Logic, Automata, and Games

Sophie Pinchinat
IRISA, university of Rennes 1, France
M2RI 2011-2012

Logics of Programs
- Introductory Example
- Kripke Structures
- Behavioral Properties

The Mu-calculus
- The Mu-calculus
- Fundamental Questions

Automata on Infinite Objects
- Generalities
- Non-deterministic Parity Tree (NDPT) Automata

Games
- Generalities
- Parity Games
- Memoryless Determinacy of Parity Games
- Solving Parity Games

Membership and Emptiness Problems for NDPT Automata

Concluding remarks

The Model-Checking Problem

The Model-checking Problem: A system $\text{Sys}$ and a specification $\text{Spec}$, decide whether $\text{Sys}$ satisfies $\text{Spec}$, or not.

Example: Mutual exclusion protocol

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

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

A state is a bit vector of the form (line no. of process 1,line no. of process 2, value of turn)

The initial state is (00000).

$\text{Spec} =$ “some state of the form (1010x) is never reached”, and “always when a state of the form (01xyz) is reached, then later a state of the form (10x’y’z’) is reached” (and similarly for Process 2, i.e. states (xy01z) and (x’y’10z’))

Kripke Structures

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

Definition

A Kripke structure over $\text{Prop}$ is $\mathcal{S} = (S, R, \lambda)$
- $S$ is a set of states
- $R \subseteq S \times S$ is a transition relation
- $\lambda : S \rightarrow 2^{\text{Prop}}$ associates those $p_i$ which are assumed true in $s$.

A rooted Kripke structure is a pair $(S, s)$ where $s$ is a distinguished initial state.
Mutual Exclusion Protocol Example

Let us use:
- $p_1$ and $p_2$ for “being in wait instruction before critical section” for Process 0 and Process 1 respectively
- $p_3$ and $p_4$ for “being in critical section” for Process 0 and Process 1 respectively

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

EXERCISE: Define the KS corresponding to the Mutual Exclusion Protocol

A Toy System

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

Paths and Words

Let $S = (S, R, \lambda)$ be a Kripke structure over $Prop = \{p_1, p_2, \ldots, p_n\}$.
- A path through $(S, s)$ is a sequence $s_0, s_1, s_2, \ldots$ where $s_0 = s$ and $(s_i, s_{i+1}) \in R$ for $i \geq 0$
- Its corresponding word ($\in (2^{Prop})^\omega$) is $\lambda(s_0), \lambda(s_1), \lambda(s_2), \ldots$

For example,

If $\alpha = \{p_1, p_2\}\{p_1\}\{p_2\}\{p_1\}000\ldots$

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

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:

$$\varphi_1, \varphi_2 (\models LTL) ::= a \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi_1 \mid X \varphi_1 \mid \varphi_1 U \varphi_2$$

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

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

Behavioral Properties

Semantics of LTL

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

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

Let

\[
F \varphi \overset{\text{def}}{=} \text{true} U \varphi, \text{ hence } \alpha^i \models F \varphi \text{ iff } \alpha^j \models \varphi \text{ for some } j \geq i.
\]

\[
G \varphi \overset{\text{def}}{=} \neg F \neg \varphi, \text{ hence } \alpha^i \models G \varphi_1 \text{ iff } \alpha^j \models \varphi_1 \text{ for every } j \geq i.
\]

Augmenting LTL: the logic $CTL^*$

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

\[
\psi_1, \psi_2 (\exists CTL^*) ::= E \psi | a | \psi_1 \lor \psi_2 | \neg \psi_1 | X \psi_1 | \psi_1 U \psi_2
\]

Semantics: for a word $\alpha$, a position $i$, and a rooted Kripke structure $(S, s)$:

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

Let $A \psi \overset{\text{def}}{=} \neg E \neg \psi$

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

Examples of formulas

- $\alpha \models GF a$ iff “in $\alpha$, $a$ occurs infinitely often”.
- $\alpha \models XX (b \Rightarrow FC) \text{ iff } \text{If } \alpha(2) = b, \text{ then } \alpha(j) = c \text{ for some } j \geq 2$.
- $\alpha \models F(a \land X(b U a)) \text{ iff } \ldots$ (EXERCISE)

Interpretation over Trees

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

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

\[ \lambda(s_0) \]

\[ \lambda(s_1) \]

\[ \lambda(s_2) \]

\[ S \]

\[ s_0 \]

\[ s_1 \]

\[ s_2 \]

EXERCISE draw the corresponding tree

We make a huge simplification:

we consider only Kripke structures which unravel as full binary trees

but the theory generalizes to arbitrary structures.

Σ-Labeled Full Binary Trees

- The full binary tree is the set \( \{0,1\}^* \) of finite words over a two element alphabet.
- The root is the empty word \( \epsilon \).
- A node is some \( w \in \{0,1\}^* \).
- Every \( w \in \{0,1\}^* \) has two children: a left son \( w_0 \) and a right son \( w_1 \).

**Definition**

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

Trees(Σ) is the set of Σ-labeled full binary trees.

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

\[ 0 \]

\[ 1 \]

\[ a \]

\[ b \]

00

01

10

11

00

01

10

11

Obviously, we will take \( \Sigma = 2^{\text{Prop}} \).

In the example, \( \text{Prop} = \{p\} \), and say \( a = \{p\}, b = \emptyset \).
The (propositional) Mu-calculus

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

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

Smooth Introduction

- Consider the CTL formula $\mathbf{EF}P$ (where $P$ is some proposition): note that
  $$\mathbf{EF}P \equiv P \lor \mathbf{EX} \mathbf{EF}P$$
  so that $\mathbf{EF}P$ is a fixed-point.
- In fact, $\mathbf{EF}P$ is the least fixed-point, e.g. the least such that
  $$Z \equiv P \lor \mathbf{EF}Z$$
- Not all modalities of e.g. CTL are needed as a “basis”

BYO modalities with fixed-point definitions
About lattices and fixed-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.

Tarski-Knaster fixed-point Theorem

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

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

**EXERCISE** What are \(\top\) and \(\bot\)?

**Theorem**

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

1. \(f(x) \leq x\): \(\forall y \in A, x \leq y\), therefore \(f(x) \leq f(y) \leq y\). So \(f(x) \leq \sqcap A = x\).
2. \(x \leq f(x)\): by monotonicity applied to (1), \(f^2(x) \leq f(x)\) so \(f(x) \in A\), and \(x \leq f(x)\).

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

Monotonic Functions

- \(f : L \rightarrow L\) is monotonic (order preserving) if \(\forall x, y \in L, x \leq y \Rightarrow f(x) \leq f(y)\)
- \(x\) is a fixed-point of \(f\) if \(f(x) = x\)
- Define \(f^0\) is the identity function, and \(f^{n+1} = f^n \circ f\).
- Note that \(f\) monotonic implies that \(f^n\) is monotonic. The identity function is monotonic and composing two monotonic functions gives a monotonic function.

Another Characterization of fixed-points

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

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

**EXERCISE** Show it.
Syntax of the Mu-calculus

- An alphabet $\Sigma$, and the associate set of propositions $\text{Prop} = \{P_a\}_{a \in \Sigma}$.
- A infinite set of variables $\text{Var} = \{Z, Z', Y, \ldots\}$.
- Formulas
  \[
  \beta, \beta' \in L_\mu := P_a \mid Z \mid \neg \beta \mid \beta \land \beta' \mid \langle \beta \rangle \mid \langle 1 \rangle \beta \mid \mu Z. \beta
  \]
  where $P_a \in \text{Prop}$, $Z \in \text{Var}$.
- Write $\langle \rangle \beta$ for $(0) \beta \lor (1) \beta$, and $[\ ] \beta$ for $(0) \beta \land (1) \beta$.
- $\beta$ is a sentence if every occurrence of a variable in $\beta$ are bounded by a $\mu$ operator.
- Write $\beta' \leq \beta$ when $\beta'$ is a subformula of $\beta$.
- As $\mu Z. \beta$ is about a least fixed-point (see later for its semantics), we need to ensure its existence, hence the notion of well-formed formulas.

well-formed formulas

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

The meaning of $\mu Z. \beta$

Recall

\[
[\mu Z. \beta]_{val} = \bigcap \{N \in \mathcal{P} \langle \{0, 1\}^* \rangle \mid [\beta]_{\text{val}[N/Z]} \subseteq N\}
\]

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

\[
f : 2^{\{0, 1\}^*} \rightarrow 2^{\{0, 1\}^*}
f(N) = [\beta]_{\text{val}[N/Z]}
\]

where $f$ is monotonic, since $\beta$ is well-formed.

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

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

Notice that if $\beta$ is sentence, then $[\mu Z. \beta]_{\text{val}} = [\mu Z. \beta]_{\text{val}'}$, for any $\text{val}, \text{val}'$; we write it $[\mu Z. \beta]^t$.

Semantics of well-formed formulas

Fix a tree $t \in \text{Trees}(\Sigma)$

- Let $\text{val} : \text{Var} \rightarrow 2^{\{0, 1\}^*}$ be a valuation of the variables. For every $N \subseteq \{0, 1\}^*$, we write $\text{val}[N/Z]$ for $\text{val}'$ defined as $\text{val}$ except that $\text{val}'(Z) = N$.

- Given a tree $t : \{0, 1\}^* \rightarrow \Sigma$, $[\beta]^t_{\text{val}} \subseteq \{0, 1\}^*$ denotes a set of nodes.

Examples of formulas

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

- $\mu Z. Z \equiv \text{false}$
- $\nu Z. Z \equiv \text{true}$
- $\mu Z. P \equiv \nu Z. P \equiv P$
Examples of formulas: about $\text{CTL}$

- What is \( \mu Z.P_a \lor \langle \rangle Z \) ?
- It is equivalent to $\text{EF} a$, whereas $\nu Z.P_a \lor \langle \rangle Z \equiv \text{true}$

\[
\mu Z.P_a \lor \langle \rangle Z \equiv P_a \lor \langle \rangle (\mu Z.P_a \lor \langle \rangle Z) \\
\quad \quad \equiv P_a \lor \langle \rangle (P_a \lor \langle \rangle (\mu Z.P_a \lor \langle \rangle Z)) \\
\quad \quad \equiv P_a \lor \langle \rangle (P_a \lor \langle \rangle (\mu Z.P_a \lor \langle \rangle Z)) \\
\quad \quad \equiv \ldots
\]

A node \( w \in \mu Z.P_a \lor \langle \rangle Z \) if either it is in \( P_a \) or it has a child who is in \( P_a \) 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 $\text{EF} a$.

Positive normal form

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

- Notice that \( \neg(d) \beta = (d) \neg \beta \), for \( d \in \{0, 1\} \).

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

Alternation Depth ($\pm 1$ in the literature)

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

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

Example: $ad(\nu Y. (\mu Z.P_a \lor \langle \rangle Z \land \langle \rangle Y)) = 2$
The Mu-calculus

Some important results

Write $L^k_\mu = \{ \beta \in L_\mu \mid ad(\beta) \leq k \}$.
- CTL $\subseteq L^1_\mu$, and this is strict (recall $\nu Z. P_a \land [ ] [ ] Z$ is not expressible in CTL$^*$).
- $ad(\nu Y. \mu Z. (\langle \rangle Y \land P_a \lor Z)) = 2$, then $\text{EGF}a$ is in $L^2_\mu$.

Theorem

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

Theorem

[BGL07] The variable hierarchy of the $\mu$-calculus is strict.

Model-checking and Satisfiability

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

Definition (informal)

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

What next?

- Tree Automata to recognize certain trees:
  $\beta \in L_\mu \leadsto A_\beta$ such that $L(A_\beta) = \{ t \in \text{Trees}(\Sigma) \mid t \models \beta \}$
- The Model-checking Problem $\leadsto$ The Membership Problem
- The Satisfiability Problem $\leadsto$ The Emptiness Problem

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

Automata on Infinite Objects
Automata on Infinite Objects

Automata on Infinite Objects [Rab69], [GH82, Mul84, EJ91], [GTW02, Chap. 8 and 9]

- Acceptance conditions: Büchi, Muller, Rabin and Streett, Parity on every branch of the run of the automaton on its input.
- Runs are trees, and accepting runs fulfill the acceptance condition.
- We consider parity acceptance condition.

Also ω-automata are automata on infinite words [Büc62, McN66], [Tho90], [GTW02, Chap. 1]

- Acceptance conditions: Büchi, Muller, Rabin and Streett, Parity
- Runs are paths, accepting runs fulfill the accepting condition.
- All coincide with ω-regular languages \( L = \bigcup_i K_i \alpha_i \) – deterministic Büchi are weaker.
- Connection with Logic LTL: LTL corresponds to FOL as well as star-free ω-regular languages.

Non-deterministic Parity Tree Automata

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

Definition

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

- \( Q(\ni q^0) \) is a finite set of states \( (q^0 \text{ the initial state}) \)
- \( \delta \subseteq Q \times \Sigma \times Q \times Q \) 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 \( \mathcal{A} \)

Example

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

\[
\begin{align*}
\delta(q_a, a) &= \{(\top, \top)\} \\
\delta(q_a, b) &= \{(q_a, q_a)\} \\
\delta(\top, a) &= \{(\top, \top)\} \\
\delta(\top, b) &= \{(\top, T)\}
\end{align*}
\]

with \( c(q_a) = 1 \) and \( c(\top) = 0 \).
The parity acceptance condition

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

Example 1

- Let $L_0$ be the set of trees the branches of which all contain an $a$. This may be expressed in $L_\mu$ as $\mu Z.P_a \lor [] Z$ in $L_\mu$.
- $L_0$ may be characterized by the following tree automaton
  $$\delta(q_a, a) = \{(T, T), (T, q_a)\}$$
  $$\delta(q_a, b) = \{(q_a, q_a)\}$$
  $$\delta(T, a) = \{(T, T)\}$$
  $$\delta(T, b) = \{(T, T)\}$$

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

Example 2

Tree automata are nondeterministic, and cannot be determinized in general.
- Let $L_0^\omega \subseteq \text{Trees}(\{a, b\})$ be the set of trees having a branch with infinitely many $a$'s.
- Consider the automaton with states $q_a, q_b, T$ and transitions ($*$ stands for either $a$ or $b$).
  $$\delta(q_a, a) = \{(q_a, T), (T, q_a)\}$$
  $$\delta(q_a, b) = \{(q_b, T), (T, q_b)\}$$
  $$\delta(T, *) = \{(T, T)\}$$

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

Example 2 (Cont.)

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


Acceptance

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

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

Other Acceptance Conditions

- Büchi is specified by a set $F \subseteq Q$
  $$\text{Acc} = \{ \gamma \mid \text{Inf}(\gamma) \cap F \neq \emptyset \}$$
- Muller is specified by a set $\mathcal{F} \subseteq \mathcal{P}(Q)$,
  $$\text{Acc} = \{ \gamma \mid \text{Inf}(\gamma) \in \mathcal{F} \}$$
- Rabin is specified by a set $\{(R_1, G_1), \ldots, (R_k, G_k)\}$ where $R_i, G_j \subseteq Q$,
  $$\text{Acc} = \{ \gamma \mid \forall i, \text{Inf}(\gamma) \cap R_i = \emptyset \text{ and } \text{Inf}(\gamma) \cap G_i \neq \emptyset \}$$
- Streett is specified by a set $\{(R_1, G_1), \ldots, (R_k, G_k)\}$ where $R_i, G_j \subseteq Q$,
  $$\text{Acc} = \{ \gamma \mid \forall i, \text{Inf}(\gamma) \cap R_i = \emptyset \text{ or } \text{Inf}(\gamma) \cap G_i \neq \emptyset \}$$

For the relationship between these conditions see [GTW02].

- Büchi tree automata are less expressive than the other acceptance conditions (which are equivalent) [Rab70]: for example, the complement of $L_{a}^{\infty}$, that is finitely many $a$’s on each branch, cannot be characterized by any Büchi tree automaton.

Regular Tree Languages and Properties

- A tree language $L \subseteq \text{Trees}(\Sigma)$ is regular iff there exists a parity tree automaton which recognizes $L$.
- Tree automata are closed under sum, projection, and complementation.
  - Tree automata cannot be determinized: $L_{a}^{3} \subseteq \text{Trees}(\{a, b\})$, the language of trees having one node labeled by $a$, is not recognizable by a deterministic tree automata (with any of the considered acceptance conditions).
  - The proof for complementation uses the determinization result for word automata. Difficult proof [GTW02, Chap. 8]. [Rab70]
- We will solve the Membership Problem and the Emptiness Problem for (nondeterministic) automata by using Parity Games.
(Parity) Games

- 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

- Formally, a play in the arena $G$ is either
  - an infinite path $\pi = v_0v_1v_2 \ldots \in V^\omega$ with $v_{i+1} \in v_iE$ for all $i \in \omega$, or
  - a finite path $\pi = v_0v_1v_2 \ldots v_l \in V^+$ with $v_{i+1} \in v_iE$ for all $i < l$, but $v_l E = \emptyset$.

Games and Winning sets

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

Parity Winning Conditions

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

Formally

- We color vertices of the arena by $\chi : V \rightarrow C$ 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 $\text{Inf}_C(\pi)$ be the set of colors that occurs infinitely often in $\chi(\pi)$.

$Win$ is the set of infinite paths $\pi$ such that $\text{min}(\text{Inf}_C(\pi))$ is even.
### Parity Games

#### Example of a parity game

A strategy for Player $\sigma$ is a function $f_\sigma : V^* V_\sigma \to V$.

A prefix play $\pi = v_0 v_1 v_2 \ldots v_l$ is conform with $f_\sigma$ if for every $i$ with $0 \leq i < l$ and $v_i \in V_\sigma$ the function $f_\sigma$ is defined and we have $v_{i+1} = f_\sigma(v_0 \ldots v_i)$.

A play is conform with $f_\sigma$ if each of its prefix is conform with $f_\sigma$.

The winning region for Player $\sigma$ is the set $W_\sigma(G) \subseteq V$ of all vertices such that Player $\sigma$ wins $(G, v)$ (to be defined rigorously).

#### Example of Winning Regions

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

**Theorem**

Every parity game is determined.

A strategy $f_\sigma$ is a positional (or memoryless) strategy whenever $f_\sigma(\pi v) = f_\sigma(\pi' v), \forall v \in V_\sigma$.

**Theorem**

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

See [GTW02, Chaps. 6 and 7]
Complexity Results

**Theorem**

\[
\text{WINS} = \{(G, v) \mid G \text{ a finite parity game and } v \text{ a winning position of Player 0} \}
\text{is in NP} \cap \text{co-NP}
\]

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

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

**EXERCISE** How would you solve a safety game?

---

**Back to Decision Problems for ND Tree Automata**

**The Membership Problem:** \( A \sim G_{A,t} \)

- Given a tree \( t \) and an NDPT automaton \( A \), we build a parity game \((G_{A,t}, v)\) s.t. \( v \) is in \( W_0(G_{A,t}) \) iff \( t \in L(A) \).

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

**The Emptiness Problem:** \( A \sim A' \sim G_{A'} \)

- For each parity automaton \( A \), we build an Input Free automaton \( A' \) such that \( L(A) \neq \emptyset \) iff \( A' \) admits a successful run.

- From \( A' \) we build a parity game \( G_{A'} \) such that (winning) strategies of Player 0 and (successful) runs of \( A' \) correspond.

Both problem reduce to solving parity games!

---

**The Game Graph** \( G_{A,t} \)

0-positions are of the form \((w, t(w), q)\).

- Moves from \((w, t(w), q)\), with \( \delta(q, t(w)) = \{(q'_1, q_1'), (q'_2, q'_2), \ldots (q'_m, q'_m)\}\) are:

\[
(w, t(w), (q, t(w), q'_1, q_1'))
\]

\[
(w, t(w), (q, t(w), q'_2, q'_2))
\]

\[
\ldots
\]

\[
(w, t(w), (q, t(w), q'_m, q'_m))
\]

- Player 0 chooses the transition \((q, t(w), q', q'')\) from \( q \) for input \( t(w) \)

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 $G_{A,t}$

$A = (Q, \Sigma, q^0, \delta, c)$
- $V_0$ 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 Finite Game with a Regular Tree

With the automaton:

$\delta(q_a, a) = \{(q_a, T), (T, q_a)\}$
$\delta(q_b, b) = \{(q_b, T), (T, q_b)\}$
$\delta(T, \epsilon) = \{(T, T)\}$
$c(q_a) = c(T) = 0$
$c(q_b) = 1$

Example of $G_{A,t}$
The Emptiness Problem of NDTA

We need the notion of input-free automata.

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

**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 $(q', x, p, p') \in \delta$ for some $p, q, q' \in Q$
- $c'(q, a) = c(q)$

From IF Automata to Parity Games

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

- Positions $V_0 = Q$ and $V_1 = \emptyset$
- 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.
Decidability of Emptiness for NDPT Automata

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

\[ A \leadsto A' \leadsto G_{A'}, \] and combined previous results.

**Complexity Issues**

**Corollary**
The Emptiness Problem for NDPT automata is in \( NP \cap co-NP \).

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

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

Finite Model Property

**Corollary**
If \( L(A) \neq \emptyset \) then \( L(A) \) contains a regular tree.

Use the memoryless winning strategy in \( G_{A'} \).

Formally, take \( A \) and its corresponding IF automaton \( A' \). Assume a successful run of \( A' \) and a memoryless strategy \( f \) for Player 0 in \( G_{A'} \) from some position \((q_I, a)\).

The subgraph \( G_{A'} \) induces a deterministic IF automaton \( A'' \) (without acc): extract the transitions out of \( G_{A'} \) from positions in \( V_1 \). \( A'' \) is a subautomaton of \( A' \).

\( A'' \) generates a regular tree \( t \) in the second component of its states. Now, \( t \in L(A) \) because \( A' \) behaves like \( A \).

Concluding remarks

What we have seen

- Binary trees as a simplified setting to represent system’s executions.
- Propositional \( \mu \)-calculus that subsumes all branching-time temporal logics (LTL, CTL, CTL*, PDL, ...).
- Non-deterministic tree automata (NDTA) to recognize regular tree languages.
- (Parity) games as abstract mathematical tools to, e.g. check emptiness and membership problems for NDTA.

\[ \Rightarrow \] The emptiness problem for NDTA is in \( NP \cap co-NP \).

\[ \Rightarrow \] Memoryless strategies deliver regular objects.

In particular, NDTA have the finite model property.
Concluding remarks

What we have not seen

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

Sophie Pinchinat (IRISA)
Logic, Automata, and Games
M2RI 2011-2012 75 / 75

References

- A. Arnold.
  The mu-calculus alternation-depth hierarchy is strict on binary trees.

- Dietmar Berwanger, Erich Grädel, and Giacomo Lenzi.
  The variable hierarchy of the μ-calculus is strict.

  Permissive strategies: from parity games to safety games.

- J. C. Bradfield.
  The modal mu-calculus alternation hierarchy is strict.

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

- R. McNaughton.

Y. Gurevich and L. Harrington.
Trees, automata, and games.

E. Grädel, W. Thomas, and T. Wilke, editors.

Giacomo Lenzi.
A hierarchy theorem for the μ-calculus.

E. A. Emerson.
Temporal and modal logic.

E. A. Emerson and J. Y. Halpern.
“Sometimes” and “Not Never” revisited: On branching versus linear time.

E. A. Emerson and C. S. Jutla.
Tree automata, mu-calculus and determinacy.

E. A. Emerson.

E. Grädel.
The μ-calculus alternation-depth hierarchy is strict on binary trees.

A. W. Mostowski.
Games with forbidden positions.

R. McNaughton and S. Papert.
*Counter-Free Automata.*

D. 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.

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

M. O. Rabin.

M. O. Rabin.
Weakly definable relations and special automata.

A. Tarski.
A lattice-theoretical fixpoint theorem and its applications.

W. Thomas.
Automata on infinite objects.

P. Wolper.
Temporal logic can be more expressive.