Logic, Automata, and Games

Sophie Pinchinat

IRISA, university of Rennes 1, France

Logic Summer School 2009

Model-Checking

- The Model-checking Problem: A system *Sys* and a specification *Spec*, decide whether *Sys* satisfies *Spec*.
- Example: Mutual exclusion protocol

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

- A state is a bit vector

  (line no. of process 1,line no. of process 2, value of turn)

  Start from `(00000)`.

- *Spec* = "a state `(1010b)` is never reached", and "always when a state `(01bcd)` is reached, then later a state `(10b'c'd')` is reached" (and similarly for Process 2, i.e. states `(bc01d)` and `(b'c'10d')`)

The Formal Approach

- Models of systems are Kripke Structures
- Specifications languages are Temporal Logics

Kripke Structures

Assume given $Prop = p_1, \ldots, p_n$ a set of atomic propositions (properties).

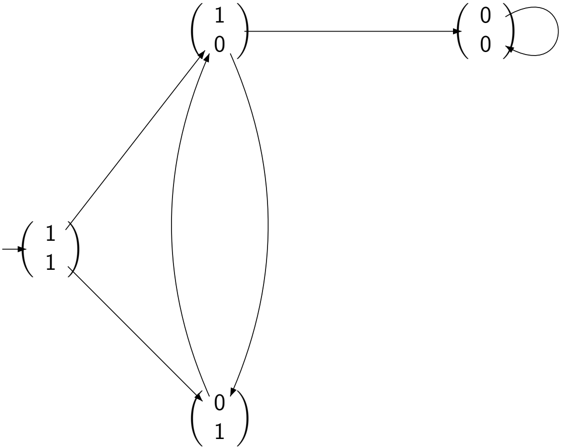
- A Kripke Structure over $Prop$ is $\mathcal{S} = (S, R, \lambda)$
  - S is a set of states (worlds)
  - $R \subseteq S \times S$ is a transition relation
  - $\lambda : S \rightarrow 2^{Prop}$ associates those p_i which are assumed true in s. Write $\lambda(s)$ as a bit vector $(b_1, \ldots, b_n)$ with $b_i = 1$ iff $p_i \in \lambda(s)$
- A rooted Kripke Structure is a pair $(\mathcal{S}, s)$ where s is a distinguished state, called the initial state.

Mutual Exclusion Protocol

- Use p_1, p_2 for "being in wait instruction before critical section of Process 1, or Process 2 respectively"
- Use p_3, p_4 for "being in critical section of Process 1, or Process 2 respectively"
- Example of label function $\lambda(01101) = \{p_1, p_4\}$ (encoded by (1001))
- The relation R is as defined by the transitions of the protocol.

A Toy System

Over two propositions p_1, p_2

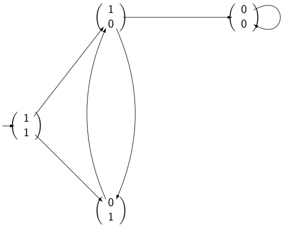


Paths and Words

Let $\mathcal{S} = (S, R, \lambda)$ be Kripke Structure over *Prop*

- 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 $(\in (\mathbb{B}^n)^\omega)$ is $\lambda(s_0), \lambda(s_1), \lambda(s_2), \ldots$.

$\alpha = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ldots$ in



- If $\alpha = \alpha(0)\alpha(1)\ldots \in (\mathbb{B}^n)^\omega$,
  1. α^i stands for $\alpha(i)\alpha(i+1)\ldots$ So $\alpha = \alpha^0$.
  2. $(\alpha(i))_j$ is the jth component of $\alpha(i)$

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:

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

wher $p \in Prop$. Other Boolean connectives $\texttt{true}, \texttt{false}, \varphi_1 \wedge \varphi_2$, $\varphi_1 \Rightarrow \varphi_2$, and $\varphi_1 \Leftrightarrow \varphi_2$ are defined via the usual abbreviations.

Semantics of LTL

Define $\alpha^i \models \varphi$ by induction over φ (where α is a word):

- $\alpha^i \models p_j$ iff $(\alpha(i))_j = 1$
- $\alpha^i \models \varphi_1 \vee \varphi_2$ 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 $k = i, \ldots, j-1$, $\alpha^k \models \varphi_1$

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

Examples

Formulas over p_1 and p_2:

1. $\alpha \models \mathbf{GF}p_1$ iff "in α, infinitely often 1 appears in the first component".
2. $\alpha \models \mathbf{X}\,\mathbf{X}\,(p_2 \;\Rightarrow\; \mathbf{F}p_1)$ iff "if the second component of $\alpha(2)$ is 1, so will be the first component of $\alpha(j)$ for some $j \geq 2$".
3. $\alpha \models \mathbf{F}(p_1 \wedge \mathbf{X}\,(\neg p_2\,\mathbf{U}\,p_1))$ iff "α has two letters $\begin{pmatrix} 1 \\ \star \end{pmatrix}$ such that in between only letters $\begin{pmatrix} \star \\ 0 \end{pmatrix}$ occur".

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:

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

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

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

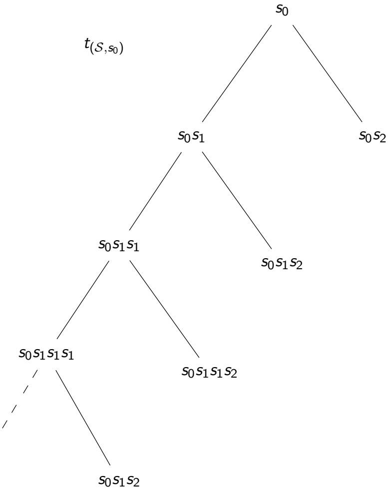
Let $\mathbf{A}\,\psi \stackrel{\text{def}}{=} \neg\mathbf{E}\,\neg\psi$

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

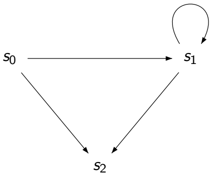
Interpretation over Trees

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

$t_{(\mathcal{S},s_0)}$



$\mathcal{S}$



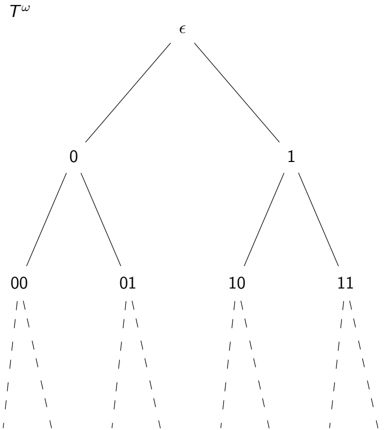
Σ-Labeled Full Binary Trees

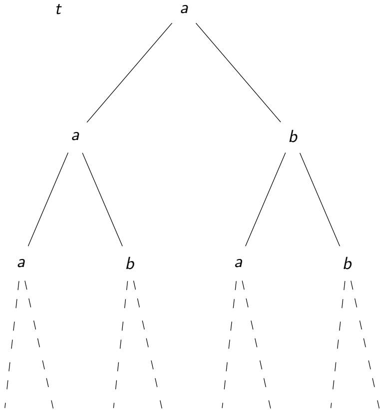
For simplicity we assume that states have exactly two successors $\Rightarrow$ we consider (only) binary trees

- The full binary tree T^ω is the set $\{0,1\}^*$ of finite words over a two element alphabet.
- The root is the empty word ϵ
- A node $w \in \{0,1\}^*$ has left son $w0$ and right son $w1$.
- A Σ-labeled full binary tree is a function $t : \{0,1\}^* \to \Sigma$
- $\mathit{Trees}(\Sigma)$ is the set of Σ-labeled full binary trees.

If the formulas are over the set Prop of propositions, then take $\Sigma = 2^{\mathit{Prop}}$ (or equivalently $\mathbb{B}^n$)

Example





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-basmodel-checking in an efficient manner.

Smooth Introduction

- Consider the CTL formula $\mathbf{E}\,\mathbf{F}p$: 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 fix-point.

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

BYO modalities with fix-point definitions

About Fix-points

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

Tarski-Knaster Fix-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 $\sqcap A = \bigcap_{S' \in A} S'$.

Theorem

[Tar55] Let f be a montonic function on $(L, \leq, \sqcup, \sqcap)$ a complete lattice. Let $A = \{y \mid f(y) \leq y\}$, and let $x = \sqcap A$ is the least fix-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 fix-point, and because all fix-point belong to A, x is the least.

And similarly for the greatest fix-point (with $A = \{y \mid f(y) \geq y\}$).

Another Characterization of Fix-points

(3) $\mu z.f(z)$, the least fix-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 $\perp \subseteq f(\perp) \subseteq f2(\perp)$...

(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 $\top \supseteq f(\top) \supseteq f^2(\top)$...

Syntax of the Mu-calculus

- Alphabet Σ and Propositions $Prop = \{P_a\}_{a \in \Sigma}$
- 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 $Z \in Var$.
- Well-formed formulas: for every formula $\mu Z.\beta$, Z appears only under the scope of an even number of $\neg$ symbols in β.
- β is a sentence if all variables in β are bounded by a μ operator.
- Write $\beta' \leq \beta$ when β' is a subformula of β.

Semantics

- Assume given a tree $t \in \mathit{Trees}(\Sigma)$ and a valuation $\mathit{val} : \mathit{Var} \to 2^{\{0,1\}^*}$ 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 labeled tree $t : \{0,1\}^* \to \Sigma$, we define $[\![\beta]\!]^t_{\mathit{val}} \subseteq \{0,1\}^*$ by:

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

The meaning of $\mu Z.\beta$

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

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

  By the assumption on "positive" occurrences of Z in β, we can show that f is monotonic (see the literature).
  Henceforth, since $(2^{\{0,1\}^*}, \emptyset, \{0,1\}^*, \subseteq)$ is a complete lattice, by [Tar55], the least fix-point (and the greatest fix-point) exists.

- Let $\nu Z.\beta \stackrel{\text{def}}{=} \neg\mu Z.\neg\beta[\neg Z/Z]$. It is a greatest fix-point.

Examples of formulas

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

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

Examples of formulas (cont.)

Write $\langle\ \rangle\beta$ for $\langle 0\rangle\beta \vee \langle 1\rangle\beta$, and $[\ ]\beta$ for $\langle 0\rangle\beta \wedge \langle 1\rangle\beta$.

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

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

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

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

  whereas $\mu Z.P_a \wedge [\,]Y \equiv \texttt{false}$
- $\mathbf{E}\ \overset{\infty}{\mathbf{F}}\ b \equiv \nu Y.\mu Z.\langle\,\rangle(b \wedge Y \vee Z)$
- Intuitively, μ (resp. ν) refers to finite (resp. infinite) prefixes of computations.
- $\nu Z.P_a \wedge [\,][\,]Z$ is not expressible in CTL*

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 $d \in \{0, 1\}$.

Alternation Depth

Let $\beta \in L_\mu$ be in postive 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 $d \in \{0, 1\}$
- $ad(\mu Z.\beta) = max(\{1, ad(\beta)\} \cup \{ad(\nu Z'.\beta') + 1 \,|\, \nu Z'.\beta' \leq \beta, Z \in free(\nu Z'.\beta')\})$
- $ad(\nu Z.\beta) = max(\{1, ad(\beta)\} \cup \{ad(\mu Z'.\beta') + 1 \,|\, \mu Z'.\beta' \leq \beta, Z \in free(\mu Z'.\beta')\})$

- Write $L_\mu^k = \{\beta \in L_\mu \mid ad(\beta) \leq k\}$.
  The hierarchy $L_\mu^0, L_\mu^1, L_\mu^2 \ldots$ is strict [Bra96, Len96].
- $ad(\mathbf{AG\,EF}\,a) = ?$: $\mathbf{AG\,EF}\,a \equiv \nu Y.(\mu Z.P_a \vee \langle\ \rangle Z) \wedge [\ ]Y$

$$\nu Y.\overbrace{(\mu Z.P_a \vee \langle\ \rangle Z)}^{0} \wedge [\ ]Y$$

and Y does not appear free in $\mu Z.P_a \vee \langle\ \rangle Z$

  hence $ad(\nu Y.(\mu Z.P_a \vee \langle\ \rangle Z) \wedge [\ ]Y) = 1$.
- CTL $\subseteq L_\mu^1$ the alternation free mu-calculus, 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 $\mathbf{E}\overset{\infty}{\mathbf{F}}\,a$ is in L_μ^2.

Model-checking and Satisfiabilty

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

Model-checking = Program Verification
Satisfiability = Program Synthesis

Next lectures

- Tree Automata: devices which recognize models of formulas:

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

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

The Satisfiability Problem $\rightsquigarrow$ The Emptiness Problem

- Games provide very powerful tools

Automata on Infinite Objects

Automata on Infinite Objects

We refer to [Tho90] and [GTW02, Chap. 1].

- Automata on (meaning with inputs as) words, trees, and graphs.
- ω-automata are automata on infinite words
  - Acceptance conditions: Büchi, Muller, Rabin and Streett, Parity
  - All coincide with ω-regular languages ($L = \bigcup_i K_i R_i^\omega$)
  - Connection with Logic LTL: LTL corresponds to star-free ω-regular languages
- Connection with Games

Automata on Infinite Trees

- Acceptance conditions: Büchi, Muller, Rabin and Streett, Parity on each branch of the run of the automaton on its input. We will focuse on parity acceptance condition.

Non-deterministic Parity Tree Automata

- A (Σ-labeled full binary) tree t is input to an automaton.
- In a current node in the tree, the automaton has to decide which state to assume in each of the two successor nodes.
- $\mathcal{A} = (Q, \Sigma, q^0, \delta, c)$ 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
  - $c : Q \to \{0, \ldots, k\}$, $k \in \mathbb{N}$ is the coloring function which assigns the index values (colors) to each states of $\mathcal{A}$

Runs

- 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

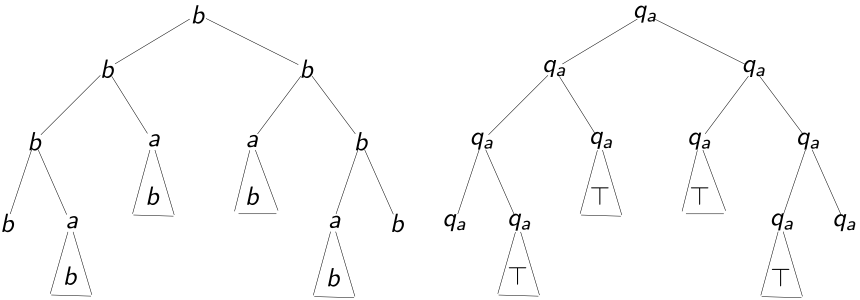
$$(\rho(w0), \rho(w1)) \in \delta(\rho(w), t(w))$$

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

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

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

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



Acceptance

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

$$min\ Inf_c(\gamma) \text{ is even}$$

- 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

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

Example 1

- Let L_0 be the set of trees whose paths have an a ($\mu Z.P_a \vee [\,]Z$ in L_μ)
- It is characterized by

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

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 path 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.)

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

- From state $\top$, $\mathcal{A}$ accepts any tree.
- Any run from q_a consists in a tree with of a single path 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 path iff there are infinitely many vertices labeled by a.

Other Acceptance Conditions

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

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

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

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

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

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

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

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

- For the relationship between these conditions see [GTW02].
- In the following, when the definition and results apply to any acceptance conditions presented so far (including parity condition), we simply denote by Acc this condition.
- Büchi tree automata are less expressive than the others (which are equivalent) [Rab70]: the complement of L_a^∞ (finitely many a's on each branch) cannot be recognized by any Büchi tree automaton.

Regular Tree Languages and Properties

- A tree language $L \subseteq Trees(\Sigma)$ is regular iff there exists a parity (Muller, Rabin, Streett) 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 see how to solve the Membership Problem and the Emptiness Problem for (nondeterministic) automata: we use 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 $\overline{\sigma} = 1 - \sigma$

Plays

- A token is placed on some initial vertex $v \in V$
- When v is a σ-vertex, the Player σ moves the token from v to some successor position $v' \in vE$.
- This is repeated infinitely often or until a vertex $\bar{v}$ without successor is reached ($\bar{v}E = \emptyset$)
- 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

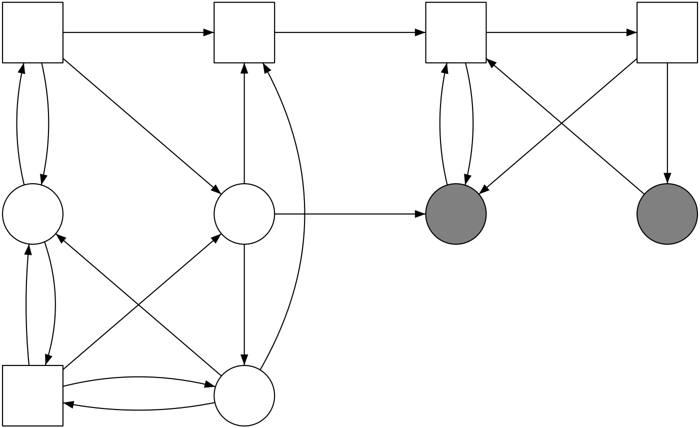
- Let be G an arena and $Win \subseteq V^\omega$ be the winning condition
- The pair $\mathcal{G} = (G, Win)$ is called a game
- Player 0 is declared the winner of a play π in the game $\mathcal{G}$ if
  - π is finite and $last(\pi) \in V_1$ and $last(\pi)E = \emptyset$, or
  - π is infinite and $\pi \in Win$.
- Player 1 wins π if Player 0 does not win π.
- Initialized game $(\mathcal{G}, v_I)$.

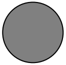
Parity Winning Conditions

- We color vertices of the arena by $\chi : V \rightarrow C$ 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

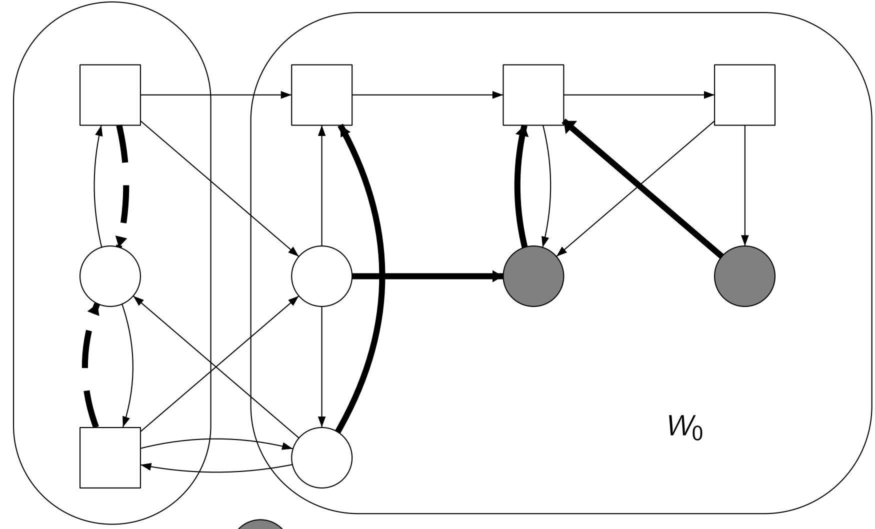
Strategies

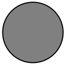
- A strategy for Player σ is a function $f_\sigma \colon V^* V_\sigma \to V$
- A prefix play $\pi = v_0 v_1 v_2 \dots v_l$ is conform with f_σ if for every i with $0 \leq i < l$ and $v_i \in V_\sigma$ the function f_σ is defined and we have $v_{i+1} = f_\sigma(v_0 \dots v_i)$.
- A play is conform with f_σ if each of its prefix is conform with f_σ.
- f_σ is a strategy for Player σ on $U \subseteq V$ if it is defined for every prefix of a play which is conform with it, starts in a vertex in U, and does not end in a dead end of Player σ.
- A strategy f_σ is a winning strategy for Player σ on U if all plays which are conform with f_σ and start from a vertex in U are wins for Player σ.
- Player σ wins a game $\mathcal{G}$ on $U \subseteq V$ if he has a winning strategy on U.

Winning Regions

- The winning region for Player σ is the set $W_\sigma(\mathcal{G}) \subseteq V$ of all vertices such that Player σ wins $(\mathcal{G}, v)$, i.e.Player 0 wins $\mathcal{G}$ on $\{v\}$.
- Hence, for any $\mathcal{G}$, Player σ wins $\mathcal{G}$ on $W_\sigma(\mathcal{G})$.

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

$$f_\sigma(\pi v) = f_\sigma(\pi' v), \text{ for every} v \in V_\sigma$$

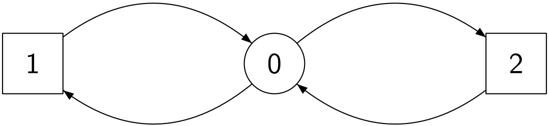
Theorem

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

See [GTW02, Chaps. 6 and 7]

Games that are not Memoryless

Colors $0, 1, 2$ must all occur infinitely often to win a play.
Player 0 must remember something (but the strategy is finite memory = forgetful strategy).



Recall: In Muller games, we specify a sets of colors $\mathcal{F} = \{F_1, \ldots, F_m\} \subseteq 2^C$ such that one F_i is "exactely" visited infinitely often: $Win = \{\pi \in V^\omega | Inf_\chi(\pi) \in \mathcal{F}\}$

Forgetful Determinacy of Regular Games

- Muller games (and any other regular games, Rabin, Streett, Rabin Chain, Buchi, ...) can be simulated by **larger** parity games.
- Hence they are also determined (from the determinacy result from [Mar75] for every game with Borel type).

Corollary

Regular games are **forgetful** *determined.*

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

Step 2. can be carried out in polynomial time: $\mathcal{G}_f$ is a subgraph of $\mathcal{G}$ where all edges $(v, v")$ where $v" \neq f(v)$ have been eliminated. Given $\mathcal{G}_f$, check existence of a vertex v' reachable from v such that (1) $\chi(v')$ is odd and (2) v' lies on cycle in $\mathcal{G}_f$ containing only priorities greater than equal to $\chi(v')$. Such v' does not exist iff Player 0 has a winning strategy. Hence, $\textsc{Wins} \in$ NP. By determinacy, deciding $(\mathcal{G}, v) \notin \textsc{Wins}$ means to decide whether v is a winning position for Player 1 (as above but 1') $\chi(v')$ is even), or use algorithm above on the dual game. Hence, $\textsc{Wins} \in$ co-NP.

Algorithms for Computing Winning Regions

We will see simple winning conditions:

- Reachability (and Safety) Games
- Buchi Games (particular parity games with priorities $0, 1$).

For the general case, there exists many algorithms, all exponential in the number of priorities; see the literature, e.g. [GTW02, Chap. 7].
Recall the problem is in NP $\cap$ co-NP.

Fundamental Open Problem
Does there exists a polynomial algorithm to solve parity games?

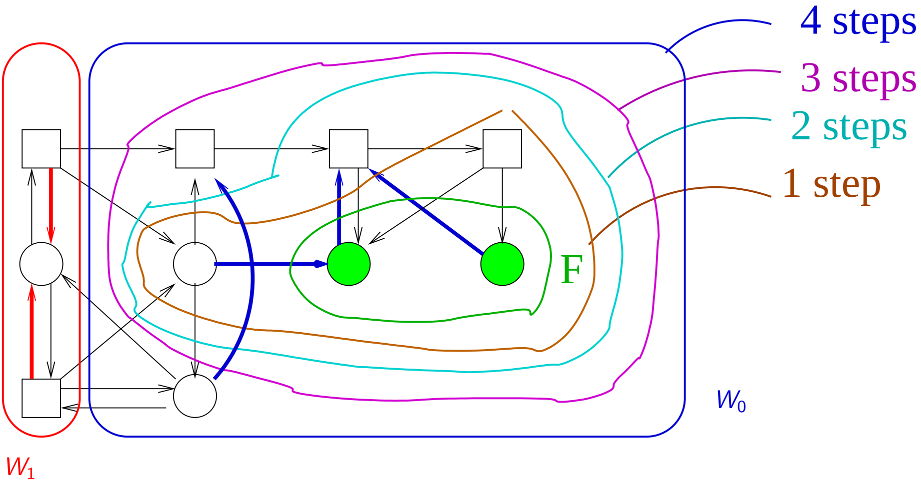
Reachability Games

Given an arena $G = (V, V_0, E)$ and a set $F \subseteq V$, we consider the winning condition

$$\text{Player 0 wins the play } \pi \;\Leftrightarrow\; \exists j, \pi(j) \in F$$

- The winning regions W_0 and W_1 are computable.
- Principle: compute the sets

$$\mathtt{Attr}_0^i(F) \overset{\text{def}}{=} \{v \in V \mid \text{from } v \text{ Player 0 can force a visit of } F \text{ in } \le i \text{ moves}\}$$


4 steps
3 steps
2 steps
1 step
F
W_0
W_1

Computing Attractors

$$\mathtt{Attr}_0^0(F) = F \qquad \mathtt{Attr}_0^{i+1}(F) = \begin{cases} \mathtt{Attr}_0^i(F) \\ \cup\{v \in V_0 \mid \exists v E v' \text{ and } v' \in \mathtt{Attr}_0^i(F)\} \\ \cup\{v \in V_1 \mid \forall v' \text{ s.t. } vEv', v' \in \mathtt{Attr}_0^i(F)\} \end{cases}$$

Then $\mathtt{Attr}_0^0(F) \subseteq \mathtt{Attr}_0^1(F) \subseteq \mathtt{Attr}_0^2(F) \subseteq \ldots$ eventually stabilizes.

The 0-Attractor of F is $\mathtt{Attr}_0(F) \stackrel{\text{def}}{=} \cup_i^{|V|} \mathtt{Attr}_0^i(F)$

The 1-Attractor of F, $\mathtt{Attr}_1(F)$, is defined analoguously.

Proposition

$$W_0 = \mathtt{Attr}_0(F) \text{ and } W_1 = V \setminus \mathtt{Attr}_0(F)$$

$\mathtt{Attr}_0(F) \subseteq W_0$: For $v \in \mathtt{Attr}_0(F) \cap V0$, the strategy is to choose $v' \in vE$ such that $dist(v', F) < dist(v, F)$.
$W_0 \subseteq \mathtt{Attr}_0(F)$: if not in $\mathtt{Attr}_0(F)$ then Player 1 has a way to keep the play away from $\mathtt{Attr}_0(F)$, hence from F.
$\mathtt{Attr}_0(F)$ can be computed in linear time: use bacward breath-first search.

Buchi Games

Given an arena $G = (V, V_0, E)$ and a set $F \subseteq V$, we consider the winning condition

$$\text{Player 0 wins the play } \pi \;\Leftrightarrow\; \exists^{\omega} j,\, \pi(j) \in F$$

that is $Inf(\pi) \cap F \neq \emptyset$.

- The winning regions W_0 and W_1 are computable.
- Principle: compute the sets

$$\texttt{Recur}_0^i(F) \stackrel{\text{def}}{=} \{v \in V \,|\, \text{from } v \text{ Player 0 can enforce at least } i \text{ visits of } F\}$$

Computing Recurrence Sets

$\mathtt{Recur}_0^0(F) = F$
$\mathtt{Attr}_0^+(R) \stackrel{\text{def}}{=} \{v \in V \,|\, \text{from } v \text{ Player 0 enforce visit of } F \text{ in } \geq 1 \text{ moves}\}$
$\mathtt{Recur}_0^{i+1}(F) = F \cap \mathtt{Attr}_0^+(\mathtt{Recur}_0^i(F))$

- $F \supseteq \mathtt{Recur}_0^1(F) \supseteq \mathtt{Recur}_0^2(F) \supseteq \dots$
- $\mathtt{Recur}_0(F) \stackrel{\text{def}}{=} \cap_{i \geq 1} \mathtt{Recur}_0^i(F) = \mathtt{Recur}_0^{i_0}(F)$ for some i_0.

Proposition

$$W_0 = \mathtt{Attr}_0(\mathtt{Recur}_0(F)) \textit{ and } W_1 = V \setminus \mathtt{Attr}_0(\mathtt{Recur}_0(F))$$

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.

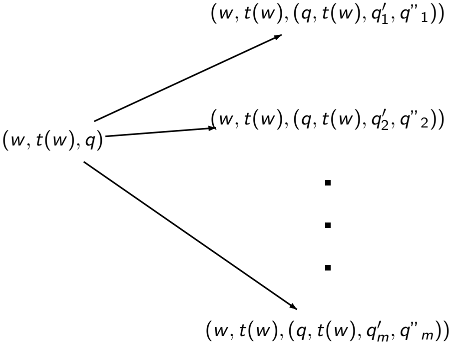
The Emptiness Problem: $\mathcal{A} \rightsquigarrow \mathcal{A}' \rightsquigarrow \mathcal{G}_{\mathcal{A}'}$

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

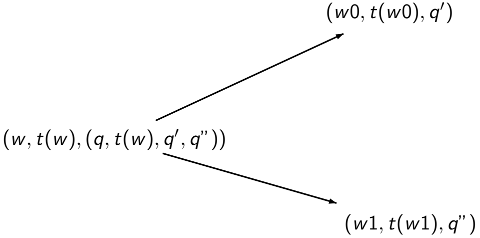
0-positions are of the form $(w, t(w), q)$.
Moves from $(w, t(w),)$, with
$\delta(q, t(w)) = \{(q'_1, q"_1), (q'_2, q"_2), \ldots (q'_m, q"_m)\}$ are:



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

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

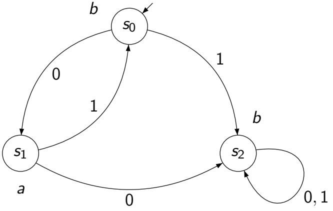
- V_0 = set of triples $(w, t(w), q) \in \{0,1\}^* \times \Sigma \times Q$
- V_1 = set of triples $(w, t(w), \tau) \in \{0,1\}^* \times \Sigma \times \delta$
- Moves ...
- Initial position in $(\epsilon, t(\epsilon), q^0) \in V_0$
- Priorities:

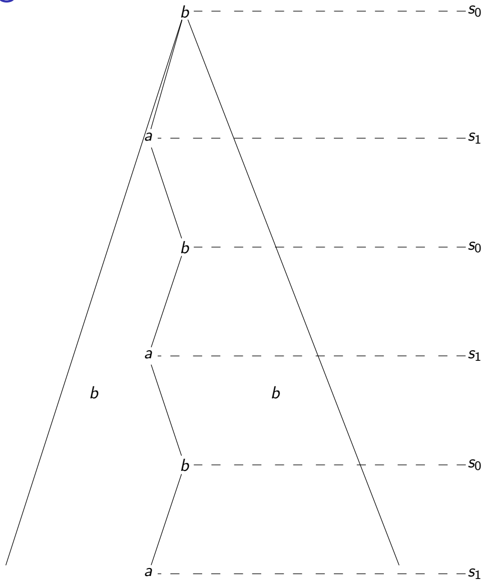
$$\chi((w, t(w), q)) = c(q)$$
$$\chi((w, t(w), (q, t(w), q', q"))) = c(q)$$

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



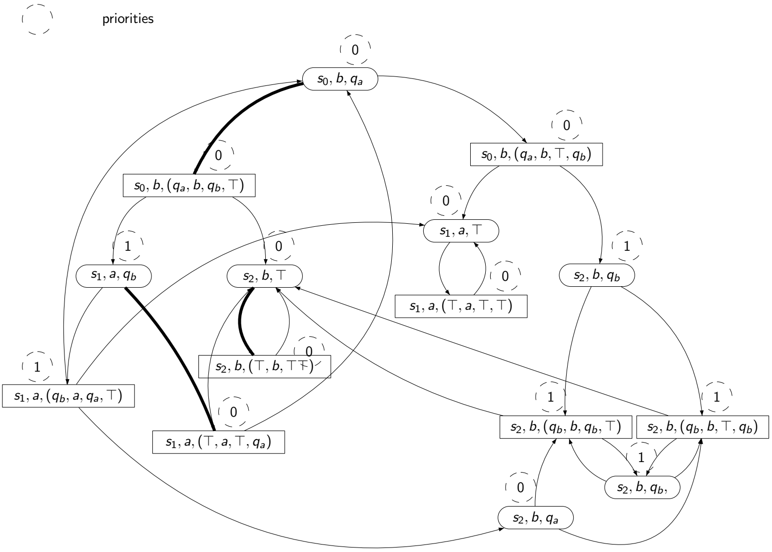


With the automaton:

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

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

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



The Emptiness Problem: 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

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

$$
(q_b, a, q_a, \top), (q_b, a, \top, q_a) \rightsquigarrow \ldots \qquad (q_b, b, q_b, \top), (q_b, b, \top, q_b) \rightsquigarrow \ldots
$$

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

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

From IF Automata to Parity Games

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

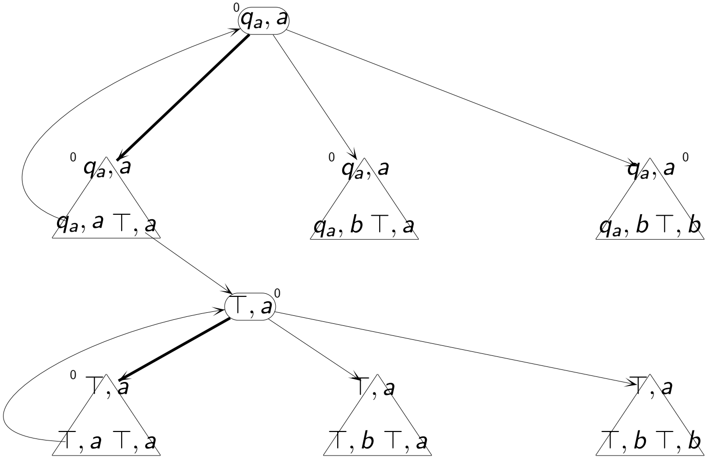
- Positions $V_0 = Q$ and $V_1 = \delta$
- Moves for all $(q, q', q'') \in \delta$
  - $(q, (q, q', q''))) \in E$
  - $((q, q', q''), q'), ((q, q', q''), q'') \in E$
- Priorities $\chi(q) = c(q) = \chi((q, q', q''))$

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]).
Important remark: the universality problem is EXPTIME-complete (already for finite trees).

Mu-Calculus and Parity Tree Automata

Mu-calculus Syntax for this lecture

we use L and R as the directions for successors:

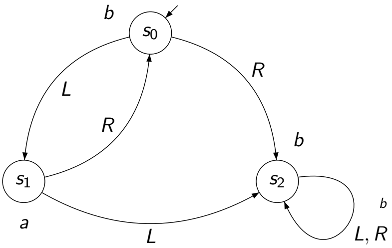
- Alphabet Σ and Propositions $Prop = \{P_a\}_{a \in \Sigma}$
- Variables $Var = \{Z, Z', Y, \dots\}$
- Formulas

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

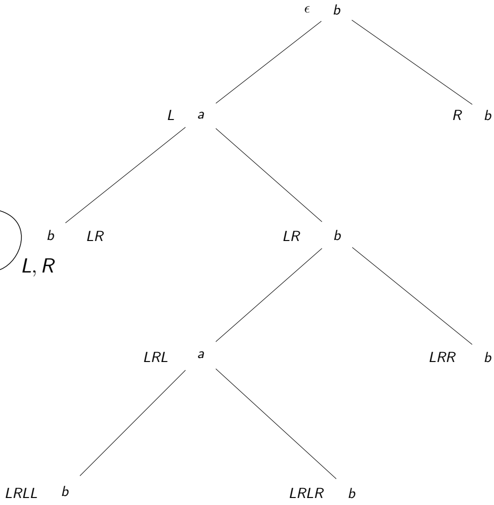
where $Z \in Var$.

Recall

$$
\begin{tikzcd}
\text{a tree } t \arrow[dddd, dashed, two heads] & & & \mathcal{G}_{\mathcal{A},t} \arrow[dddd, dashed, two heads] \\
& \times & & \\
& & \mathcal{A} \arrow[uur, two heads] \arrow[ddr, two heads] & \\
& \times & & \\
\text{a KS } (\mathcal{S}, s^0) & & & \mathcal{G}_{\mathcal{A},(\mathcal{S},s^0)}
\end{tikzcd}
$$

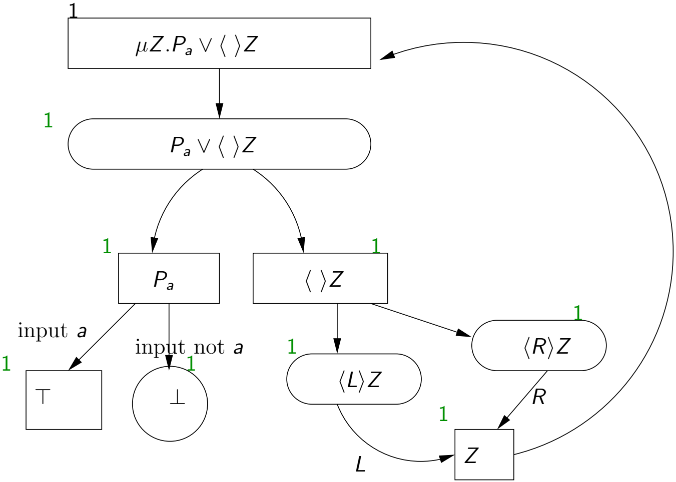


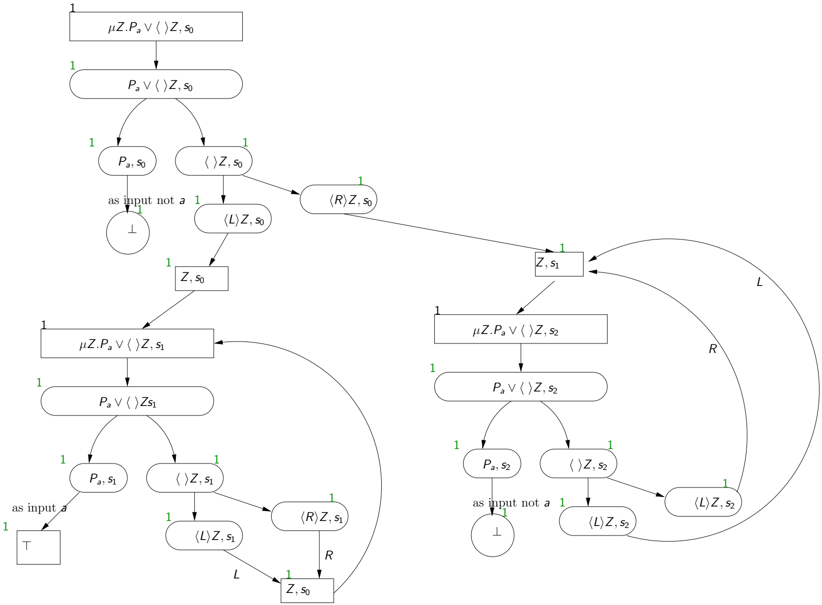
$state(LRLL) = s_2$
hence the label b



Automaton for the Formula **EF** a

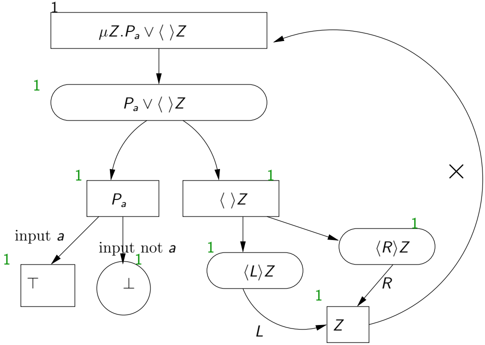
Or equivalently, for the Mu-calculus formula $\mu Z.P_a\langle\ \rangle Z$

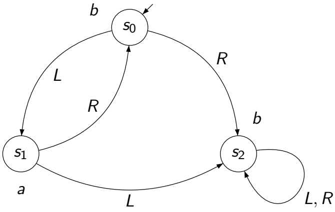




μZ.P_a ∨ ⟨ ⟩Z, s_0
P_a ∨ ⟨ ⟩Z, s_0
P_a, s_0
⟨ ⟩Z, s_0
⟨R⟩Z, s_0
as input not a
⊥
⟨L⟩Z, s_0
Z, s_0
Z, s_1
L
R
μZ.P_a ∨ ⟨ ⟩Z, s_1
P_a ∨ ⟨ ⟩Zs_1
P_a, s_1
⟨ ⟩Z, s_1
as input a
⊤
⟨R⟩Z, s_1
⟨L⟩Z, s_1
Z, s_0
μZ.P_a ∨ ⟨ ⟩Z, s_2
P_a ∨ ⟨ ⟩Z, s_2
P_a, s_2
⟨ ⟩Z, s_2
⟨L⟩Z, s_2
⟨L⟩Z, s_2
1

The Game $\mathcal{G}(\mathcal{A}(\mathbf{EF}a), (\mathcal{S}, s_0))$

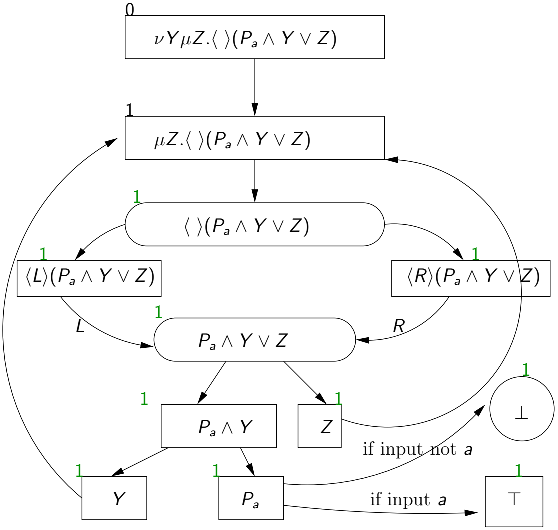




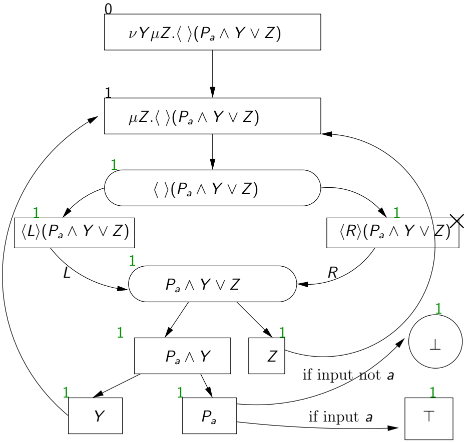
On the board ...

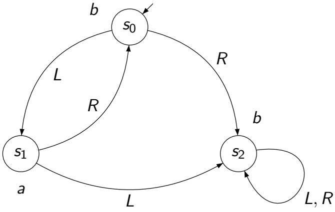
Automaton for the Formula $\mathbf{E}\ \overset{\infty}{\mathbf{F}}\ a$

Or equivalently, for the Mu-calculus formula $\nu Y.\mu Z.\langle\ \rangle(P_a \wedge Y \vee Z)$



The Game $\mathcal{G}(\mathcal{A}(\mathbf{E}\overset{\infty}{\mathbf{F}}\ a), (\mathcal{S}, s_0))$

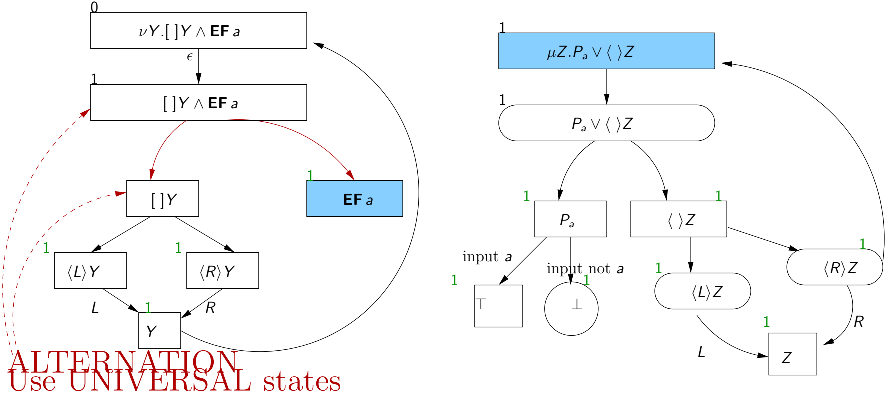




On the board ...

Automaton for the Formula **AG EF** *a*

Or equivalently, for the Mu-calculus formula $\nu Y.[\,]Y \wedge (\mu Z.P_a \vee \langle\,\rangle Z)$



Alternating Tree Automata

- For NDPT automata

$$\delta(q, a) = \{(q'_1, q''_1), (q'_2, q''_2)\}$$

means: From state q on input labeled by a,
(1) non-deterministically choose between the two "disjuncts" (q'_1, q''_1) and (q'_2, q''_2), and
(2) proceed accordingly to the Left and Right sons of w in t.

- Notice: (q'_1, q''_1) and $(q'_2, q''_2)]$ are disjuncts, e.g.

(q'_1, q''_1) is the instruction:

"Proceed left with q'_1 and proceed right with q''_1"

$$(q'_1, L) \wedge (q''_1, R)$$

We then write,

$$\delta(q, a) = (q'_1, L) \wedge (q''_1, R) \vee (q'_2, L) \wedge (q''_2, R)$$

Formal Definition of ATA

- Universal moves, similar to alternating Turing machines extend non-deterministic Turing machines.
- An alternating tree automaton is $\mathcal{A} = (Q, Q^{\exists}, Q^{\forall}, \Sigma, q0, \delta, Acc)$
  - $\{Q^{\exists}, Q^{\forall}\}$ is a partition of Q
  - $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{L, R, \epsilon\})$ is a function and ϵ-transitions are allowed.

$$\delta(q, a) = (q', \epsilon) \wedge (q_1, L) \wedge (q_2, L) \wedge (q_3, R) \vee \ldots$$

- Alternating Tree Automata extend NDPT Automata
- Notice that different "copies" of the automaton can proceed along the same subtree, e.g. $\mathcal{A}, q_1$ and $\mathcal{A}, q_1'$ on the left subtree of nodes labeled by a.

Semantics of ATA

- see [GTW02, Chap. 9]
- Parity games provide a natural way to define $L(\mathcal{A})$ for every ATA $\mathcal{A}$.
- Determinacy of games gives the closure by complemention, and the construction is easy:
  Dualize “Players” and shift the colors.

$$\delta(q,a) = [(q',\epsilon) \wedge (q_1,L) \wedge (q_2,L) \wedge (q_3,R)] \vee \dots$$

$$\rightsquigarrow$$

$$\bar{\delta}(q,a) = [(q',\epsilon) \vee (q_1,L) \vee (q_2,L) \vee (q_3,R)] \wedge \dots$$

Properties of Alternating Tree Automata

- Closed under disjunction and conjunction
- Closed under negation (complementation), see proof next slide
- Unfortunately, it is difficult to show that alternating automata are closed under projection. [MS95] showed that any alternating automaton is equivalent to a non-deterministic automaton (exponential number of states).

Complementation of Alternating Parity Tree Automata

Lemma

For every alternating parity tree automaton $\mathcal{A}$ there is a dual parity tree automaton $\bar{\mathcal{A}}$ such that $L(\bar{\mathcal{A}}) = Trees(\Sigma) \setminus L(\mathcal{A})$. Moreover, regarding size, $|\bar{\mathcal{A}}| = |\mathcal{A}|$

$\mathcal{A} = (Q, Q^{\exists}, Q^{\forall}, \Sigma, q0, \delta, Acc) \rightsquigarrow \bar{\mathcal{A}} = (Q, Q^{\forall}, Q^{\exists}, \Sigma, q0, \bar{\delta}, \bar{c})$ where $\bar{c}(q) = c(q) + 1$ for every $q \in Q$. Now, compare $\mathcal{G}(\mathcal{A}, t)$ and $\mathcal{G}(\bar{\mathcal{A}}, t)$

- Same graph but positions of Player 0 become positions of Player 1, and vice versa.
- For every infinite play π, π is winning for Player 0 in $\mathcal{G}(\mathcal{A}, t)$ iff π is winning for Player 1 in $\mathcal{G}(\bar{\mathcal{A}}, t)$.
  Hence Player 0 has a winning strategy in $\mathcal{G}(\mathcal{A}, t)$ iff Player 1 has a winning strategy in $\mathcal{G}(\bar{\mathcal{A}}, t)$ (same strategy).
- So, $t \in L(\mathcal{A})$ iff $t \notin L(\bar{\mathcal{A}})$

Decision Problems

- Membership Problem for ATA
  $\mathcal{A} = (Q, Q^{\exists}, Q^{\forall}, \Sigma, q0, \delta, c)$, k colors, and $t \in \mathit{Trees}(\Sigma)$, does $t \in L(\mathcal{A})$?
  - t is regular, as the unravelling of some finite Kripke Structure $(\mathcal{S}, s^0)$.
  - Build the finite parity game $\mathcal{G}(\mathcal{A}, (\mathcal{S}, s^0))$ and solve it (decidable).
  - The size of $\mathcal{G}(\mathcal{A}, (\mathcal{S}, s^0))$: $|Q| \times |S|$ positions and k priorities
  - Complexity in NP $\cap$ co-NP (as for parity games)
- Emptiness Problem for ATA
  $\mathcal{A} = (Q, Q^{\exists}, Q^{\forall}, \Sigma, q0, \delta, c)$, is $L(\mathcal{A}) = \emptyset$?
  - See [GTW02, Chap. 9]
  - Alternatively, transform $\mathcal{A}$ into a non-deterministic tree automaton $\mathcal{B}$, and solve emptiness of non-deterministic tree automata
  - Complexity: EXPTIME-complete

Mu-calculus and Alternating Parity Tree Automata

- From the Mu-calculus to Alternating Tree Automata: Given a sentence $\beta \in L_\mu$ (in positive normal form), we construct in polynomial time an ATA $\mathcal{A}_\beta$ such that

$$L(\beta) = L(\mathcal{A}_\beta)$$

  The automaton has $|\beta|$ states and $O(|ad(\beta|)$ colors.
- From Alternating Tree Automata to the Mu-calculus: given an AT Automaton $\mathcal{A}$ we can build a formula $\beta_{\mathcal{A}}$ "equivalent" to $\mathcal{A}$. The translation from Alternating Parity Tree Automata to the Mu-calculus uses vectorial Mu-calulus, see [AN01].

Summary about the Mu-Calculus

- The Mu-calculus $\equiv$ Alternating Parity Tree Automata ($\equiv$ NDPT Automata)
- They all characterize regular languages of infinite trees.
- The Mu-calculus $\equiv$ MSO on trees
- More generally: The Mu-calculus $\equiv$ bisimulation invariant properties of MSO [JW95]
- Complexity results:
  - Satisfiability is EXPTIME-complete ([SE89, EJ88]).
  - Model-checking is NP $\cap$ co-NP; it is open whether it is in P.
- The Mu-calculus subsumes every temporal logics.
  - CTL translates into the alternation free fragment of the Mu-calculus. It has a polynomial time model-checking procedure (retrieve why according to previous results).
  - CTL* can be translated into the Mu-calculus [Dam94], but there is an exponential blow-up.

Importance of Games

- Useful for fundamental problems on automata
- henceforth for the Satisfiability and Model-checking Problem of modal and temporal logics.
- A "reversed" reduction:

  A parity game $\mathcal{G}, V_0, V_1, E)$ with a priority function $\chi : V \to \{0, \ldots, k-1\}$ (k priorities) can be seen as a Kripke Structure (V, E, λ) where λ maps states onto the set of propositions $\{V_0, V_1, P_0, \ldots, P_k\}$ where $P_i = \{v \mid \chi(v) = i\}$.
  The formula

  $$Win_k \stackrel{\text{def}}{=} \nu Z_0.\mu Z_1.\ldots.\theta Z_{k-1} \bigvee_{j=0}^{k-1} ((V_0 \wedge P_j \wedge (\langle . \rangle Z_j) \vee (V_1 \wedge P_j \wedge ([.]Z_j))$$

  (where $\theta = \nu$ if k is odd, and $\theta = \mu$ if k is even)
  characterizes the winning region W_0 of Player 0 in any parity game with priorities $0, \ldots, k-1$.

A. Arnold and D. Niwinski.
Rudiments of mu-calculus.
North-Holland, 2001.

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.

M. Dam.
CTL$\star$ and ECTL$\star$ as fragments of the modal μ-calculus.
Theoretical Computer Science, 126(1):77–96, 1994.

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.

The complexity of tree automata and logics of programs.
In *Proc. 29th IEEE Symp. Foundations of Computer Science, White Plains, New York*, pages 328–337, October 1988.

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.

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.

D. Janin and I. Walukiewicz.
Automata for the modal mu-calculus and related results.
In Jirí Wiedermann and Petr Hájek, editors, *Mathematical Foundations of Computer Science 1995, 20th International Symposium*, volume 969 of *lncs*, pages 552–562, Prague, Czech Republic, 1 September– 28 August 1995. Springer.

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.

D. Martin.
Borel determinacy.
Annales of Mathematics, 102:363–371, 1975.

A. W. Mostowski.

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

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.

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

R. S. Streett and E. A. Emerson.
An automata theoretic decision procedure for the propositional mu-calculus.
Information and Computation, 81(3):249–264, 1989.

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.