathbf{E}\,\mathbf{F}P$ (where P is some proposition): note that

$$\mathbf{E}\,\mathbf{F}P \equiv P \vee \mathbf{EX}\,\mathbf{E}\,\mathbf{F}P$$

  so that $\mathbf{E}\,\mathbf{F}P$ is a fixed-point.

- In fact, $\mathbf{E}\,\mathbf{F}P$ is the least fixed-point, e.g. the least such that $Z \equiv P \vee \mathbf{E}\,\mathbf{F}Z$.
- Not all modalities of e.g. CTL are needed as a "basis"

BYO modalities with fixed-point definitions

About lattices and fixed-points

See "Introduction to Lattices and Order", by B. A. Davey and H. A. Priestley. Cambridge 2002.

A lattice $(L, \leq)$ consists of a set L and a partial order $\leq$ such that any pair of elements has a greatest lower bound, the meet $\sqcap$, and a least upper bound, the join $\sqcup$, with the following properties:

(associative law)	$(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$
(commutative law)	$x \sqcup y = y \sqcup x$
(idempotency law)	$x \sqcup x = x$
(absorption law)	$x \sqcup (x \sqcap y) = x$

And similarly for $\sqcap$.

For example, given a set S, the powerset of S, $(\mathcal{P}(S), \subseteq)$, is a lattice.

Monotonic Functions

- $f : L \to L$ is monotonic (order preserving) if

$$\forall x, y \in L, x \leq y \;\Rightarrow\; f(x) \leq f(y)$$

- x is a fixed-point of f if $f(x) = x$
- Define f^0 is the identity function, and $f^{n+1} = f^n \circ f$.
- Note that f monotonic implies that f^n is monotonic. The identity function is monotonic and composing two monotonic functions gives a monotonic function.

Tarski-Knaster fixed-point Theorem

A lattice $(L \leq, \sqcup, \sqcap)$ is complete if for all $A \subseteq L$, $\sqcup A$ and $\sqcap A$ are defined; then there exist a minimum element $\bot = \sqcap L$ and a maximum element $\top = \sqcup L$.

This is the case for $(\mathcal{P}(S), \subseteq)$: given a set $A \subseteq \mathcal{P}(S)$ of subsets, $\sqcup A = \bigcup_{S' \in A} S'$ and $\sqcap A = \bigcap_{S' \in A} S'$.

EXERCISE What are $\top$ and $\bot$? □

Theorem

[Tar55] Let f be a monotonic function on $(L, \leq, \sqcup, \sqcap)$ a complete lattice. Let $A = \{y \mid f(y) \leq y\}$, then $x = \sqcap A$ is the least fixed-point of f.

(1) $f(x) \leq x$: $\forall y \in A$, $x \leq y$, therefore $f(x) \leq f(y) \leq y$. So $f(x) \leq \sqcap A = x$.

(2) $x \leq f(x)$: by monotonicity applied to (1), $f^2(x) \leq f(x)$ so $f(x) \in A$, and $x \leq f(x)$.

x is then a fixed-point, and because all fixed-points belong to A, x is the least. And similarly for the greatest fixed-point (with $A = \{y \mid f(y) \geq y\}$).

Another Characterization of fixed-points

(3) $\mu z.f(z)$, the least fixed-point of f, is equal to $\sqcup_i f^i(\emptyset)$, where i ranges over all ordinals of cardinality at most the state space L; when L is finite, $\mu z.f(z)$ is the union of the following ascending chain $\bot \subseteq f(\bot) \subseteq f^2(\bot)$...

(4) $\nu z.f(z) = \sqcap_i f^i(\top)$, where i ranges over all ordinals of cardinality at most the state space L; when L is finite, $\nu z.f(z)$ is the intersection of the following descending chain $\top \supseteq f(\top) \supseteq f^2(\top)$...

EXERCISE Show it. □

Syntax of the Mu-calculus

- An alphabet Σ, and the associate set of propositions $Prop = \{P_a\}_{a \in \Sigma}$.
- A infinite set of variables $Var = \{Z, Z', Y, \dots\}$.
- Formulas

$$\beta, \beta' \in L_\mu ::= P_a \mid Z \mid \neg\beta \mid \beta \wedge \beta' \mid \langle 0 \rangle \beta \mid \langle 1 \rangle \beta \mid \mu Z.\beta$$

  where $P_a \in Prop, Z \in Var$.
- Write $\langle\ \rangle\beta$ for $\langle 0 \rangle\beta \vee \langle 1 \rangle\beta$, and $[\ ]\beta$ for $\langle 0 \rangle\beta \wedge \langle 1 \rangle\beta$.
- β is a sentence if every occurrence of a variable in β are bounded by a μ operator.
- Write $\beta' \leq \beta$ when β' is a subformula of β.
- As $\mu Z.\beta$ is about a least fixed-point (see later for its semantics), we need to ensure its existence, hence the notion of well-formed formulas.

well-formed formulas

For every subformula $\mu Z.\beta$, Z appears only under the scope of an even number of $\neg$ symbols in β.

Semantics of well-formed formulas

- Fix a tree $t \in \mathit{Trees}(\Sigma)$
- Let $\mathit{val} : \mathit{Var} \to 2^{\{0,1\}^*}$ be a valuation of the variables. For every $N \subseteq \{0,1\}^*$, we write $\mathit{val}[N/Z]$ for val' defined as val except that $\mathit{val}'(Z) = N$
- Given a tree $t : \{0,1\}^* \to \Sigma$, $[\![\beta]\!]^t_{\mathit{val}} \subseteq \{0,1\}^*$ denotes a set of nodes.

$$
\begin{array}{lcl}
[\![Z]\!]^t_{val} & = & val(Z) \\
[\![P_a]\!]^t_{val} & = & t^{-1}(a) \\
[\![\neg\beta]\!]^t_{val} & = & \{0,1\}^* \setminus [\![\beta]\!]^t_{val} \\
[\![\beta \wedge \beta']\!]^t_{val} & = & [\![\beta]\!]^t_{val} \cap [\![\beta']\!]^t_{val} \\
[\![\langle 0 \rangle \beta]\!]^t_{val} & = & \{w \in \{0,1\}^* \mid w0 \in [\![\beta]\!]^t_{val}\} \\
[\![\langle 1 \rangle \beta]\!]^t_{val} & = & \{w \in \{0,1\}^* \mid w1 \in [\![\beta]\!]^t_{val}\} \\
[\![\mu Z.\beta]\!]^t_{val} & = & \bigcap\{N \in \mathcal{P}(\{0,1\}^*) \mid [\![\beta]\!]^t_{val[N/Z]} \subseteq N\}
\end{array}
$$

The meaning of $\mu Z.\beta$

- Recall

$$[\![\mu Z.\beta]\!]^t_{val} = \bigcap \{N \in \mathcal{P}(\{0,1\}^*) \mid [\![\beta]\!]^t_{val[N/Z]} \subseteq N\}$$

- $\mu Z.\beta$ denotes the least fixed-point of

$$f : 2^{\{0,1\}^*} \longrightarrow 2^{\{0,1\}^*}$$
$$f(N) = [\![\beta]\!]^t_{val[N/Z]}$$

  where f is monotonic, since β is well-formed.

  By [Tar55] (for the lattice $(2^{\{0,1\}^*}, \emptyset, \{0,1\}^*, \subseteq)$), f has a least fixed-point (and a greatest fixed-point) and this is precisely the value of $[\![\mu Z.\beta]\!]^t$.

- Let $\nu Z.\beta \stackrel{\text{def}}{=} \neg\mu Z.\neg\beta[\neg Z/Z]$. It is a greatest fixed-point.
- Notice that if β is sentence, then $[\![\mu Z.\beta]\!]^t_{val} = [\![\mu Z.\beta]\!]^t_{val'}$, for any val, val'; we write it $[\![\mu Z.\beta]\!]^t$.

Examples of formulas

We assume we have true and false in the syntax, with
$[\![\texttt{true}]\!]^t_{val} = \{0,1\}^*$ and $[\![\texttt{false}]\!]^t_{val} = \emptyset$.

- $\mu Z.Z \equiv \texttt{false}$
- $\nu Z.Z \equiv \texttt{true}$
- $\mu Z.P \equiv \nu Z.P \equiv P$

Examples of formulas: about **CTL**

- What is "$\mu Z.P_a \vee \langle\ \rangle Z$" ?
- It is equivalent to **E F**a, whereas $\nu Z.P_a \vee \langle\ \rangle Z \equiv \texttt{true}$

$$\begin{aligned} \mu Z.P_a \vee \langle\ \rangle Z &\equiv P_a \vee \langle\ \rangle(\mu Z.P_a \vee \langle\ \rangle Z) \\ &\equiv P_a \vee \langle\ \rangle(P_a \vee \langle\ \rangle(\mu Z.P_a \vee \langle\ \rangle Z)) \\ &\equiv P_a \vee \langle\ \rangle(P_a \vee \langle\ \rangle(P_a \vee \langle\ \rangle(\mu Z.P_a \vee \langle\ \rangle Z))) \\ &\equiv \ldots \end{aligned}$$

A node $w \in [\![\, \mu Z.P_a \vee \langle\ \rangle Z \,]\!]^t$ if either it is in $[\![\, P_a \,]\!]^t$ or it has a child who is either in $[\![\, P_a \,]\!]^t$ or who has a child who is in $[\![\, P_a \,]\!]^t$ or who has a child who ... The least set of nodes with this property is the set of nodes having a path eventually hitting a descendant node labeled by a. Hence the formula **EF** a.

- $\mathbf{A}\, a\, \mathbf{U}\, b \equiv \mu Z.P_b \vee P_a \wedge [\,]Z$, since

$$\mu Z.P_b \vee P_a \wedge [\,]Z \equiv P_b \vee P_a \wedge [\,](P_b \vee P_a \wedge [\,](P_b \vee P_a \wedge [\,](...)))$$

  whereas $\nu Z.P_b \vee P_a \wedge [\,]Z \equiv \mathbf{A}\, a\, \mathbf{W}\, b$, the weak until.

- $\mathbf{AG}\, a \equiv \nu Y.P_a \wedge [\,]Y$, since

$$\nu Y.P_a \wedge [\,]Y \equiv P_a \wedge [\,](P_a \wedge [\,](P_a \wedge [\,](...)))$$

  whereas $\mu Z.P_a \wedge [\,]Y \equiv \texttt{false}$

- $\mathbf{AG}\,\mathbf{EF}\, a \equiv \nu Y.(\mu Z.P_a \vee \langle\,\rangle Z) \wedge [\,]Y$
- $\mathbf{E}\,\mathbf{GF} b \equiv \nu Y.\mu Z.\langle\,\rangle(b \wedge Y \vee Z)$
- Intuitively, μ (resp. ν) refers to finite (resp. infinite) prefixes of computations.
- $\nu Z.P_a \wedge [\,][\,]Z$ is not expressible in CTL* [MP71, Wol83].

Positive normal form

We push negation innermost in the formulas
$\Rightarrow$ formulas in positive normal form

- Notice that $\neg\langle d \rangle \beta = \langle d \rangle \neg \beta$, for $d \in \{0, 1\}$.

  EXERCISE What if we do not assume states always have successors? (that is branches in the tree might be finite) □

Alternation Depth (± 1 in the literature)

Let $\beta \in L_\mu$ be in positive normal form.
We define $ad(\beta)$, the alternation depth of β inductively by:

- $ad(P_a) = ad(\neg P_a) = ad(Z) = 0$
- $ad(\beta \wedge \beta') = ad(\beta \vee \beta') = max\{ad(\beta), ad(\beta')\}$
- $ad(\langle d \rangle \beta) = ad(\beta)$, for $d \in \{0, 1\}$
- $ad(\mu Z.\beta) = max(\{1, ad(\beta)\} \cup \{ad(\nu Z'.\beta') + 1 \,|\, \nu Z'.\beta' \leq \beta, Z \in free(\nu Z'.\beta')\})$
- $ad(\nu Z.\beta) = max(\{1, ad(\beta)\} \cup \{ad(\mu Z'.\beta') + 1 \,|\, \mu Z'.\beta' \leq \beta, Z \in free(\mu Z'.\beta')\})$

Example: $ad(\nu Y.(\mu Z.P_a \vee \langle\,\rangle Z \wedge [\,]Y)) = 2$

Some important results

Write $L_\mu^k = \{\beta \in L_\mu \mid ad(\beta) \leq k\}$.

- CTL $\subseteq L_\mu^1$, and this is strict (recall $\nu Z.P_a \wedge [\,][\,]Z$ is not expressible in CTL*)
- $ad(\nu Y.\mu Z.(\langle\,\rangle Y \wedge P_a \vee Z)) = 2$, then $\mathbf{E\,GF}a$ is in L_μ^2.

Theorem

[Arn99, Bra96, Len96] The alternation hierarchy $L_\mu^0, L_\mu^1, L_\mu^2 \ldots$ is strict.

Theorem

[BGL07] The variable hierarchy of the μ-calculus is strict.

Model-checking and Satisfiability

- Write $t \models \beta$ whenever $\epsilon \in [\![\beta]\!]^t_{val}$.
- Let $L(\beta) \stackrel{\text{def}}{=} \{t \in Trees(\Sigma) \mid t \models \beta\}$
- The Model-checking Problem (Program Verification):
  Given regular tree t and a sentence $\beta \in L_\mu$, is it the case that $t \models \beta$?
- The Satisfiability Problem (Program Synthesis):
  Does there exist a tree t such that $t \models \beta$?
  Does there exist a regular tree? (The finite model property)

Definition (informal)

A tree is regular if it is obtained by unraveling a (finite) Kripke structure.

What next?

- Tree Automata to recognize certain trees:

$$\beta \in L_\mu \rightsquigarrow \mathcal{A}_\beta \text{ such that } L(\mathcal{A}_\beta) = \{t \in \mathit{Trees}(\Sigma) \mid t \models \beta\}$$

The Model-checking Problem $\rightsquigarrow$ The Membership Problem

The Satisfiability Problem $\rightsquigarrow$ The Emptiness Problem

- Games (two-player zero-sum) provide very powerful tools.

Automata on Infinite Objects

Automata on Infinite Objects

Automata with inputs like infinite words and infinite trees (and graphs).

- Automata on Infinite Trees [Rab69], [GH82, Mul84, EJ91], [GTW02, Chap. 8 and 9]
  - Acceptance conditions: Büchi, Muller, Rabin and Streett, Parity on every branch of the run of the automaton on its input.
  - Runs are trees, and accepting runs fulfill the acceptance condition.
  - We consider parity acceptance condition.
- Also ω-automata are automata on infinite words [Büc62, McN66], [Tho90], [GTW02, Chap. 1]
  - Acceptance conditions: Büchi, Muller, Rabin and Streett, Parity
  - Runs are paths, accepting runs fulfill the accepting condition.
  - All coincide with ω-regular languages ($L = \bigcup_i K_i R_i^\omega$) – deterministic Büchi are weaker.
  - Connection with Logic LTL: LTL corresponds to FOL as well as star-free ω-regular languages.

Non-deterministic Parity Tree Automata

- A (Σ-labeled full binary) tree t is input of an automaton.
- In a current node in the tree, the automaton has to decide which state to assume in each of the two child nodes.

Definition

A non-deterministic parity tree (NDPT) automaton is a structure $\mathcal{A} = (Q, \Sigma, q^0, \delta, c)$ where

- $Q(\ni q^0)$ is a finite set of states (q^0 the initial state)
- $\delta \subseteq Q \times \Sigma \times Q \times Q$ is the transition relation
- $c : Q \to \{0, \ldots, k\}$, $k \in \boldsymbol{N}$ is the coloring function which assigns the index values (colors) to each states of $\mathcal{A}$

Runs

Definition

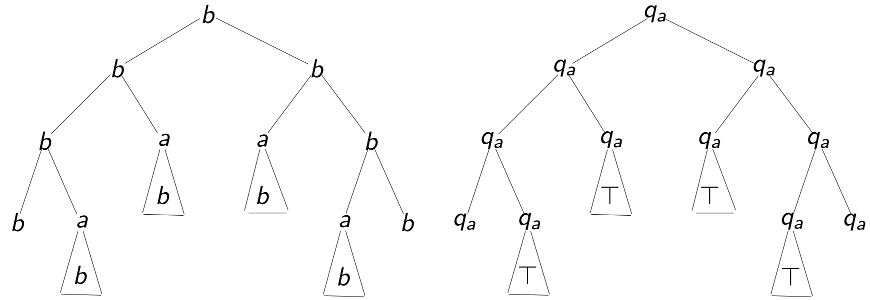
A run of $\mathcal{A} = (Q, \Sigma, q^0, \delta, c)$ on an input tree $t \in \mathit{Trees}(\Sigma)$ is a tree $\rho \in \mathit{Trees}(Q)$ satisfying

- $\rho(\epsilon) = q^0$, and
- for every node $w \in \{0,1\}^*$ of t (and its sons $w0$ and $w1$), we have

$$(\rho(w0), \rho(w1)) \in \delta(\rho(w), t(w))$$

Example

Consider the automaton with states q_a (initial) and $\top$, and the following transitions:



$$\begin{array}{lcllcl} \delta(q_a, a) & = & \{(\top, \top)\} & \delta(q_a, b) & = & \{(q_a, q_a)\} \\ \delta(\top, a) & = & \{(\top, \top)\} & \delta(\top, b) & = & \{(\top, \top)\} \end{array}$$

with $c(q_a) = 1$ and $c(\top) = 0$.

The parity acceptance condition

- Given a run ρ, for a branch γ in ρ write
  $\mathit{Inf}_c(\gamma) \stackrel{\text{def}}{=} \{j \in \{0, \ldots, k\} \mid c(\gamma(i)) = j \text{ for infinitely many } i\}$
- A run ρ is accepting (successful) iff for every branch $\gamma \in \{0,1\}^\omega$ of the tree ρ the parity acceptance condition is satisfied:

$$\mathit{min}\ \mathit{Inf}_c(\gamma) \text{ is even}$$

Example 1

- Let L_0 be the set of trees the branches of which all contain an a. This may be expressed in L_μ as $\mu Z.P_a \vee [\,]Z$ in L_μ.
- L_0 may be characterized by the following tree automaton

$$\begin{array}{rclrcl} \delta(q_a, a) & = & \{(\top, \top)\} & \delta(q_a, b) & = & \{(q_a, q_a)\} \\ \delta(\top, a) & = & \{(\top, \top)\} & \delta(\top, b) & = & \{(\top, \top)\} \end{array}$$

with q_a initial, $c(q_a) = 1$, and $c(\top) = 0$.

Example 2

Tree automata are nondeterministic, and cannot be determinized in general.

- Let $L_a^\infty \subseteq \mathit{Trees}(\{a, b\})$ be the set of trees having a branch with infinitely many a's.
- Consider the automaton with states $q_a, q_b, \top$ and transitions ($*$ stands for either a or b).

$$
\begin{array}{rcl}
\delta(q_*, a) & = & \{(q_a, \top), (\top, q_a)\} \\
\delta(q_*, b) & = & \{(q_b, \top), (\top, q_b)\} \\
\delta(\top, *) & = & \{(\top, \top)\}
\end{array}
$$

and coloring $c(q_b) = 1$ and $c(q_a) = c(\top) = 0$
(only 0 and 1 colors, this a Büchi condition)

Example 2 (Cont.)

$$\begin{array}{rcl} \delta(q_*, a) & = & \{(q_a, \top), (\top, q_a)\} \\ \delta(q_*, b) & = & \{(q_b, \top), (\top, q_b)\} \\ \delta(\top, *) & = & \{(\top, \top)\} \end{array}$$

with $c(q_b) = 1$ and $c(q_a) = c(\top) = 0$

- From state $\top$, $\mathcal{A}$ accepts any tree.
- Any run from q_a consists in a tree with of a single branch labeled with states q_a, q_b, whereas the rest of the run tree is labeled with $\top$. There are infinitely many states q_a on this branch iff there are infinitely many nodes labeled by a.

Acceptance

- A tree t is accepted by $\mathcal{A}$ iff there exists an accepting run of $\mathcal{A}$ on t.
- The tree language recognized by $\mathcal{A}$ is

$$L(\mathcal{A}) \stackrel{\text{def}}{=} \{t \mid t \text{ is accepted by } \mathcal{A}\}$$

Other Acceptance Conditions

- Büchi is specified by a set $F \subseteq Q$

$$Acc = \{\gamma \mid Inf(\gamma) \cap F \neq \emptyset\}$$

- Muller is specified by a set $\mathcal{F} \subseteq \mathcal{P}(Q)$,

$$Acc = \{\gamma \mid Inf(\gamma) \in \mathcal{F}\}$$

- Rabin is specified by a set $\{(R_1, G_1), \dots, (R_k, G_k)\}$ where $R_i, G_j \subseteq Q$,

$$Acc = \{\gamma \mid \forall i, Inf(\gamma) \cap R_i = \emptyset \text{ and } Inf(\gamma) \cap G_i \neq \emptyset\}$$

- Streett is specified by a set $\{(R_1, G_1), \dots, (R_k, G_k)\}$ where $R_i, G_j \subseteq Q$,

$$Acc = \{\gamma \mid \forall i, Inf(\gamma) \cap R_i = \emptyset \text{ or } Inf(\gamma) \cap G_i \neq \emptyset\}$$

Other Acceptance Conditions

- For the relationship between these conditions see [GTW02].
- Büchi is specified by a set $F \subseteq Q$ and this acceptabce condition for runs is:

$$Acc = \{\gamma \mid Inf(\gamma) \cap F \neq \emptyset\}$$

  Büchi tree automata are less expressive than the other acceptance conditions (which are equivalent) [Rab70]: for example, the complement of L_a^∞, that is finitely many a's on each branch, cannot be characterized by any Büchi tree automaton.

Regular Tree Languages and Properties

- A tree language $L \subseteq Trees(\Sigma)$ is regular iff there exists a parity tree automaton which recognizes L.
- Tree automata are closed under sum, projection, and complementation.
  - Tree automata cannot be determinized: $L_a^{\exists} \subseteq Trees(\{a, b\})$, the language of trees having one node labeled by a, is not recognizable by a deterministic tree automata (with any of the considered acceptance conditions).
  - The proof for complementation uses the determinization result for word automata. Difficult proof [GTW02, Chap. 8], [Rab70]
- We will solve the Membership Problem and the Emptiness Problem for (nondeterministic) automata by using Parity Games.

(Parity) Games

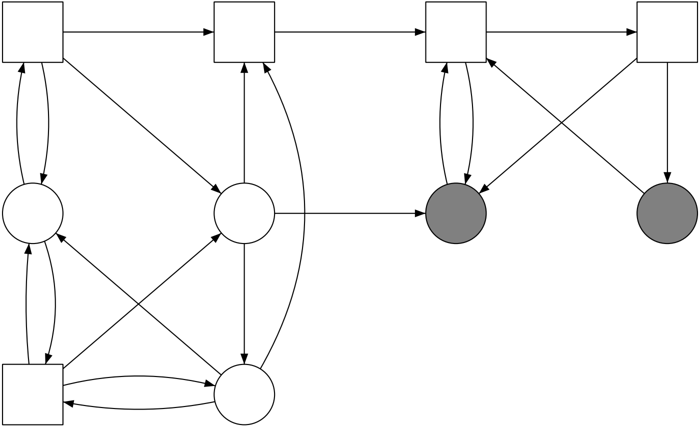
(Parity) Games

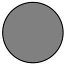
- Two-person games on directed graphs.
- How are they played?
- What is a strategy? What does it mean to say that a player wins the game?
- Determinacy, forgetful strategies, memoryless strategies

Arena

An arena (or a game graph) is

- $G = (V_0, V_1, E)$
- V_0 = Player 0 positions, and V_1 = Player 1 positions (partition of V)
- $E \subseteq V \times V$ is the edged-relation
- write $\sigma \in \{0, 1\}$ to designate a player, and $\overline{\sigma} = 1 - \sigma$



 color 0 and the rest is colored 1

Plays

- Formally, a play in the arena G is either
  - an infinite path $\pi = v_0 v_1 v_2 \ldots \in V^\omega$ with $v_{i+1} \in v_i E$ for all $i \in \omega$, or
  - a finite path $\pi = v_0 v_1 v_2 \ldots v_l \in V^+$ with $v_{i+1} \in v_i E$ for all $i < l$, but $v_l E = \emptyset$.

Games and Winning sets

- Let be G an arena and $Win \subseteq V^\omega$ be the winning condition
- Player 0 is declared the winner of a play π in the game $\mathcal{G}$ if
  - π is finite and $last(\pi) \in V_1$ and $last(\pi)E = \emptyset$, or
  - π is infinite and $\pi \in Win$.

Parity Winning Conditions

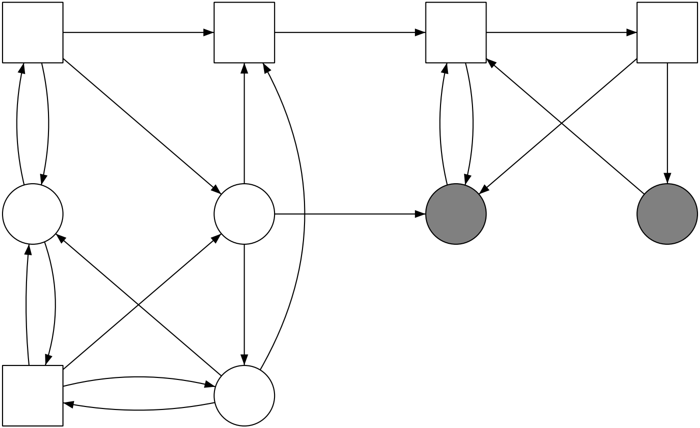
Informally, an infinite play is winning if the minimal color that occurs infinitely often even.

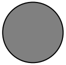
Formally

- We color vertices of the arena by $\chi : V \to C$ where C is a finite set of so-called colors; it extends to plays $\chi(\pi) = \chi(v_0)\chi(v_1)\chi(v_2)\ldots$.
- C is a finite set of integers called priorities
- Let $Inf_\chi(\pi)$ be the set of colors that occurs infinitely often in $\chi(\pi)$.

  Win is the set of infinite paths π such that $min(Inf_C(\pi))$ is even.

Example of a parity game

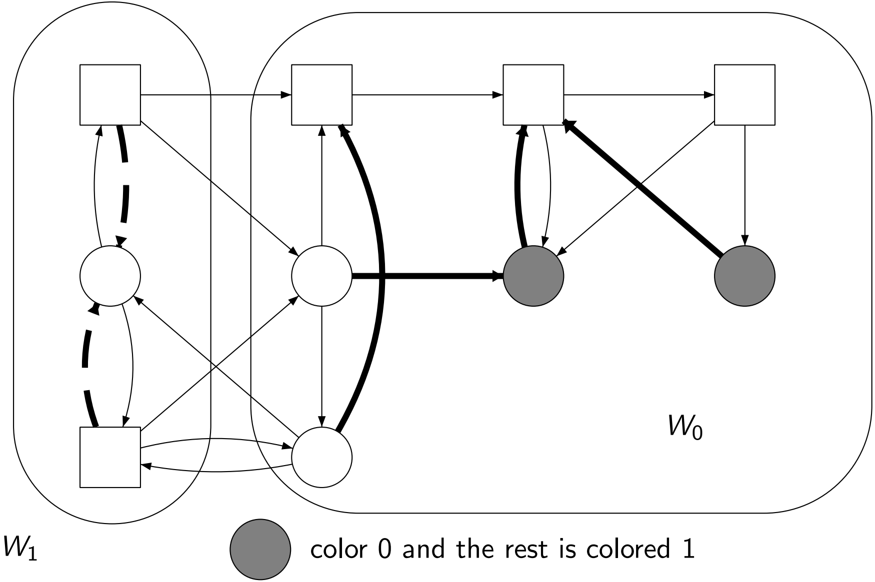


 color 0 and the rest is colored 1

Strategies and winning region

- A strategy for Player σ is a function $f_\sigma \colon V^* V_\sigma \to V$
- A prefix play $\pi = v_0 v_1 v_2 \ldots v_l$ is conform with f_σ if for every i with $0 \leq i < l$ and $v_i \in V_\sigma$ the function f_σ is defined and we have $v_{i+1} = f_\sigma(v_0 \ldots v_i)$.
- A play is conform with f_σ if each of its prefix is conform with f_σ.
- The winning region for Player σ is the set $W_\sigma(\mathcal{G}) \subseteq V$ of all vertices such that Player σ wins $(\mathcal{G}, v)$ (to be defined rigorously)

Example of Winning Regions



color 0 and the rest is colored 1

Determinacy of Parity Games

- A game $\mathcal{G} = ((V, E), Win)$ is determined when the sets $W_\sigma(\mathcal{G})$ and $W_{\overline{\sigma}}(\mathcal{G})$ form a partition of V.

Theorem

Every parity game is determined.

- A strategy f_σ is a positional (or memoryless) strategy whenever

$$f_\sigma(\pi v) = f_\sigma(\pi' v), \forall v \in V_\sigma$$

Theorem

[EJ91, Mos91] In every parity game, both players win memoryless.

See [GTW02, Chaps. 6 and 7]

Complexity Results

Theorem

$\textsc{Wins} =$
$\{(\mathcal{G}, v) \mid \mathcal{G}$ *a finite parity game and v a winning position of Player 0*$\}$
is in NP $\cap$ co-NP

1. Guess a memoryless strategy f of Player 0
2. Check whether f is memoryless winning strategy

[BJW02] proposed a reduction from parity games to safety games, that leads to an algorithm in $O(n(n/k)^{\lceil k/2 \rceil})$ ($k+1$ colors).

EXERCISE How would you solve a safety game? □

Back to Decision Problems for ND Tree Automata

The Membership Problem: $\mathcal{A} \rightsquigarrow \mathcal{G}_{\mathcal{A},t}$

1. Given a tree t and an NDPT automaton $\mathcal{A}$, we build a parity game $(\mathcal{G}_{\mathcal{A},t}, v_I)$ s.t. v_I is in $W_0(\mathcal{G}_{\mathcal{A},t})$ iff $t \in L(\mathcal{A})$.

   Moreover, if t is regular (i.e. represented by a finite KS $(\mathcal{S}, s)$), we can build a finite game.

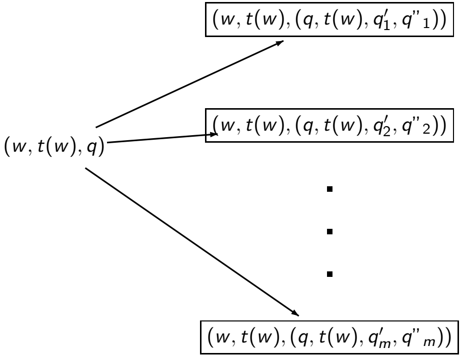
The Emptiness Problem: $\mathcal{A} \rightsquigarrow \mathcal{A}' \rightsquigarrow \mathcal{G}_{\mathcal{A}'}$

1. For each parity automaton $\mathcal{A}$, we build an Input Free automaton $\mathcal{A}'$ such that $L(\mathcal{A}) \neq \emptyset$ iff $\mathcal{A}'$ admits a successful run.
2. From $\mathcal{A}'$ we build a parity game $\mathcal{G}_{\mathcal{A}'}$ such that (winning) strategies of Player 0 and (successful) runs of $\mathcal{A}'$ correspond.

Both problem reduce to solving parity games!

The Membership Problem: The Game Graph $\mathcal{G}_{\mathcal{A},t}$

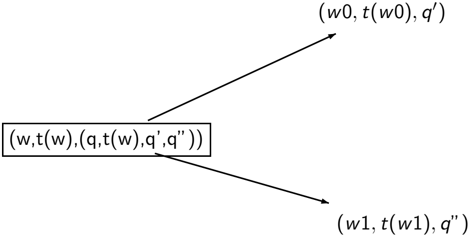
0-positions are of the form $(w, t(w), q)$.
Moves from $(w, t(w),)$, with
$\delta(q, t(w)) = \{(q'_1, q"_1), (q'_2, q"_2), \dots (q'_m, q"_m)\}$ are:



Player 0 chooses the transition $(q, t(w), q', q")$ from q for input $t(w)$

The Game Graph $\mathcal{G}_{\mathcal{A},t}$

1-positions are of the form $(w, t(w), (q, t(w), q', q"))$.
2 possible moves from $(w, t(w), (q, t(w), q', q"))$:



Player 1 chooses the branch in the run (left q', or right $q"$)

The Game Graph $\mathcal{G}_{\mathcal{A},t}$

$\mathcal{A} = (Q, \Sigma, q^0, \delta, c)$

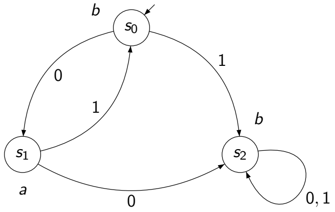
- V_0 = set of triples $(w, t(w), q) \in \{0,1\}^* \times \Sigma \times Q$
- V_1 = set of triples $(w, t(w), \tau) \in \{0,1\}^* \times \Sigma \times \delta$
- Moves ...
- Initial position in $(\epsilon, t(\epsilon), q^0) \in V_0$
- Priorities:

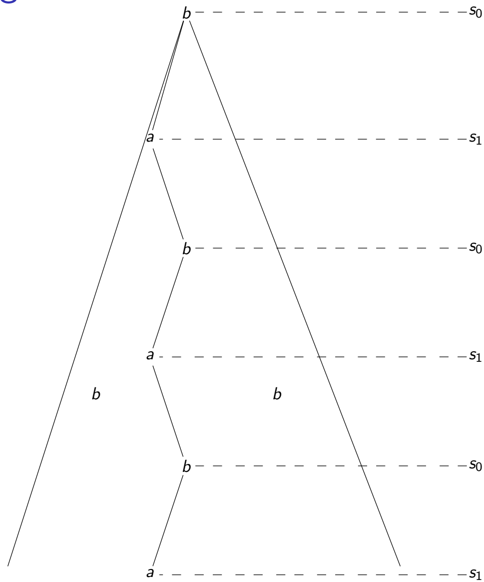
$$\chi((w, t(w), q)) = c(q)$$
$$\chi((w, t(w), (q, t(w), q', q"))) = c(q)$$

The Game Graph $\mathcal{G}_{\mathcal{A},t}$

- V_0: $(w, t(w), \text{state } q)$
- V_1: $(w, t(w), \text{transition } (q, t(w), q', q"))$
- Moves from V_0: from $(w, t(w), q)$, Player 0 can move to $(w, t(w), (q, t(w), q', q"))$, for every $(q, t(w), q', q") \in \delta$
- Moves from V_0: from $(w, t(w), (q, t(w), q', q"))$, Player 1 can moves to $(w0, t(w0), q')$ or to $(w1, t(w1), q")$.

The Finite Game with a Regular Tree



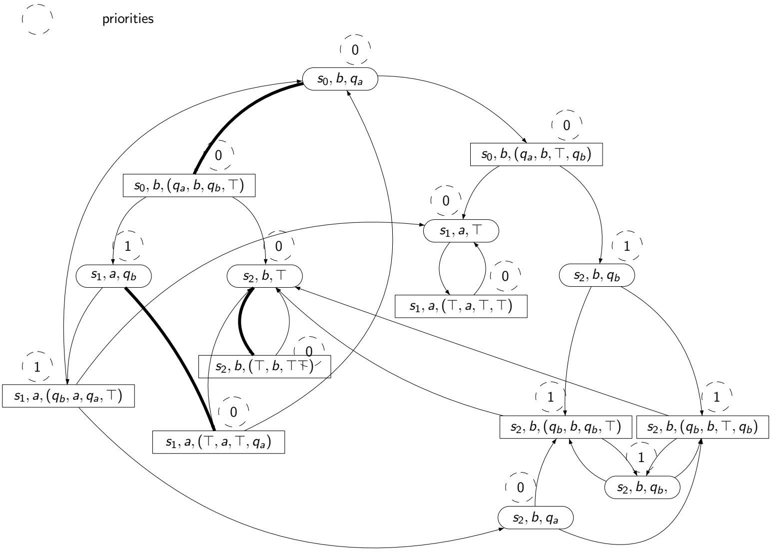


With the automaton:

$$\delta(q_*, a) = \{(q_a, \top), (\top, q_a)\}$$
$$\delta(q_*, b) = \{(q_b, \top), (\top, q_b)\}$$
$$\delta(\top, *) = \{(\top, \top)\}$$

$$c(q_a) = c(\top) = 0$$
$$c(q_b) = 1$$

Example of $\mathcal{G}_{\mathcal{A},t}$



The Emptiness Problem of NDTA

We need the notion of input-free automata.

- An input-free (IF) automaton is $\mathcal{A}' = (Q, \delta, q_I, Acc)$ where $\delta \subseteq Q \times Q \times Q$.

Lemma

For each parity automaton $\mathcal{A}$ there exists an IF automaton $\mathcal{A}'$ such that $L(\mathcal{A}) \neq \emptyset$ iff $\mathcal{A}'$ admits a successful run.

- $\mathcal{A} = (Q, \Sigma, q^0, \delta, c)$ and define $\mathcal{A}' = (Q \times \Sigma, \{q_I\} \times \Sigma, \delta', c')$. $\mathcal{A}'$ will guess non-deterministically the second component of its states, i.e. the labeling of a model. Formally,
  - for each $(q, a, q', q'') \in \delta$, we generate $((q, a), (q', x), (q'', y)) \in \delta'$, if $(q', x, p, p'), (q'', y, r, r') \in \delta$ for some $p, p', q, q' \in Q$
  - $c'(q, a) = c(q)$

Example IF Automaton

$$
\begin{array}{lcl}
\mathcal{A} & \rightsquigarrow & \mathcal{B} \\
(q_a, a, q_a, \top), (q_a, a, \top, q_a) & \rightsquigarrow & ((q_a, a), (q_a, a), (\top, a)), ((q_a, a), (\top, a), (q_a, a)) \\
 & & ((q_a, a), (q_a, b), (\top, a)), ((q_a, a), (\top, b), (q_a, a)) \\
 & & ((q_a, a), (q_a, a), (\top, b)), ((q_a, a), (\top, a), (q_a, b)) \\
 & & ((q_a, a), (q_a, b), (\top, b)), ((q_a, a), (\top, b), (q_a, b))
\end{array}
$$

$$
\begin{array}{lcl}
(q_a, b, q_b, \top), (q_a, b, \top, q_b) & \rightsquigarrow & ((q_a, b), (q_b, a), (\top, a)), ((q_a, a), (\top, a), (q_b, a)) \\
 & & ((q_a, b), (q_b, b), (\top, a)), ((q_a, a), (\top, b), (q_b, a)) \\
 & & ((q_a, b), (q_b, a), (\top, b)), ((q_a, a), (\top, a), (q_b, b)) \\
 & & ((q_a, b), (q_b, b), (\top, b)), ((q_a, a), (\top, b), (q_b, b))
\end{array}
$$

$(q_b, a, q_a, \top), (q_b, a, \top, q_a) \rightsquigarrow \dots$ $\quad$ $(q_b, b, q_b, \top), (q_b, b, \top, q_b) \rightsquigarrow \dots$

$$
\begin{array}{lcl}
(\top, a, \top, \top) & \rightsquigarrow & ((\top, a), (\top, a), (\top, a)) \\
 & & ((\top, a), (\top, b), (\top, a)) \\
 & & ((\top, a), (\top, a), (\top, b)) \\
 & & ((\top, a), (\top, b), (\top, b))
\end{array}
$$

$(\top, b, \top, \top) \rightsquigarrow \dots$

$$c'((q_a, *)) = c(q_a) = 0, c'((\top, *)) = c(\top) = 0, c'((q_b, *)) = c(q_b) = 1$$

From IF Automata to Parity Games

$\mathcal{A}$ an IF automaton $\rightsquigarrow$ a parity game $\mathcal{G}_{\mathcal{A}}$

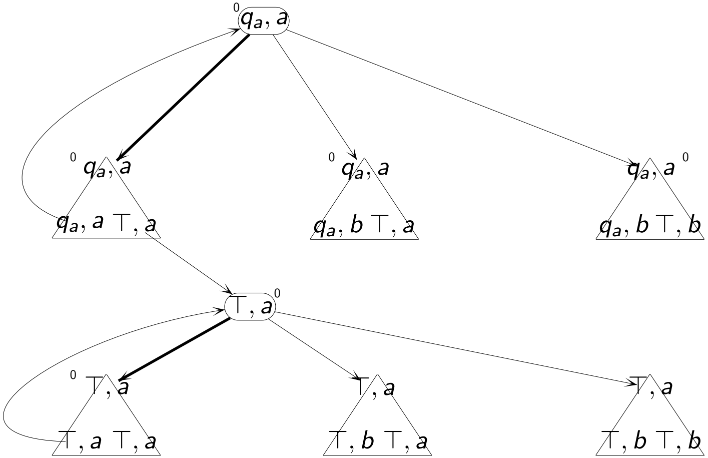
- Positions $V_0 = Q$ and $V_1 = \delta$
- Moves for all $(q, q', q'') \in \delta$
  - $(q, (q, q', q"\,)) \in E$
  - $((q, q', q"\,), q'), ((q, q', q"\,), q"\,) \in E$
- Priorities $\chi(q) = c(q) = \chi((q, q', q"\,))$

Lemma

(Winning) Strategies of Player 0 and (successful) runs of $\mathcal{A}$ correspond.

Notice that $\mathcal{G}_{\mathcal{A}}$ has a finite number of positions.

Example of $\mathcal{G}_{\mathcal{A}}$



Decidability of Emptiness for NDPT Automata

Theorem

For parity tree automata it is decidable whether their recognized language is empty or not.

$\mathcal{A} \rightsquigarrow \mathcal{A}' \rightsquigarrow \mathcal{G}_{\mathcal{A}'}$, and combined previous results.

Finite Model Property

Corollary

If $L(\mathcal{A}) \neq \emptyset$ then $L(\mathcal{A})$ contains a regular tree.

Use the memoryless winning strategy in $\mathcal{G}_{\mathcal{A}'}$.

Formally, take $\mathcal{A}$ and its corresponding IF automatan $\mathcal{A}'$. Assume a successful run of $\mathcal{A}'$ and a memoryless strategy f for Player 0 in $\mathcal{G}_{\mathcal{A}'}$ from some position (q_I, a).
The subgraph $\mathcal{G}_{\mathcal{A}'_f}$ induces a deteministic IF automaton $\mathcal{A}$" (without acc): extract the transitions out of $\mathcal{G}_{\mathcal{A}_f}$ from positions in V_1. $\mathcal{A}$" is a subautomaton of $\mathcal{A}'$.
$\mathcal{A}$" generates a regular tree t in the second component of its states. Now, $t \in L(\mathcal{A})$ because $\mathcal{A}'$ behaves like $\mathcal{A}$.

Complexity Issues

Corollary

The Emptiness Problem for NDPT automata is in NP $\cap$ co-NP.

Notice that the size of $\mathcal{G}_{\mathcal{A}'}$ is polynomial in the size of $\mathcal{A}$ (see [GTW02, p. 150, Chap. 8]).

Remark

The universality problem is EXPTIME-complete (already for finite trees).

What we have seen

- Binary trees as a simplified setting to represent system's executions.
- Propositional μ-calculus that subsumes all branching-time temporal logics (LTL, CTL, CTL*, PDL, ...).
- Non-determinsitic tree automata (NDTA) to recognize regular tree languages.
- (Parity) games as abstract mathematical tools to, *e.g.* check emptiness and membership problems for NDTA.
  $\Rightarrow$ The emptiness problem for NDTA is in $NP \cap \text{co-}NP$.
  $\Rightarrow$ Memoryless strategies deliver regular objects.

In particular, NDTA have the finite model property.

What we have not seen

- A generalization of NDTA as Alternating Tree Automata (ATA) and the Simulation Theorem [MS95] that states an exponential time procedure to convert ATA into NDTA.
  $\Rightarrow$ ATA have the finite model property.
  $\Rightarrow$ Checking emptiness of ATA is in *EXPTIME*(in fact, complete).
  BUT checking membership for ATA is in $NP \cap$ co-NP.
- The two-way translation μ-calculus formulas $\leftrightarrow$ ATA.
  $\Rightarrow$ The μ-calculus has the finite model property.
  $\Rightarrow$ Satisfiability of μ-calculus formulas is in *EXPTIME*.
  $\Rightarrow$ Model-checking μ-calculus formulas is in $NP \cap$ co-NP.

A. Arnold.
The mu-calculus alternation-depth hierarchy is strict on binary trees.
Research Report 1215-99, LaBRI, Université Bordeaux I, 1999.

Dietmar Berwanger, Erich Grädel, and Giacomo Lenzi.
The variable hierarchy of the μ-calculus is strict.
Theory Comput. Syst., 40(4):437–466, 2007.

J. Bernet, D. Janin, and I. Walukiewicz.
Permissive strategies: from parity games to safety games.
Theoretical informatics and applications, 36:251–275, 2002.

J. C. Bradfield.
The modal mu-calculus alternation hierarchy is strict.
In *Proc. Concurrency Theory, 7th International Conference, CONCUR'96, Pisa, Italy, LNCS1119*, pages 233–246, 1996.

J. R. Büchi.
On a decision method in restricted second order arithmetic.

In *Proc. 1960 Int. Congr. Logic, Methodology and Philosophy of Science, London*, pages 1–11. Stanford Univ. Press, 1962.

E. A. Emerson and J. Y. Halpern.
"Sometimes" and "Not Never" revisited: On branching versus linear time.
In *Proc. 10th ACM Symp. Principles of Programming Languages, Austin, Texas*, pages 127–140, January 1983.

E. A. Emerson and C. S. Jutla.
Tree automata, mu-calculus and determinacy.
In *Proceedings 32nd Annual IEEE Symp. on Foundations of Computer Science, FOCS'91, San Jose, Puerto Rico, 1–4 Oct 1991*, pages 368–377. IEEE Computer Society Press, Los Alamitos, California, 1991.

E. A. Emerson.
Temporal and modal logic.
In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 16, pages 995–1072. Elsevier Science Publishers, 1990.

Y. Gurevich and L. Harrington.
Trees, automata, and games.
In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 60–65, San Francisco, California, May 1982.

E. Grädel, W. Thomas, and T. Wilke, editors.
Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], volume 2500 of *Lecture Notes in Computer Science*.
Springer, 2002.

Giacomo Lenzi.
A hierarchy theorem for the μ-calculus.
In F. Meyer auf der Heide and B. Monien, editors, *Proceedings 23rd Int. Coll. on Automata, Languages and Programming, ICALP'96, Paderborn, Germany, 8–12 July 1996*, volume 1099 of *Lecture Notes in Computer Science*, pages 87–97. Springer-Verlag, Berlin, 1996.

R. McNaughton.

Testing and generating infinite sequences by a finite automaton.
Information and Control, 9:521–530, 1966.

A. W. Mostowski.
Games with forbidden positions.
Research Report 78, Univ. of Gdansk, 1991.

R. McNaughton and S. Papert.
Counter-Free Automata.
MIT Press, Cambridge, MA, 1971.

David E. Muller and Paul E. Schupp.
Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra.
Theoretical Computer Science, 141(1–2):69–107, 17 April 1995.

D. Muller.
Alternating automata on infinite objects, determinacy and Rabin's theorem.

In *Automata on Infinite Words, Le Mont Doré, LNCS 192*, pages 100–107. Springer-Verlag, May 1984.

M. O. Rabin.
Decidability of second-order theories and automata on infinite trees.
Trans. Amer. Math. Soc., 141:1–35, 1969.

M. O. Rabin.
Weakly definable relations and special automata.
In *Symp. Math. Logic and Foundations of Set Theory*, pages 1–23, 1970.

A. Tarski.
A lattice-theoretical fixpoint theorem and its applications.
Pacific J. Math., 5:285–309, 1955.

W. Thomas.
Automata on infinite objects.
In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 4, pages 133–191. Elsevier Science Publishers, 1990.

P. Wolper.
Temporal logic can be more expressive.
Information and Control, 56:72–99, 1983.