Automata, Logics, and Infinite Games

S. Pinchinat

IRISA, Rennes, France

Master2 RI 2007

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).

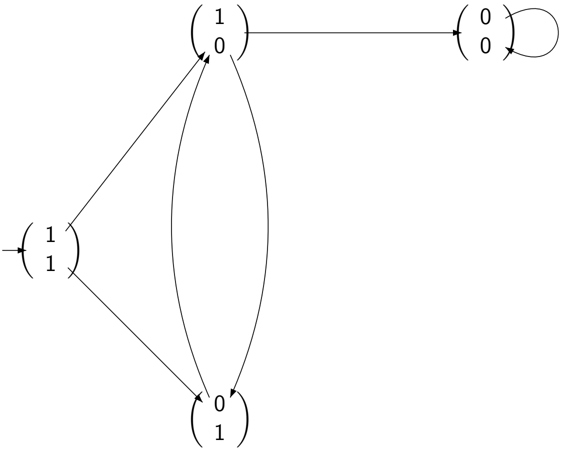
- A Kripke Structure over $Prop$ is $\mathcal{S} = (S, R, \lambda)$
  - S is a set of states (worlds)
  - $R \subseteq S \times S$ is a transition relation
  - $\lambda : S \rightarrow 2^{Prop}$ associates those p_i which are assumed true in s. Write $\lambda(s)$ as a bit vector $(b_1, \ldots, b_n)$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

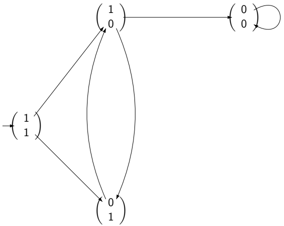


Paths and Words

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 $(\mathcal{S}, s)$:

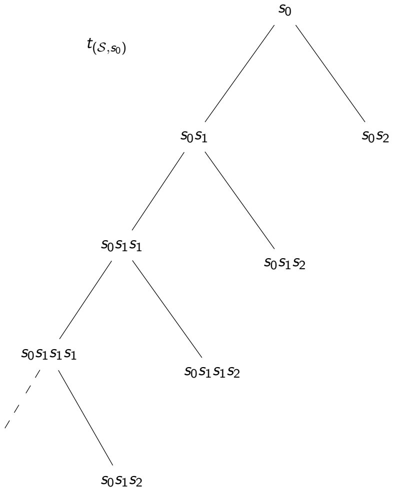
$$\alpha^i \models \mathbf{E}\,\psi \text{ iff } \alpha'^i \models \psi \text{ for some } \alpha' \text{ in } (\mathcal{S}, s) \text{ st. } \alpha[0, \ldots, i] = \alpha'[0, \ldots, i]$$

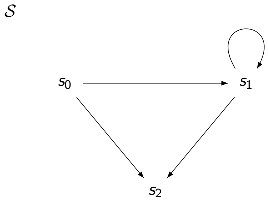
Let $\mathbf{A}\,\psi \stackrel{\text{def}}{=} \neg\mathbf{E}\,\neg\psi$

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 $t_{(\mathcal{S},s)}$.
- Paths of $\mathcal{S}$ are retrieved in the tree $t_{(\mathcal{S},s)}$ as branches.





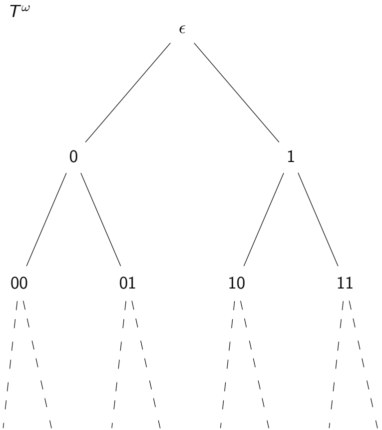
Σ-Labeled Full Binary Trees

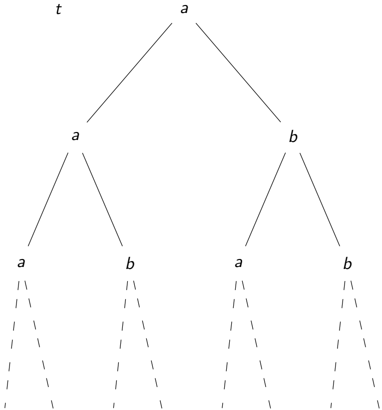
For simplicity we assume that states have exactly two successors $\Rightarrow$ we consider (only) binary trees

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

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

Example





Model-checking and Satisfiabilty

- The Model-checking Problem: does a tree t satisfy the specification *Spec*?
- The Satisfiability Problem: Is there a tree model of the specification *Spec*?

$$\begin{array}{rcl} \text{Model-checking} & = & \text{Program Verification} \\ \text{Satisfiability} & = & \text{Program Synthesis} \end{array}$$

About the content of this course

- Tree Automata: devices which recognize models of formulas:

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

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

  The Satisfiability Problem $\rightsquigarrow$ The Emptiness Problem

- Games are fundamental to solve those

- Mu-calculus is a unifying logical formalism

Games

- Two-person games on directed graphs.
- How they are 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

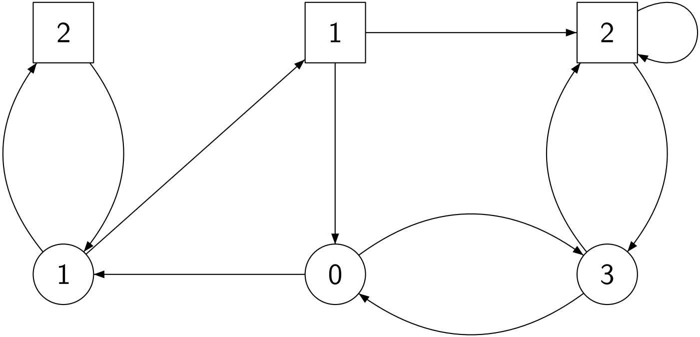
- Let be G an arena and $Win \subseteq V^{\omega}$ be the winning condition
- The pair $\mathcal{G} = (G, Win)$ is called a game
- Player 0 is declared the winner of a play π in the game $\mathcal{G}$ if
  - π is finite and $last(\pi) \in V_1$ and $last(\pi)E = \emptyset$, or
  - π is infinite and $\pi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

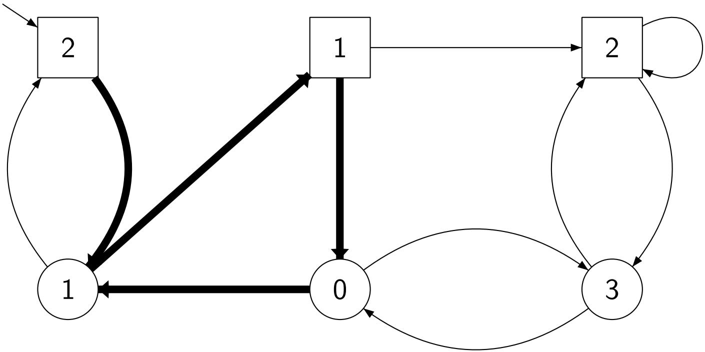
Parity Game Example



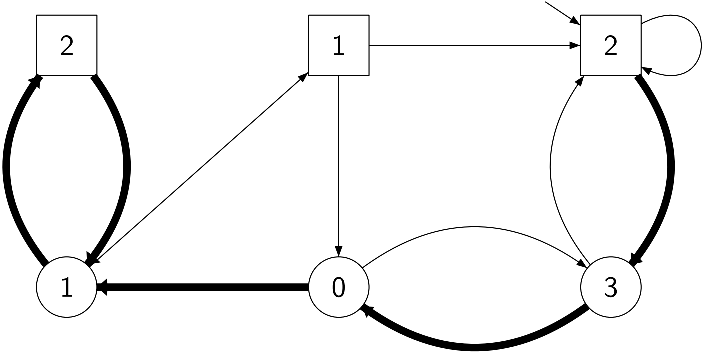
Strategies

- A strategy for Player σ is a function $f_\sigma \colon V^* V_\sigma \to V$
- A prefix play $\pi = v_0 v_1 v_2 \ldots v_l$ 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.
- 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 Play for Player 0



Winning Play for Player 1



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 $W_\sigma(\mathcal{G})$.

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 positional (or memoryless) strategy whenever when defined for πv and $\pi' v$, we have $f_\sigma(\pi v) = f_\sigma(\pi' v)$.

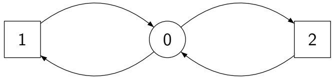
Theorem

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

See [GTW02, Chaps. 6 and 7]

Games that are not Memoryless

In Muller games, a set $\mathcal{F} \subseteq 2^C$ is given and $Win = \{\pi \in V^\omega | Inf_\chi(\pi) \in \mathcal{F}\}$
Here every color must occur infinitely often; Player 0 must remember something (but the strategy is finite memory = forgetful strategy)



Forgetful Determinacy of Regular Games

Muller games (and any other regular games, Rabin, Streett, Rabin Chain, Buchi, ...) can be simulated by larger parity games. They are also determined (also see determinacy result from [Mar75] for every game with Borel type). As a corollary of previous results, we have the very general following result for

Corollary

Regular games are forgetful determined.

Algorithmic Results

Theorem

$\textsc{Wins} =$
$\{(\mathcal{G}, v) \,|\, \mathcal{G}$ *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

Read “Algorithms for Parity Games”, Chapter 7 of Automata, Logics, and Infinite Games A Guide to Current Research. Series: Lecture Notes in Computer Science , Vol. 2500 Grdel, Erich; Thomas, Wolfgang; Wilke, Thomas (Eds.) 2002, XI, 385 p.

Automata on Infinite Objects

- We refer to [Tho90]
- Connection with Logic LTL, CTL* - membership and emptiness -
- Connection with Games
- Automata on words, trees, and graphs.

ω-automata

We refer to [GTW02, Chap. 1]

- Inputs are infinite words.
- Acceptance conditions: Buchi, Muller, Rabin and Streett, Parity
- All coincide with ω-regular languages ($L = \bigcup_i K_i R_i^\omega$)
- *LTL* corresponds to star-free languages

Automata on Infinite Trees

- Acceptance conditions: Buchi, Muller, Rabin and Streett, Parity on each branch of the input tree.
- Buchi tree automata are weaker [Rab70].
  [KSV96] L is recognizable by a nondeterministic Buchi word automaton but not by a deterministic Buchi word automaton iff $\mathit{trees}(L)$ is recognizable by a Rabin tree automaton and not by a Buchi tree automaton.
- Here we restrict to labeled full binary trees and to parity acceptance conditions, but the resultts generalize.

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)
  - $\delta \subseteq Q \times \Sigma \times Q \times$ 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

- A run of $\mathcal{A}$ 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

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

- A run ρ is accepting (successful) iff for every path $\pi \in \{0,1\}^\omega$ of the tree ρ the parity acceptance condition is satisfied:

$$\mathit{minInf}_c(\rho) \text{ is even}$$

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

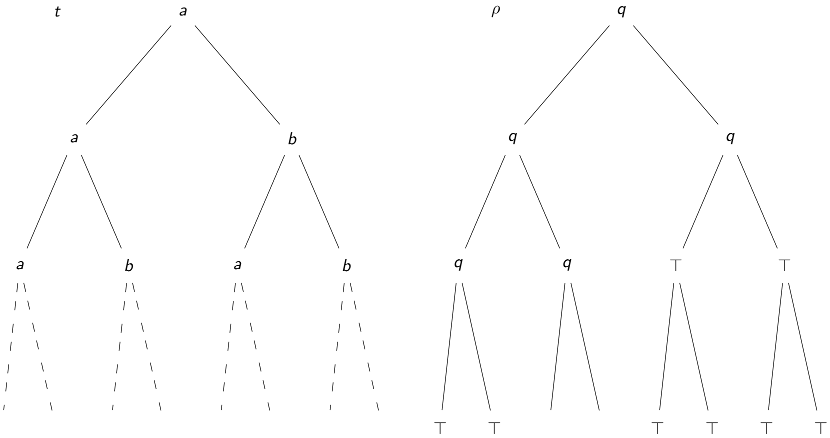
Example 1

- Let L_0 be the set of trees whose every path has an a ($\mathbf{F}a$ in LTL)
- Consider the automaton with states $q_a, \top$, transitions

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

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

Example Run



Other Acceptance Conditions

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

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

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

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

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

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

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

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

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.

Example 2

- 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 $c(q_a) = c(\top) = 0$ (this a Buchi condition, only 0 and 1 colors)

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 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.

Regular Tree Languages and Properties

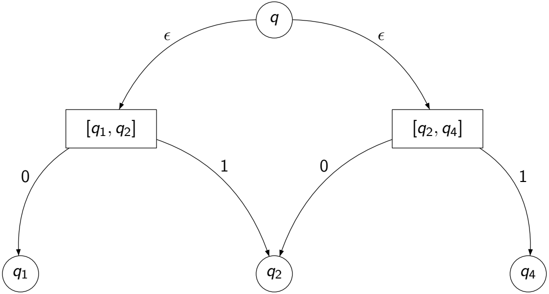
- A tree language $L \subseteq Trees(\Sigma)$ is regular iff there exists a parity (Muller, Rabin, Streett) tree automaton which recognizes L.
- The complement of L_a^∞ (finitely many a's on each branch) is not recognizable by any Buchi tree automaton
- Tree automata are closed under sum, projection, and complementation.
  - Tree automata cannot be determinized: $L_a^\exists \subseteq Trees(\{a, b\})$, the language of trees ahaving one node labedled 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]

Alternating Tree Automata

- Design an automaton for the language $\{t \in \mathit{Trees}(\{a,b,c\}) \mid t \models \mathbf{A}\,\mathbf{F}a \wedge \mathbf{A}\,\mathbf{F}b \wedge \mathbf{A}\,\mathbf{F}c\}$
- Quite difficult to design with a non-deterministic tree automaton (combinatorics between the occurrences of a and b and c) but easy to write as
$$\delta(q,*) = (q_a,\epsilon) \wedge (q_b,\epsilon) \wedge (q_c,\epsilon)$$
where q_a (resp. q_b, q_c) is the initial states of the automaton for $\mathbf{A}\,\mathbf{F}a$ (resp. $\mathbf{A}\,\mathbf{F}b$, $\mathbf{A}\,\mathbf{F}c$).
- The automaton splits into three "copies" checking in parallel $\mathbf{A}\,\mathbf{F}a$, $\mathbf{A}\,\mathbf{F}b$, and $\mathbf{A}\,\mathbf{F}c$.

Alternation

- Recall $\delta(q, a) = \{(q_1, q_2), (q_2, q_4)\}$ means from state q and node w in the input tree (with $t(w) = a$): (1) non-deterministically choose between the two "disjuncts" $[q_1, q_2]$ and $[q_2, q_4]$, and (2) proceed accordingly to the left and right sons of w in t.
- We extend the non-deterministic tree automaton with a notion of universal moves (similar to alternating Turing machines extend non-deterministic Turing machines).



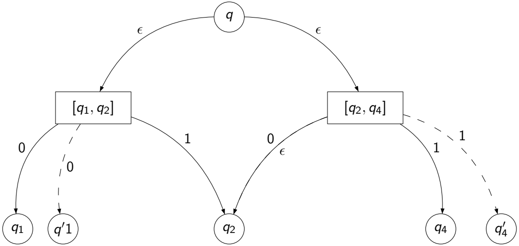
Alternating Tree Automata extend Non-deterministic Tree Automata

- In the transitions relation, we allow positive Boolean combinations of terms (q, d), $d \in \{0, 1, \epsilon\}$:
  - For non-deterministic automata, we had $\delta(q, a) = (q_1, 0) \wedge (q_2, 1) \vee (q_2, 0) \wedge (q_4, 1)$
  - Now we can write things like $\delta(q, a) = (q_1, 0) \wedge (q_1', 0) \wedge (q_2, 1) \vee (q_2, 0) \wedge (q_4, 1) \wedge (q_5, \epsilon)$

  Notice that different "copies" of he 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.

Example

$\delta(q, a) = (q_1, 0) \wedge (q_1', 0) \wedge (q_2, 1) \vee (q_2, 0) \wedge$



$(q_4, 1)$

- We use parity games to define the semantics of ATA
- Parity games provide a straightforward construction to complement ATA (parity accpetance). Determinacy of games gives the correction of this construction.
- We use parity games to show the decidability of the membership problem (for emptiness see [GTW02, Chap. 9]).
- We will see that ATA have a logical counter part: the Mu-calculus, an extension of modal logic with fix-points.

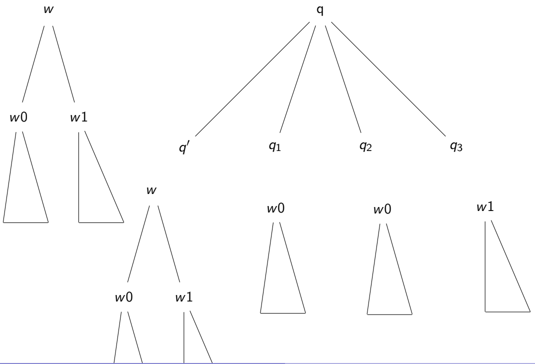
Formal Definition of ATA

An alternating tree automaton is $\mathcal{A} = (Q, Q^{\exists}, Q^{\forall}, \Sigma, q^0, \delta, Acc)$

- $\{Q^{\exists}, Q^{\forall}\}$ is a partition of Q
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \epsilon\})$ is a function and ϵ-transitions are allowed.

We can write $\delta(q, a) = (q', \epsilon) \wedge (q_1, 0) \wedge (q_2, 0) \wedge (q_3, 1) \vee \ldots$

We could give the semantics in terms of runs, as before, but the runs are tree with possibly a degree > 2



Semantics of Alternation Tree Automata

Runs and Acceptance of the automaton are formalized in terms of two-player games.

- Given a tree $t \in \mathit{Trees}(\Sigma)$, we define the acceptance game $\mathcal{G}(\mathcal{A}, t)$ by:
- $V_0 = \{0,1\}^* \times Q^{\exists}$
- $V_1 = \{0,1\}^* \times Q^{\forall}$
- From each position (w, q) and $(q', d) \in \delta(q, t(w))$, there is an edged to (wd, q')
- The acceptance condition Acc consists of the sequences $(w_0, q_0)(w_1, q_1)\ldots$ such that the sequence $q_0 q_1 \ldots$ is in Acc
- $\mathcal{A}$ accepts a tree t iff Player 0 has a winning strategy in $\mathcal{G}(\mathcal{A}, t)$

Alternation Tree Automata over Kripke Structures

Follow the same lines

- Consider a rooted Kripke Structure $(\mathcal{S}, s^0)$ (which unfolds as a tree)
- Define $\mathcal{G}(\mathcal{A}, (\mathcal{S}, s^0))$ as for trees, but notice that if $\mathcal{S}$ is finite so is $\mathcal{G}(\mathcal{A}, (\mathcal{S}, s^0))$
- $\mathcal{A}$ accepts $(\mathcal{S}, s^0)$ iff Player 0 has a winning strategy in $\mathcal{G}(\mathcal{A}, (\mathcal{S}, s^0))$

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. Muller and Schupp showed that

Theorem

(Simulation Theorem) [MS95]
Any alternating tree automaton is equivalent to a non-deterministic tree automaton (with an exponential blow up in the 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, $|\bar{\mathcal{A}}| = |\mathcal{A}|$

Proof $\mathcal{A} = (Q, Q^{\exists}, Q^{\forall}, \Sigma, q^0, \delta, Acc) \rightsquigarrow \bar{\mathcal{A}} = (Q, Q^{\forall}, Q^{\exists}, \Sigma, q^0, \delta, \bar{c})$ 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

- the Membership Problem: given an ATA $\mathcal{A}$ and a tree t, does $t \in L(\mathcal{A})$? (see next slide)
- the Emptiness Problem: given $\mathcal{A}$, is $L(\mathcal{A}) = \emptyset$?

the Membership Problem

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

the Emptiness Problem

$\mathcal{A} = (Q, Q^\exists, Q^\forall, \Sigma, q^0, \delta, c)$, is $L(\mathcal{A}) = \emptyset$?

- First method: Simulation Theorem, and use an algorithm to solve the emptiness of non-deterministic tree automata.x
- Second method: Based on Parity Games on Times (see [GTW02, Chap. 9]).
- Complexity of the Emptiness Problem: EXPTIME-complete

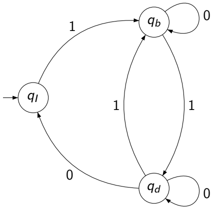
We now look at the Emptiness of Non-deterministic Tree Automaton

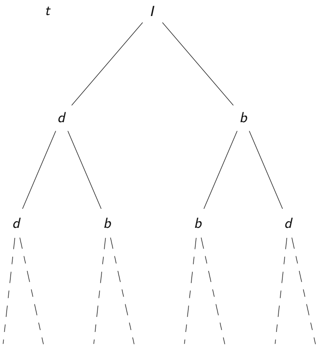
Input-free Automata

- An input-free (IF) automaton is $\mathcal{A}' = (Q, \delta, q_I, Acc)$ where $\delta \subseteq Q \times Q \times Q$
- We may remove Acc.
- Runs are defined as usual; they are trees.
- Determnistic $\Rightarrow$ unique tree, and it is regular
  t is regular iff $\{t^u \mid u \in \{0,1\}^*\}$ is finite
  where $t^u(v) = t(uv)$

Regular Tree generated by deterministic finite-state Automata with an input function

- $A = (Q, \{0,1\}, \Delta, q_I, f)$ a finite automaton
- $f : Q \rightarrow \Sigma'$ an input function
- It generates the tree such that $t(w) = f(\Delta(q_I, w))$





and Deterministic Finite Automata on $\{0,1\}$ and Deterministic IF Automata (without Acc)

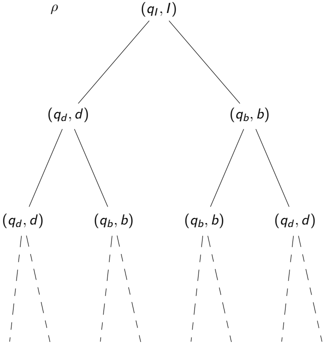
- Let $A = (Q, \{0,1\}, \Delta, q_I, f : Q \rightarrow \Sigma')$
- Define $\mathcal{B} = (Q \times \Sigma', \delta, (q_I, f(q_I)))$ by
  $\forall q \in Q$,

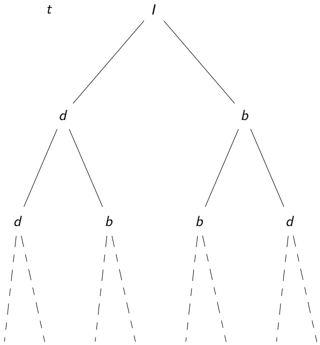
$$((q, f(q)), (\Delta(q,0), f(\Delta(q,0))), (\Delta(q,1), f(\Delta(q,1))) \in \delta$$

- $\mathcal{B}$ is deterministic and a run of $\mathcal{B}$ generates in the second component of its states the trees that A generates.

Example

$\mathcal{B}$ has states $\{(q_I, I), (q_b, b), (q_d, d)\}$ and transitions $((q_I, I), (q_d, d), (q_d, d))$, $((q_d, d), (q_d, d), (q_d, d))$, $((q_d, d), (q_d, d), (q_d, d))$. (q_I, I) is intial





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.

Proof.

$\mathcal{A} = (Q, \Sigma, q^0, \delta, c)$ 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 (ie 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 $p, p', q, q' \in Q$
- $c'(q, a) = c(q)$

Esay to see that lemma holds for this construction. □

From IF Automata to Parity Games

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

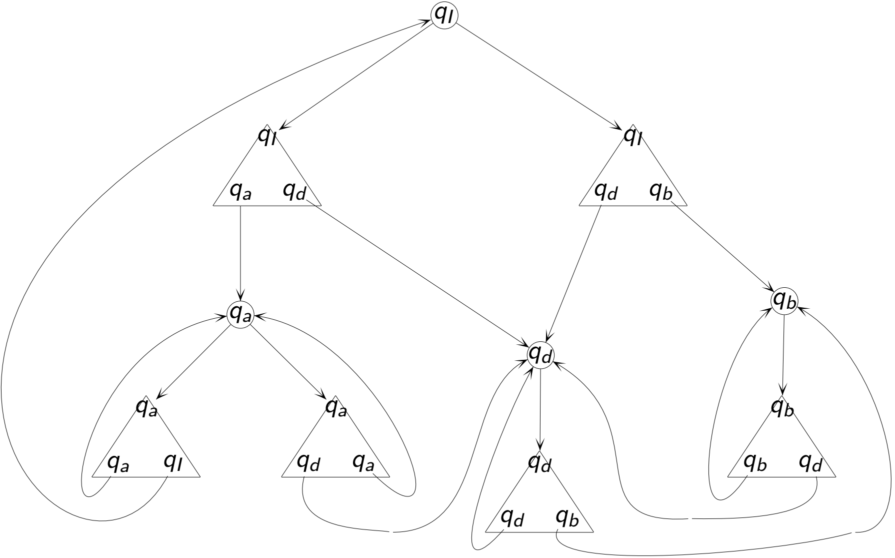
- Positions $V_0 = Q$ and $V_1 = \delta$
- Moves for all $(q, q', q'') \in \delta$
  - $(q, (q, q', q"))\in E$
  - $((q, q', q"), q'), ((q, q', q"), q") \in E$
- Priorities $\chi(q) = c(q) = \chi((q, q', q"))$

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 Nondeterministic Tree Automata

Theorem

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

Proof.

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

Finite Model Property

Corollary

If the language of a parity tree automaton is not empty then it contains a regular tree.

Proof.

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 $Aacc$): 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 parity non-deterministic tree automata is in $NP \cap co\text{-}NP$.

Proof.

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).

The Mu-calculus

Syntax

- Alphabet Σ and Propositions $Prop = \{P_a\}_{a \in \Sigma}$
- Variables $Var = \{Z, Z', \dots\}$
- 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 $\mathit{val} : \mathit{Var} \rightarrow 2^{\{0,1\}^*}$ of the variables.
- For every $N \subseteq \{0,1\}^*$, we write $\mathit{val}[N/Z]$ for val' defined as val except that $\mathit{val}'(Z) = N$
- We define $[\![\, \beta \,]\!]^t_{\mathit{val}} \subseteq \{0,1\}^*$ by:

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

About Fix-points

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

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

By the assumption on "positive" occurrences of Z in β, τ is monotonic: $N' \subseteq N$ implies $\tau(N') \subseteq \tau(N)$ (prove it). 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.

Tarski-Knaster Theorem

Theorem

(Tarski-Knaster) Assume a set D. Let $\tau : 2^D \rightarrow 2^D$ be monotonic, then

$\mu z.\tau(z) = \cap\{z \mid \tau(z) = z\} = \cap\{z \mid \tau(z) \subseteq z\}$

$\nu z.\tau(z) = \cup\{z \mid \tau(z) = z\} = \cup\{z \mid \tau(z) \supseteq z\}$

$\mu z.\tau(z) = \cup_i \tau^i(\emptyset)$, *where i ranges over all ordinals of cardinality at most the state space D; when D is finite, $\mu z.\tau(z)$ is the union of the following ascending chain $\emptyset \subseteq \tau(\emptyset) \subseteq \tau^2(\emptyset)$...*

$\nu z.\tau(z) = \cap_i \tau^i(D)$, *where i ranges over all ordinals of cardinality at most the state space D; when D is finite, $\nu z.\tau(z)$ is the intersection of the following descending chain $D \supseteq \tau(D) \supseteq \tau^2(D)$...*

Therefore, if t is regular, i.e. representing the unravelling of a finite rooted KS $(\mathcal{S}, s)$, the fix-points can be effectively computed.

- "Trivial" formulas:
  $\mu Z.Z$, $\nu Z.Z$, $\mu Z.P$, $\nu Z.P$, $\mu Z.\langle 0\rangle Z \vee \langle 1\rangle Z$, $\nu Z.\langle 0\rangle Z \wedge \langle 1\rangle Z$.
- Intuitively, μ (resp. ν) correspond to finite (resp. infinite) computations.
  - $\mu Z.P_b \vee (\langle 0\rangle Z \vee \langle 1\rangle Z) \wedge P_a$ is equivalent to the CTL formula $\mathbf{E}\, a\, \mathbf{U}\, b$.
  - $\nu Z.P_a \wedge (\langle 0\rangle Z \wedge \langle 1\rangle Z)$ is equivalent to $\mathbf{AG}\, a$.

  (prove it)

- We can push negation inside a formula (notice that $\neg\langle d\rangle\beta = \langle d\rangle\neg\beta$, for $d \in \{0,1\}$)to get a formula in positive normal form.

- Write $t \models \beta$ whenever $\epsilon \in [\![\, \beta \,]\!]^t_{val}$.
- Define $L(\beta) \overset{\text{def}}{=} \{t \in \mathit{Trees}(\Sigma) \mid t \models \beta\}$

Alternation Depth

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

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

For example, $ad(\nu Z.\mu Z'.(Z \wedge P_a) \vee \langle 0 \rangle Z') = 1$ ("infinitely often a along the branch 0^ω").

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

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

Hence the model-checking and the satisfiability problems for the Mu-calculus reduce to the membership and emptiness problems for ATA.

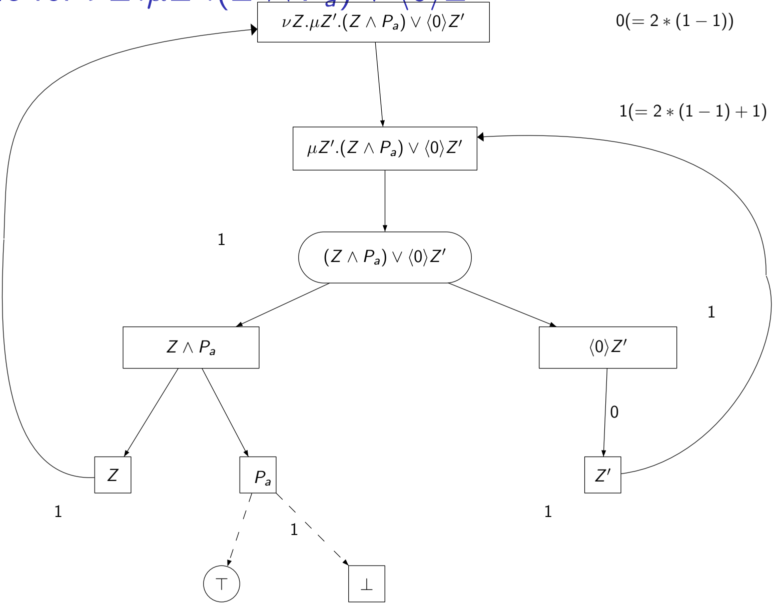
Defintion of $\mathcal{A}_\beta$

$\mathcal{A}_\beta \stackrel{\text{def}}{=} (Q, \Sigma, q^0, \delta, c)$ where

- $Q = \{\alpha \,|\, \alpha \leq \beta\} \cup \{\top, \bot\}$ and $q_I = \beta$
- $Q^\exists$ is composed of all subformulas of the form $\alpha \vee \alpha'$, $Q^\forall$ contains the rest.
- $\delta : Q \times \Sigma \to \mathcal{P}(Q \times \{0, 1, \epsilon\})$ is defined by induction over $\alpha \in Q$:
  - $\delta(P_a, a) = \{(\top, \epsilon)\}$ and $\delta(P_a, b) = \{(\bot, \epsilon)\}$ for all $b \neq a$
  - $\delta(\neg P_a, a) = \{(\bot, \epsilon)\}$ and $\delta(\neg P_a, b) = \{(\top, \epsilon)\}$ for all $b \neq a$
  - $\delta(Z, a) = \{\beta_Z, \epsilon)\}$
  - $\delta(\alpha \wedge \alpha') = \{(\alpha, \epsilon), (\alpha', \epsilon)\}$ and the same for $\delta(\alpha \vee \alpha')$
  - $\delta(\langle d \rangle \alpha) = (\alpha, d)$, for $d \in \{0, 1\}$
  - $\delta(\theta Z.\alpha) = (\alpha, \epsilon)$, for $\theta \in \{\mu, \nu\}$
- The coloring function c is defined by (let $M = ad(\beta)$)
  - $c(\alpha) = 2 * (M - ad(\alpha))$ if α is a ν-formula
  - $c(\alpha) = 2 * (M - ad(\alpha)) + 1$ if α is a μ-formula
  - $c(\alpha) = M$ if α is not a fix-point formula.

For the correctness of the construction see [GTW02, Chap. 10].

Example for $\nu Z.\mu Z'.(Z \wedge P_a) \vee \langle 0 \rangle Z'$



From Alternating Tree Automata to the Mu-calculus

The translation from Alternating Parity Tree Automata to the Mu-calculus uses vectorial Mu-calulus, see [AN01].

Summary

- The Mu-calculus and Alternating Parity Tree Automata have the same expressive power.
- Complexity results:
  - The satisfiability problem for the Mu-calculus is EXPTIME-complete ([SE89, EJ88]).
  - The model-checking for the Mu-calculus 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.

Mu-calculus and Parity Games

We have seen a reduction from the Model-checking Problem of the Mu-calculus to Parity Games (via Automata), but there is a reduction in the reverse direction.

A parity game $\mathcal{G}, V_0, V_1, E)$ with a priority function $\chi : V \rightarrow \{0, \ldots, k-1\}$ (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) defines the winning region of Player 0 in any parity game with priorities $0, \ldots, k-1$.

André Arnold and Irène Guessarian.
Mathematics for Computer Science.
Prentice-Hall, Masson, 1996.

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

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.

Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi.
Relating word and tree automata.
In *Logic in Computer Science*, pages 322–332, 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.