Tree Automata

http://list.mpri.master.univ-paris7.fr/wws/info/cours-1-18

cours-1-18@mpri.master.univ-paris7.fr

Florent Jacquemard
Ștefan Ciobâcă

INRIA Saclay & LSV (CNRS/ENS Cachan)

florent.jacquemard@inria.fr
ciobaca@lsv.ens-cachan.fr

TATA book http://tata.gforge.inria.fr

(chapters 1, 2, 3, 7, 8)

Tree Automata Techniques and Applications

Hubert Comon Max Dauchet Rémi Gilleron
Florent Jacquemard Denis Lugiez Christof Löding
Sophie Tison Marc Tommasi

Part I

Motivations and Plan

Plan

Logic

example:

$$t \models \forall x\ a(x) \Rightarrow \exists y\ y > x \wedge b(y)$$

- automata = semantic tools for deciding logics (e.g. monadic second order logic).
- caracterization of tree models with automata = "compilation" formula $\rightarrow$ automaton
- decision of the satisfiability $\equiv$ emptiness decision for the automaton

Automated deduction

trees = terms in first order logic (Herbrand models)

Automatization of inductive reasoning, inductive reducibility

Example :

axioms: $0 + x = x$, $s(x) + y = s(x + y)$.
$x + 0 = x$ is an inductive theorem: it is valid in the initial model

- domain of the initial model: $0, s(0), s(s(0)), \ldots$
- $=$ irreducible terms (normal forms)
- terms generated by the tree grammar $\mathsf{N} := 0 \mid s(\mathsf{N})$.

$x + y$ is inductively reducible.

Automated deduction

Example :

axioms: $p(s(x)) = x, \quad s(p(x)) = x$
$0 + x = x, \quad s(x) + y = s(x + y), \quad p(x) + y = p(x + y).$
$x + 0 = x$ is an inductive theorem.

Normal forms = domain of the initial model, generated by:

$$\begin{array}{lcl} \mathsf{N}_0 & := & 0, \\ \mathsf{N}_s & := & s(\mathsf{N}_0) \mid s(\mathsf{N}_s), \\ \mathsf{N}_p & := & s(\mathsf{N}_0) \mid s(\mathsf{N}_p). \end{array}$$

Evaluation Strategies

Evaluation in functional programming languages

Strategies for reduction by rewriting

- terms in normal form (irreducible)
- terms normalizable
- existence of needed redex

Example :

$\vee(\top, x_2) = \top$, $\vee(x_1, \top) = \top$: no needed redex.
$\exists x_1, \forall x_2, \vee(x_1, x_2) = \top$ (evaluation of x_2 not necessary)
$\exists x_2, \forall x_1, \vee(x_1, x_2) = \top$ (evaluation of x_1 not necessary)

test: existence of a $\vee(x_1, \Omega)$ and a $\vee(\Omega, x_2)$ evaluated to $\top$.
with the construction of a tree automaton and emptiness decision.

tous ces langages sont reconnaissable par TA.

Program Verification

Regular Model Checking:
Model Checking techniques for infinite state systems.
static analysis of safety properties using symbolic reachability analysis techniques.

Abstract models

- Pushdown systems for sequential programs with procedure calls
- Petri nets for multi-threaded programs (without procedure calls)

Generalization: process algebra

$$p ::= 0 \mid X \mid p \cdot p \mid p \| p$$

Program verification

$$p ::= 0 \mid X \mid p \cdot p \mid p \| p$$

- 0: null process (termination)
- X: program point
- $p \cdot p$: sequential composition
- $p \| p$: parallel composition

Transition rules:

- procedure call: $X \to Y \cdot Z$ ($Z =$ return point)
- conditional continuation: $Y_i \cdot Z \to t_i$
- dynamic thread creation: $X \to Y \| Z$ ($Z =$ return point)
- handshake : $X \| Y \to X' \| Y'$

actually $\cdot$ is modulo A and $\|$ modulo $AC \Rightarrow$ unranked tree model

Verification

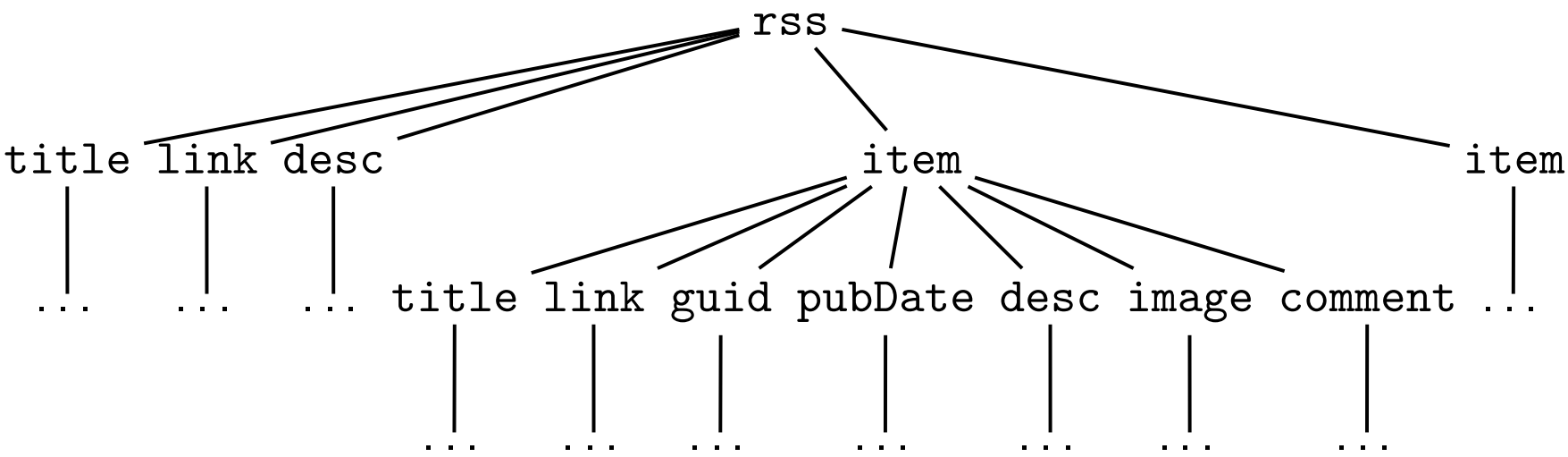
Same technique applies to the verification of other infinite-states systems.

- configuration / state = tree
  - process,
  - message exchanged in a protocole,
  - local network with a tree shape,
  - tree data structure in memory, with pointers (e.g. binary search trees)...
- set of configurations = tree language L
- transition relation between configurations (post)
- safety: $\text{post}^*(L_{\text{init}}) \cap L_{\text{error}} = \emptyset$.

Web data (XML Document)

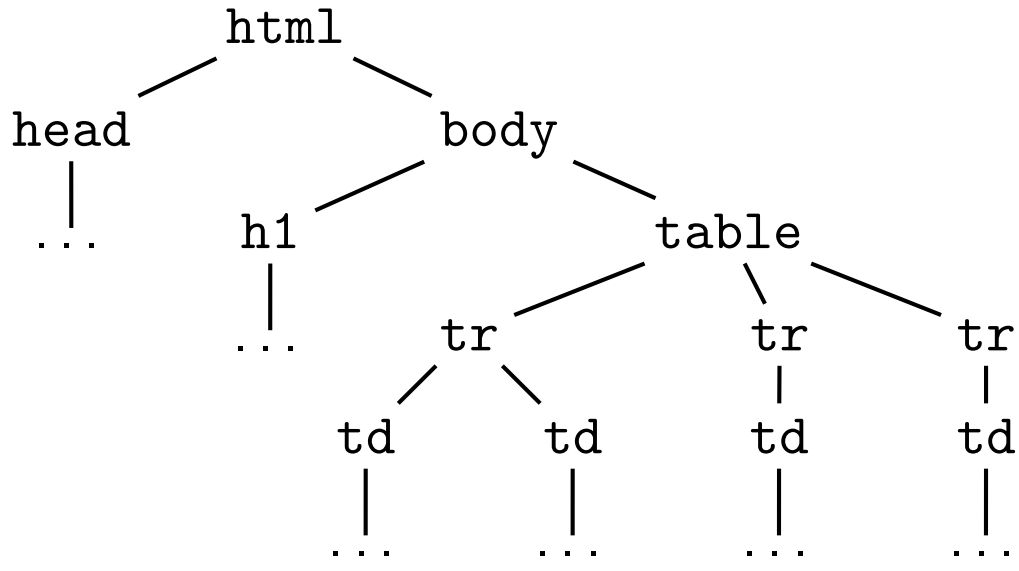
```
<rss version="2.0">
    <title>My blog</title>
    <link>http://myblog.blogspot.com</link>
    <description>bla bla bla</description>
    <item>
      <title>Concert</title>
      <link>http://myblog.blogspot.com/me/Mon blog/...</link>
      <guid>5f7da0aa-a593-4a2e</guid>
      <pubDate>Fri, 21 Mar 2009 14:40:02 +0100</pubDate>
      <description>...</description>
      <image href="..."></image>
      <comment link="..." count="0" enabled="0">...</comment>
    </item>
    <item>
      <title>Journée de surf</title>
      ...
    </item>
</rss>
```

Web data


rss
title
link
desc
item
item
...
...
...
title
link
guid
pubDate
desc
image
comment
...
...
...
...
...
...
...
...

HTML Document

```
<html>
 <head>...</head>
 <body>
  <h1>...</h1>
  <table>
   <tr>
    <td>...</td>
    <td>...</td>
   </tr>
   <tr>
    <td>...</td>
   </tr>
   <tr>
    <td>...</td>
   </tr>
  </table>
 </body>
</html>
```



XML Document Processing

documents = unranked trees

conformity

- class of documents with a predefined structure (valid documents)

  ex:

```
<table>
  <tr> <td> c11 </td> <td> c12 </td> </tr>
  <tr> <td> c21 </td> <td> c22 </td> </tr>
</table>
```

- defined by a schema (DTD, XML schema...) = tree language
- All the schema formalisms in use currently correspond to tree automata.

XML Documents

ranked	unranked
tree automata	schemas (DTD, XML schema, Relax NG)
membership	validation
emptiness	query satisfiability
inclusion	schema entailment
tree transducers, rewrite systems	transformation languages (XSLT)
rewrite closure	type inference

Plan

Trees

- finite ranked trees (terms in first order logic)
- finite unranked ordered trees
- finite unranked unordered trees
- infinite trees...

$\Rightarrow$ several classes of tree automata.

Properties

- determinism,
- Boolean closures,
- closures under transformations
  (homomorphismes, transducers, rewrite systems...)
- minimization,
- decision problems, complexity,
  - membership,
  - emptiness,
  - universality,
  - inclusion, equivalence,
  - emptiness of intersection,
  - finiteness...
- pumping and star lemma,
- expressiveness, correspondence with logics.

Plan

1. finite ranked tree automata
2. correspondence with the monadic second order logic of the tree (Thatcher and Wright's theorem).
3. finite unranked ordered tree automata = Hedge Automata
   Application to XML document processing.
4. finite unranked unordered tree automata
   = Presburger automata

- alternating and 2 ways automata (on words and trees)
- representation of tree automata as Horn clause sets
- infinite word automata (conditions of Müller and Büchi), infinite tree automata (Rabin automata).

Part II

Finite Ranked Trees

Terms (in first order logic)

Plan

Terms and Rewriting

Definitions and Expressiveness

Determinism and Boolean Closures

Decision Problems

Minimization

Tree Transformations

Signature

Definition : Signature

A signature Σ is a finite set of function symbols each of them with an arity greater or equal to 0.

We denote Σ_i the set of symbols of arity i.

Example :

$\{+ : 2, s : 1, 0 : 0\}$, $\{\wedge : 2, \vee : 2, \neg : 1, \top, \bot : 0\}$.

We also consider a countable set $\mathcal{X}$ of variable symbols.

Terms

Definition : Term

The set of terms over the signature Σ and $\mathcal{X}$ is the smallest set $\mathcal{T}(\Sigma, \mathcal{X})$ such that:

- $\Sigma_0 \subseteq \mathcal{T}(\Sigma, \mathcal{X})$,
- $\mathcal{X} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$,
- if $f \in \Sigma_n$ and if $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$, then $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})$.

The set of ground terms (terms without variables, i.e. $\mathcal{T}(\Sigma, \emptyset)$) is denoted $\mathcal{T}(\Sigma)$.

Example :

x, $\neg(x)$, $\wedge\big(\vee(x, \neg(y)), \neg(x)\big)$.

Terms (2)

A term where each variable appears at most once is called linear.
A term without variable is called ground.

Depth $\mathsf{h}(t)$:

- $\mathsf{h}(a) = \mathsf{h}(x) = 0$ if $a \in \Sigma_0$, $x \in \mathcal{X}$,
- $\mathsf{h}\big(f(t_1, \ldots, t_n)\big) = \max\{\mathsf{h}(t_1), \ldots, \mathsf{h}(t_n)\} + 1$.

Positions

A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ can also be seen as a function from the set of its positions $\mathcal{P}os(t)$ into $\Sigma \cup \mathcal{X}$.
The empty position (root) is denoted ε.

$\mathcal{P}os(t)$ is a subset of $\mathbb{N}^*$ satisfying the following properties:

- $\mathcal{P}os(t)$ is closed under prefix,
- for all $p \in \mathcal{P}os(t)$ such that $t(p) \in \Sigma_n$ $(n \geq 1)$, $\{pj \in \mathcal{P}os(t) \mid j \in \mathbb{N}\} = \{p1, ..., pn\}$,
- every $p \in \mathcal{P}os(t)$ such that $t(p) \in \Sigma_0 \cup \mathcal{X}$ is maximal in $\mathcal{P}os(t)$ for the prefix ordering.

The size of t is defined by $\|t\| = |\mathcal{P}os(t)|$.

Subterm $t|_p$ at position $p \in \mathcal{P}os(t)$:

- $t|_\varepsilon = t$,
- $f(t_1, \ldots, t_n)|_{ip} = t_i|_p$.

The replacement in t of $t|_p$ by s is denoted $t[s]_p$.

Positions (example)

Example :

$t = \wedge(\wedge(x, \vee(x, \neg(y))), \neg(x))$,
$t|_{11} = x$, $t|_{12} = \vee(x, \neg(y))$, $t|_2 = \neg(x)$,
$t[\neg(y)]_{11} = \wedge(\wedge(\neg(y), \vee(x, \neg(y))), \neg(x))$.

Substitutions, Contexts

Definition : Substitution

A *substitution* is a function of finite domain from $\mathcal{X}$ into $\mathcal{T}(\Sigma, \mathcal{X})$. We extend the definition to $\mathcal{T}(\Sigma, \mathcal{X}) \to \mathcal{T}(\Sigma, \mathcal{X})$ by:

$$f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma) \quad (n \geq 0)$$

Definition : Contexte

A *context* is a linear term.

The application of a context $C \in \mathcal{T}(\Sigma, \{x_1, \ldots, x_n\})$ to n terms $t_1, \ldots, t_n$, denoted $C[t_1, \ldots, t_n]$, is $C\sigma$ with $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

Term Rewriting

A *rewrite system* $\mathcal{R}$ is a finite set of rewrite rules of the form $\ell \to r$ with $\ell, r \in \mathcal{T}(\Sigma, \mathcal{X})$.

The relation $\xrightarrow[\mathcal{R}]{}$ is the smallest binary relation containing $\mathcal{R}$, and closed under application of contexts and substitutions.
i.e. $s \xrightarrow[\mathcal{R}]{} t$ iff $\exists p \in \mathcal{P}os(s), \ell \to r \in \mathcal{R}, \sigma,\ s|_p = \ell\sigma$ and $t = s[r\sigma]_p$.

We note $\xrightarrow[\mathcal{R}]{*}$ the reflexive and transitive closure of $\xrightarrow[\mathcal{R}]{}$.

Example :

$\mathcal{R} = \{+(0, x) \to x, +(s(x), y) \to s(+(x, y))\}$.

$$
\begin{array}{lll}
+\big(s(s(0)), +(0, s(0))\big) & \xrightarrow[\mathcal{R}]{} & +\big(s(s(0)), s(0)\big) \\
& \xrightarrow[\mathcal{R}]{} & s\big(+(s(0), s(0))\big) \\
& \xrightarrow[\mathcal{R}]{} & s\big(s\big(+(0, s(0))\big)\big) \\
& \xrightarrow[\mathcal{R}]{} & s(s(s(0)))
\end{array}
$$

Plan

Terms and Rewriting

Definitions and Expressiveness

Determinism and Boolean Closures

Decision Problems

Minimization

Tree Transformations

Bottom-up Finite Tree Automata

Definition : Tree Automata

A *tree automaton* (TA) over a signature Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ where Q is a finite set of *states*, $Q^{\mathsf{f}} \subseteq Q$ is the subset of final states and Δ is a set of transition rules of the form: $f(q_1, \ldots, q_n) \to q$ with $f \in \Sigma_n$ $(n \geq 0)$ and $q_1, \ldots, q_n, q \in Q$.

The state q is called the head of the rule.

We say that a ground term $t \in \mathcal{T}(\Sigma)$ is accepted by $\mathcal{A}$ in the state q iff $t \xrightarrow[\Delta]{*} q$.

The language of $\mathcal{A}$ in the state q is
$L(\mathcal{A}, q) := \left\{ t \in \mathcal{T}(\Sigma) \;\middle|\; t \xrightarrow[\Delta]{*} q \right\}$.

The language of $\mathcal{A}$ is $L(\mathcal{A}) := \bigcup_{q^{\mathsf{f}} \in Q^{\mathsf{f}}} L(\mathcal{A}, q^{\mathsf{f}})$ (regular language).

Tree Automata: example 1

Example :

$\Sigma = \{\wedge : 2, \vee : 2, \neg : 1, \top, \bot : 0\}$,

$$\mathcal{A} = \left(\Sigma, \{q_0, q_1\}, \{q_1\}, \left\{ \begin{array}{rclrcl} \bot & \to & q_0 & \top & \to & q_1 \\ \neg(q_0) & \to & q_1 & \neg(q_1) & \to & q_0 \\ \vee(q_0, q_0) & \to & q_0 & \vee(q_0, q_1) & \to & q_1 \\ \vee(q_1, q_0) & \to & q_1 & \vee(q_1, q_1) & \to & q_1 \\ \wedge(q_0, q_0) & \to & q_0 & \wedge(q_0, q_1) & \to & q_0 \\ \wedge(q_1, q_0) & \to & q_0 & \wedge(q_1, q_1) & \to & q_1 \end{array} \right\} \right)$$

$$\begin{array}{ll} & \wedge(\wedge(\top, \vee(\top, \neg(\bot))), \neg(\top)) \xrightarrow[\mathcal{A}]{} \wedge(\wedge(\top, \vee(\top, \neg(\bot))), \neg(q_1)) \\ \xrightarrow[\mathcal{A}]{} & \wedge(\wedge(q_1, \vee(q_1, \neg(q_0))), \neg(q_1)) \xrightarrow[\mathcal{A}]{} \wedge(\wedge(q_1, \vee(q_1, \neg(q_0))), q_0) \\ \xrightarrow[\mathcal{A}]{} & \wedge(\wedge(q_1, \vee(q_1, q_1)), q_0) \xrightarrow[\mathcal{A}]{} \wedge(\wedge(q_1, q_1), q_0) \xrightarrow[\mathcal{A}]{} \wedge(q_1, q_0) \xrightarrow[\mathcal{A}]{} q_0 \end{array}$$

Tree Automata: example 2

Example :

$\Sigma = \{\wedge : 2, \vee : 2, \neg : 1, \top, \bot : 0\}$,
TA recognizing the ground instances of $\neg(\neg(x))$:

$$\mathcal{A} = \left(\Sigma, \{q, q_\neg, q_\mathsf{f}\}, \{q_\mathsf{f}\}, \left\{ \begin{array}{rclrcl} \bot & \to & q & \top & \to & q \\ \neg(q) & \to & q & \neg(q) & \to & q_\neg \\ \neg(q_\neg) & \to & q_\mathsf{f} & & & \\ \vee(q,q) & \to & q & \wedge(q,q) & \to & q \end{array} \right\} \right)$$

Example :

Ground terms embedding the pattern $\neg(\neg(x))$: $\mathcal{A} \cup \{\neg(q_\mathsf{f}) \to q_\mathsf{f}, \vee(q_\mathsf{f}, q_*) \to q_\mathsf{f}, \vee(q_*, q_\mathsf{f}) \to q_\mathsf{f}, \ldots\}$ (propagation of q_f).

Linear Pattern Matching

Proposition :

Given a linear term $t \in \mathcal{T}(\Sigma, \mathcal{X})$, there exists a TA $\mathcal{A}$ recognizing the set of ground instances of t: $L(\mathcal{A}) = \left\{ t\sigma \;\middle|\; \sigma : \mathcal{X} \to \mathcal{T}(\Sigma) \right\}$.

Regular Langages

Other definition (recursive) of $L(\mathcal{A}, q)$:

$$\begin{array}{rcl} L(\mathcal{A}, q) & = & \left\{ a \in \Sigma_0 \;\middle|\; a \to q \in \Delta \right\} \\ & \cup & \displaystyle\bigcup_{f(q_1,\ldots,q_n)\to q \in \Delta} f\big(L(\mathcal{A}, q_1), \ldots, L(\mathcal{A}, q_n)\big) \end{array}$$

with $f(L_1, \ldots, L_n) := \left\{ f(t_1, \ldots, t_n) \;\middle|\; t_1 \in L_1, \ldots, t_n \in L_n \right\}$.

Runs

Definition : Run

A *run* of a TA $(\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ on a term $t \in \mathcal{T}(\Sigma)$ is a function $r : \mathcal{P}os(t) \to Q$ such that for all $p \in \mathcal{P}os(t)$, if $t(p) = f \in \Sigma_n$, $r(p) = q$ and $r(pi) = q_i$ for all $1 \leq i \leq n$, then $f(q_1, \ldots, q_n) \to q \in \Delta$.

The run r is *accepting* if $r(\varepsilon) \in Q^{\mathsf{f}}$.
$L(\mathcal{A})$ is the set of ground terms of $\mathcal{T}(\Sigma)$ for which there exists an accepting run.

Pumping Lemma

Lemma : Pumping Lemma

Let $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$.
$L(\mathcal{A}) \neq \emptyset$ iff there exists $t \in L(\mathcal{A})$ such that $h(t) \leq |Q|$.

Lemma : Iteration Lemma

For all TA $\mathcal{A}$, there exists $k > 0$ such that for all term $t \in L(\mathcal{A})$ with $h(t) > k$, there exists 2 contexts $C, D \in \mathcal{T}(\Sigma, \{x_1\})$ with $D \neq x_1$ and a term $u \in \mathcal{T}(\Sigma)$ such that $t = C\big[D[u]\big]$ and for all $n \geq 0$, $C\big[D^n[u]\big] \in L(\mathcal{A})$.

usage: to show that a language is not regular.

Non Regular Languages

We show with the Pumping and Iteration lemmatas that the following tree languages are not regular:

- $\{f(t,t) \mid t \in \mathcal{T}(\Sigma)\}$,
- $\{f(g^n(a), h^n(a)) \mid n \geq 0\}$,
- $\{t \in \mathcal{T}(\Sigma) \mid |\mathcal{P}os(t)| \text{ is prime}\}$.

Epsilon-transitions

We extend the class TA into TAε with the addition of another type of transition rules of the form $q \xrightarrow{\varepsilon} q'$ (ε-transition).
with the same expressiveness as TA.

Proposition : Suppression of ε-transitions

For all TAε $\mathcal{A}_\varepsilon$, there exists a TA (without ε-transition) $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}_\varepsilon)$. The size of $\mathcal{A}$ is polynomial in the size of $\mathcal{A}_\varepsilon$.

pr.: We start with $\mathcal{A}_\varepsilon$ and we add $f(q_1, \ldots, q_n) \to q'$ if there exists $f(q_1, \ldots, q_n) \to q$ and $q \xrightarrow{\varepsilon} q'$.

Top-Down Tree Automata

Definition : Top-Down Tree Automata

A top-down tree automaton over a signature Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{init}}, \Delta)$ where Q is a finite set of *states*, $Q^{\mathsf{init}} \subseteq Q$ is the subset of initial states and Δ is a set of transition rules of the form: $q \to f(q_1, \ldots, q_n)$ with $f \in \Sigma_n$ $(n \geq 0)$ and $q_1, \ldots, q_n, q \in Q$.

A ground term $t \in \mathcal{T}(\Sigma)$ is accepted by $\mathcal{A}$ in the state q iff $q \xrightarrow[\Delta]{*} t$.

The language of $\mathcal{A}$ starting from the state q is
$L(\mathcal{A}, q) := \left\{ t \in \mathcal{T}(\Sigma) \;\middle|\; q \xrightarrow[\Delta]{*} t \right\}$.

The language of $\mathcal{A}$ is $L(\mathcal{A}) := \bigcup_{q^{\mathsf{i}} \in Q^{\mathsf{init}}} L(Q, q^{\mathsf{i}})$.

Top-Down Tree Automata (expressiveness)

Proposition : Expressiveness

The set of top-down tree automata languages is exactly the set of regular tree languages.

Remark: Notations

In the next slides of this chapter,

$$\text{TA} = \text{Bottom-Up Tree Automata}$$

Plan

Terms and Rewriting

Definitions and Expressiveness

Determinism and Boolean Closures

Decision Problems

Minimization

Tree Transformations

Determinism

Definition : Determinism

A TA $\mathcal{A}$ is *deterministic* if for all $f \in \Sigma_n$, for all states $q_1, \ldots, q_n$ of $\mathcal{A}$, there is at most one state q of $\mathcal{A}$ such that $\mathcal{A}$ contains a transition $f(q_1, \ldots, q_n) \to q$.

If $\mathcal{A}$ is deterministic, then for all $t \in \mathcal{T}(\Sigma)$, there exists at most one state q of $\mathcal{A}$ such that $t \in L(\mathcal{A}, q)$. It is denoted $\mathcal{A}(t)$ or $\Delta(t)$.

Completeness

Definition : Completeness

A TA $\mathcal{A}$ is *complete* if for all $f \in \Sigma_n$, for all states $q_1, \ldots, q_n$ of $\mathcal{A}$, there is at least one state q of $\mathcal{A}$ such that $\mathcal{A}$ contains a transition $f(q_1, \ldots, q_n) \to q$.

If $\mathcal{A}$ is complete, then for all $t \in \mathcal{T}(\Sigma)$, there exists at least one state q of $\mathcal{A}$ such that $t \in L(\mathcal{A}, q)$.

Completion

Proposition : Completion

For all TA $\mathcal{A}$, there exists a complete TA $\mathcal{A}_c$ such that $L(\mathcal{A}_c) = L(\mathcal{A})$. Moreover, if $\mathcal{A}$ is deterministic, then $\mathcal{A}_c$ is deterministic. The size of $\mathcal{A}_c$ is polynomial in the size of $\mathcal{A}$, its construction is PTIME.

pr.: add a trash state $q_\perp$.

add a trash state $q_\perp$ for missing left members + propagation of $q_\perp$: $f(q_*, \ldots, q_*, q_\perp, q_*, \ldots, q_*) \to q_\perp$, $q_* \in Q \cup \{q_\perp\}$.

Lemma

for all $t \in \mathcal{T}(\Sigma)$, $t \in L(\mathcal{A})$ iff $t \in L(\mathcal{A}_c)$.

pr.:

$\Rightarrow$ is immediate

$\Leftarrow$ by contradiction, an accepting run r of $\mathcal{A}_c$ on t cannot contain $q_\perp$. Otherwise, $r(\varepsilon) = q_\perp$, which is not final.

Determinization

Proposition : Determinization

For all TA $\mathcal{A}$, there exists a deterministic TA $\mathcal{A}_{det}$ such that $L(\mathcal{A}_{det}) = L(\mathcal{A})$. Moreover, if $\mathcal{A}$ is complete, then $\mathcal{A}_{det}$ is complete. The size of $\mathcal{A}_{det}$ is exponential in the size of $\mathcal{A}$, its construction is EXPTIME.

pr.: subset construction.

Exercice :

Determinise and complete the previous TA (pattern matching of $\neg(\neg(x))$):

$$\mathcal{A} = \left(\Sigma, \{q, q_\neg, q_\mathsf{f}\}, \{q_\mathsf{f}\}, \left\{ \begin{array}{rclrcl} \bot & \to & q & \top & \to & q \\ \neg(q) & \to & q & \neg(q) & \to & q_\neg \\ \neg(q_\neg) & \to & q_\mathsf{f} & \neg(q_\mathsf{f}) & \to & q_\mathsf{f} \\ \vee(q,q) & \to & q & \wedge(q,q) & \to & q \\ \vee(q_\mathsf{f}, q_*) & \to & q_\mathsf{f} & \vee(q_*, q_\mathsf{f}) & \to & q_\mathsf{f} \end{array} \right\} \right)$$

subset construction: transitions

$$f(S_1, \ldots, S_n) \to \{q \mid \exists q_1 \in S_1 \ldots \exists q_n \in S_n \; f(q_1, \ldots, q_n \to q \in \Delta\}$$

where $S_1, \ldots, S_n \subseteq Q$.

Lemma

for all $t \in \mathcal{T}(\Sigma)$, $t \in L(\mathcal{A}_d, S)$ *iff* $S = \{q \in Q \mid t \in L(\mathcal{A}, q)\}$.

pr.:: induction on t.
REM: (see below) deterministic and complete TA = finite
Σ-algebra with homomorphism $\delta : \mathcal{T}(\Sigma) \to Q$. $L(\mathcal{A}) = \delta^{-1}(Q_\mathsf{f})$.

Top-Down Tree Automata and Determinism

Definition : Determinism

A top-down tree automaton $(\Sigma, Q, Q^{\text{init}}, \Delta)$ is *deterministic* if $|Q^{\text{init}}| = 1$ and for all state $q \in Q$ and $f \in \Sigma$, Δ contains at most one rule with left member q and symbol f.

The top-down tree automata are in general not determinizable .

Proposition :

There exists a regular tree language which is not recognizable by a deterministic top-down tree automaton.

pr.: $L = \{f(a,b), f(b,a)\}$.

Boolean Closure of Regular tree Languages

Proposition : Closure

The class of regular tree languages is closed under union, intersection and complementation.

op.	technique	time of calcul and size of automata
$\cup$	disjoint $\cup$	linear
$\cap$	Cartesian product	quadratic
$\neg$	determinization, completion, invert final / non-final states	exponential (lower bound)

Remark :

For the deterministic TA, the construction for the complementation is polynomial.

Plan

Cleaning

Definition : Clean

A state q of a TA $\mathcal{A}$ is called *inhabited* if there exists at least one $t \in L(\mathcal{A}, q)$. A TA is called *clean* if all its states are inhabited.

Proposition : Cleaning

For all TA $\mathcal{A}$, there exists a clean TA $\mathcal{A}_{clean}$ such that $L(\mathcal{A}_{clean}) = L(\mathcal{A})$. The size of $\mathcal{A}_{clean}$ is smaller than the size of $\mathcal{A}$, its construction is PTIME.

pr.: state marking algorithm, running time $O\big(|Q| \times \|\Delta\|\big)$.

state marking: we construct $M \subseteq Q$ containing all the inhabited states.

- start with $M = \emptyset$
- for all $f \in \Sigma$, of arity $n \geq 0$, and all $q_1, \ldots, q_n \in M$ such that there exists $f(q_1, \ldots, q_n) \to q$ in Δ, add q to M (if it was not already).

We iterate the last step until a fixpoint M_* is reached. We can show that $q \in M_*$ iff $existst \in L(\mathcal{A}, q)$.

Membership Problem

Definition : Membership

INPUT: a TA $\mathcal{A}$ over Σ, a term $t \in \mathcal{T}(\Sigma)$.
QUESTION: $t \in L(\mathcal{A})$?

Proposition : Membership

The membership problem is decidable in polynomial time.

Exact complexity:

- non-deterministic bottom-up: LOGCFL-complet
- deterministic bottom-up: unknown (LOGDCFL)
- deterministic top-down: LOGSPACE-complete.

if $\mathcal{A}$ is deterministic, we compute in polynomial time the unique run of $\mathcal{A}$ sur t.
if $\mathcal{A}$ is non-deterministic, we compute in polynomial time the unique run of $\mathcal{A}_{det}$ on t, computing the necessary states on the fly.
the number of states computed is $\leq \|t\|$. construction of r:

bottom relabeling of t:

- for p leaf position, $r(p) = \{q \mid t(p) \to q \in \Delta\}$
- for p internal node, $t(p)$ of arity n,

$$r(p) = \{q \mid \exists q_1 \in r(p1), \ldots, \exists q_n \in r(pn)\, t(p)(q_1, \ldots, q_n) \to q \in \Delta\}$$

LOGCFL: problems reducible in log-space into context-free language, between NL (NlogSPACE) and NC1 (Boolean circuits of depth log(n)).

LOGDCFL: problems reducibles in log-space into deterministic context-free language.

Emptiness Problem

Definition : Emptiness

INPUT: a TA $\mathcal{A}$ over Σ.
QUESTION: $L(\mathcal{A}) = \emptyset$?

Proposition : Emptiness

The emptiness problem is decidable in linear time.

pr.:
quadratic: clean, check if the clean automaton contains a final state.
linear: reduction to propositional HORN-SAT.
linear bis: optimization of the data structures for the cleaning (exo).

Remark :

The problem of the emptiness is PTIME-complete.

PTIME-complete: réduction of the problem called *generalisation*.
The generalisation: problem is PTIME-hard.

generalisation-IN: Q finite, $f : Q \times Q \to Q$, $V \subseteq Q$, $q \in Q$.

generalisation-QUESTION: $q \in f^*(V, V)$ (smallest set, containing V, closed by appl. of f.)

Instance-Membership Problem

Definition : Instance-Membership (IM)

INPUT: a TA $\mathcal{A}$ over Σ, a term $t \in \mathcal{T}(\Sigma, \mathcal{X})$.
QUESTION: does there exists $\sigma : vars(t) \to \mathcal{T}(\Sigma)$ s.t. $t\sigma \in L(\mathcal{A})$?

Proposition : Instance-Membership

1. The problem IM is decidable in polynomial time when t is linear.
2. The problem IM is NP-complet when $\mathcal{A}$ is deterministic.
3. The problem IM is EXPTIME-complete in general.

Problem of the Emptiness of Intersection

Definition : Emptiness of Intersection

INPUT: n TA $\mathcal{A}_1, \ldots, \mathcal{A}_n$ over Σ.
QUESTION: $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_n) = \emptyset$?

Proposition : Emptiness of Intersection

The problem of the emptiness of intersection is EXPTIME-complete.

pr.: EXPTIME: n applications of the closure under $\cap$ and emptiness decision.

EXPTIME-hardness: APSPACE $=$ EXPTIME
reduction of the problem of the existence of a successful run (starting from an initial configuration) of an alternating Turing machine (ATM) $M = (\Gamma, S, s_0, S_\mathsf{f}, \delta)$.

Let $M = (\Gamma, S, s_0, S_\mathsf{f}, \delta)$ be a Turing Machine (Γ: input alphabet, S: state set, s_0 initial state, S_f final states, δ: transition relation).
First some notations.

- a *configuration* of M is a word of $\Gamma^*\Gamma_S\Gamma^*$ where $\Gamma_S = \{a^s \mid a \in \Gamma, s \in S\}$. In this word, the letter of Γ_S indicates both the current state and the current position of the head of M.
- a *final configuration* of M is a word of $\Gamma^*\Gamma_{S_\mathsf{f}}\Gamma^*$.
- an *initial configuration* of M is a word of $\Gamma_{s_0}\Gamma^*$.
- a *transition* of M (following δ) between two configurations v and v' is denoted $v \rhd v'$

The initial configuration v_0 is accepting iff there exists a final configuration v_f and a finite sequence of transitions $v_0 \rhd \ldots \rhd v_\mathsf{f}$?
This problem whether v_0 is accepting is undecidable in general.
If the tape is polynomially bounded (we are restricted to configurations of length $n = |v_0|^c$, for some fixed $c \in \mathbb{N}$), the problem is PSPACE complete.
M alternating: $S = S_\exists \uplus S_\forall$.
Definition accepting configurations:

- every final configuration (whose state is in S_f) is accepting
- a configuration c whose state is in $S_\exists$ is accepting if it has at least one successor accepting
- a configuration c whose state is in $S_\forall$ is accepting if all its successors are accepting

Theorem (Chandra, Kozen, Stockmeyer 81)

APSPACE = EXPTIME

In order to show EXPTIME-hardness, we reduce the problem of deciding whether v_0 is accepting for M alternating and polynomially bounded.
Hypotheses (non restrictive):

- $s_0 \in S_\exists$ or $s_0 \in S_\forall \cap S_f$
- s_0 is non reentering (it only occurs in v_0)
- every configuration with state in $S_\forall$ has 0 or 2 successors
- final configurations are restricted to $\flat_{S_f}\flat^*$ where $\flat \in \Gamma$ is the blank symbol.
- S_f is a singleton.

2 technical definitions: for $k \leq n$,

$$\mathsf{view}(v,k) = \begin{array}{ll} v[k]v[k+1] & \text{if } k = 1 \\ v[k-1]v[k] & \text{if } k = n \\ v[k-1]v[k]v[k+1] & \text{otherwise} \end{array}$$

$$\mathsf{view}(v,v_1,v_2,k) = \langle \mathsf{view}(v,k), \mathsf{view}(v_1,k), \mathsf{view}(v_2,k) \rangle$$

$v \rhd_k \langle v_1, v_2 \rangle$ iff

1. if $v[k] \in \Gamma_S$, then $\exists w \rhd w_1, w_2$ s.t. $\mathsf{view}(v,v_1,v_2,k) = \mathsf{view}(w,w_1,w_2,k)$
2. if $v[k] = a \in \Gamma$, then $v_1[k] \in \{a\} \cup a_S$ and $v_2 = \varepsilon$ or $v_2[k] \in \{a\} \cup a_S$.

first item: around position k, we have two correct transitions of M. This can be tested by the membership of $\mathsf{view}(v,v_1,v_2,k)$ to a given set which only depends on M.

Lemma

$v \rhd v_1, v_2$ *iff* $\forall k \leq n \; v \rhd_k \langle v_1, v_2 \rangle$.

Term representations of runs:

rem. a run of M is not a sequence of configurations but a tree of configurations (because of alternation).

Signature Σ: $\emptyset$: constant, Γ: unary, S: unaires, p binary.

Notation: if $v = a_1 \dots a_n$, $v(x)$ denotes $a_n(a_{n-1}(\dots a_1(x)))$.

Term representations of runs:

- $v_{\mathsf{f}}(p(\emptyset, \emptyset))$ with v_{f} final configuration,
- $v(p(t_1, t_2))$ with v $\forall$-configuration, $t_1 = v'_1(p(t_{1,1}, t_{1,2}))$, $t_2 = v'_2(p(t_{2,1}, t_{2,2}))$ are two term representations of runs, and $v_1 \rhd v'_1$, $v_2 \rhd v'_2$
- $v(p(t_1, \emptyset))$ with v $\exists$-configuration, $t_1 = v'_1(p(t_{1,1}, t_{1,2}))$ term representations of run, and $v_1 \rhd v'_1$.

notations for $t_1 = v'_1(p(t_{1,1}, t_{1,2}))$:

- $\mathsf{head}(t_1) = v_1$
- $\mathsf{left}(t_1) = t_{1,1}$
- $\mathsf{right}(t_1) = t_{1,2}$.

This recursive definition suggest the construction of a TA recognizing term representations of successful runs. The difficulty

is the conditions $v_1 \rhd v_1'$, $v_2 \rhd v_2'$, for which we use the above lemma.

We build $2n$ deterministic automata :

for all $1 < k < n$, $\mathcal{A}_k$ recognizes

- $v_{\mathsf{f}}(p(\emptyset, \emptyset))$ (recall there is only 1 final configuration by hyp.)
- $v(p(t_1, t_2))$ such that $t_1 \neq \emptyset$ and
  - $v \rhd_k \langle \mathsf{head}(t_1), \mathsf{head}(t_2) \rangle$
  - $\mathsf{left}(t_1) \in L(\mathcal{A}_k)$, $\mathsf{right}(t_1) \in L(\mathcal{A}_k) \cup \{\emptyset\}$,
  - $t_2 = \emptyset$ or $\mathsf{left}(t_2) \in L(\mathcal{A}_k)$, $\mathsf{right}(t_2) \in L(\mathcal{A}_k) \cup \{\emptyset\}$

idea: $\mathcal{A}_k$ memorizes $\mathsf{view}(\mathsf{head}(t_1), k)$ and $\mathsf{view}(\mathsf{head}(t_2), k)$ and compare with $\mathsf{view}(v, k)$.

for all $1 < k < n$, $\mathcal{A}_k'$ recognizes the terms $v_0(p(t_1, t_2))$ with $t_1 = t_2 = \emptyset$ (if s_0 universal and final) or $t_2 = \emptyset$ (if s_0 existential, not final) and $t_1, t_2 \in T$, minimal set of terms without s_0 containing

- $\emptyset$
- $v(p(t_1, t_2))$ such that $t_1 \neq \emptyset$ and
  - $v \rhd_k \langle \mathsf{head}(t_1), \mathsf{head}(t_2) \rangle$
  - $\mathsf{left}(t_1) \in T$, $\mathsf{right}(t_1) \in T$,

- $t_2 = \emptyset$ or $\mathsf{left}(t_2) \in T$, $\mathsf{right}(t_2) \in T$

$$\text{representations of successful runs} = \bigcap_{k=1}^{n} L(\mathcal{A}_k) \cap L(\mathcal{A}'_k).$$

Problem of Universality

Definition : Universality

INPUT: a TA $\mathcal{A}$ over Σ.
QUESTION: $L(\mathcal{A}) = \mathcal{T}(\Sigma)$

Proposition : Universality

The problem of universality is EXPTIME-complete.

pr.: EXPTIME: Boolean closure and emptiness decision.

EXPTIME-hardness: again APSPACE $=$ EXPTIME.

Remark :

The problem of universality is decidable in polynomial time for the deterministic (bottom-up) TA.

pr.: completion and cleaning.

we build $\mathcal{A}$ recognizing the terms which are **not** representing a successful run of M. $\mathcal{A}$ is (necessarily) ND, this makes the construction simpler than for Theorem on emptiness of intersection.

$\mathcal{A}$ is the union of

- $\mathcal{A}_1$: error in encoding (ill-formed term)
- $\mathcal{A}_2$: not initial (use hyp. of Theorem $\emptyset$ of $\cap$)
- $\mathcal{A}_3$: non final "leaf" (use hyp. of Theorem $\emptyset$ of $\cap$)
- $\mathcal{A}_4$: problem with state $\forall$ or $\exists$.
  We use the hyp. that $\exists$ configurations have successor and $\forall$ configurations have 0 or 2 successors.
  $\mathcal{A}_4$ detect a wrong number of successors.
- $\mathcal{A}_5$: exists wrong transition.
  top-down ND, choose 1 k, 1 configuration v, 1 "successor" v_i $(1 \leq i \leq 2)$ and test that $v \not\rhd_k v_i$.

Problems of Inclusion an Equivalence

Definition : Inclusion

INPUT: two TA $\mathcal{A}_1$ and $\mathcal{A}_2$ over Σ.
QUESTION: $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$

Definition : Equivalence

INPUT: two TA $\mathcal{A}_1$ and $\mathcal{A}_2$ over Σ.
QUESTION: $L(\mathcal{A}_1) = L(\mathcal{A}_2)$

Proposition : Inclusion, Equivalence

The problems of inclusion and equivalence are EXPTIME-complete.

pr.: $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)} = \emptyset$.
EXPTIME-hardness: universality is $\mathcal{T}(\Sigma) = L(\mathcal{A}_2)$?

Remark :

If $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic, it is $O\big(\|\mathcal{A}_1\| \times \|\mathcal{A}_2\|\big)$.

Problem of Finiteness

Definition : Finiteness

INPUT: a TA $\mathcal{A}$
QUESTION: is $L(\mathcal{A})$ finite?

Proposition : Finiteness

The problem of finiteness is decidable in polynomial time.

pumping: $L(\mathcal{A})$ is of infinite cardinality iff
$\exists t \in L(\mathcal{A}), |Q| < h(t) \leq 2|Q|$ (but the test is not polynomial).
polynomial: search for the existence of a loop
$\exists q$ inhabited (quadratic test), $q_{\mathsf{f}} \in Q\mathsf{f}$, C, D contexts s.t.

- $C[q] \xrightarrow[\mathcal{A}]{*} q$ (test in $O(\|\mathcal{A}\|)$) and

- $D[q] \xrightarrow[\mathcal{A}]{*} q_{\mathsf{f}}$ (test in $O(\|\mathcal{A}\|)$).

Plan

Theorem of Myhill-Nerode

Definition :

A *congruence* $\equiv$ on $\mathcal{T}(\Sigma)$ is an equivalence relation such that for all $f \in \Sigma_n$, if $s_1 \equiv t_1, \ldots, s_n \equiv t_n$, then $f(s_1, \ldots, s_n) \equiv f(t_1, \ldots, t_n)$.

Given $L \subseteq \mathcal{T}(\Sigma)$, the congruence $\equiv_L$ is defined by:

$s \equiv_L t$ if for all context $C \in \mathcal{T}\big(\Sigma, \{x\}\big)$, $C[s] \in L$ iff $C[t] \in L$.

Theorem : Myhill-Nerode

The three following propositions are equivalent:

1. L is regular
2. L is a union of equivalence classes for a congruence $\equiv$ of finite index
3. $\equiv_L$ is a congruence of finite index

Proof

$1 \Rightarrow 2$. $\mathcal{A}$ deterministic, def. $s \equiv_{\mathcal{A}} t$ iff $\mathcal{A}(s) = \mathcal{A}(t)$.

$2 \Rightarrow 3$. we show that if $s \equiv t$ then $s \equiv_L t$, hence the index of $\equiv_L \leq$ index of $\equiv$ (since we have $\equiv \subseteq \equiv_L$). If $s \equiv t$ then $C[s] \equiv C[t]$ for all $C[\,]$ (induction on C), hence $C[s] \in L$ iff $C[t] \in L$, i.e. $s \equiv_L t$.

$3 \Rightarrow 1$. we construct $\mathcal{A}_{\mathsf{min}} = (Q_{\mathsf{min}}, Q^{\mathsf{f}}_{\mathsf{min}}, \Delta_{\mathsf{min}})$,

- $Q_{\mathsf{min}} =$ equivalence classes of $\equiv_L$,
- $Q^{\mathsf{f}}_{\mathsf{min}} = \{[s] \mid s \in L\}$,
- $\Delta_{\mathsf{min}} = \{f\big([s_1], \ldots, [s_n]\big) \to \big[f(s_1, \ldots, s_n)\big]\}$

Clearly, $\mathcal{A}_{\mathsf{min}}$ is deterministic, and for all $s \in \mathcal{T}(\Sigma)$, $\mathcal{A}_{\mathsf{min}}(s) = [s]_L$, i.e. $s \in L(\mathcal{A}_{\mathsf{min}})$ iff $s \in L$.

Note that the $\#$ of states of $\mathcal{A}_{\mathsf{min}}$ is $\leq \#$ of states of $\mathcal{A}$ f.a. $\mathcal{A}$ s.t. $L(\mathcal{A}) = L$, because $\equiv_{\mathcal{A}}$ is finer that $\equiv_{L(\mathcal{A})}$ ($\equiv_{\mathcal{A}} \subseteq \equiv_{L(\mathcal{A})}$ hence index $\equiv_{L(\mathcal{A})} \leq$ index $\equiv_{\mathcal{A}}$).

Minimization

Corollary :

For all DTA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$, there exists a unique DTA $\mathcal{A}_{\mathsf{min}}$ whose number of states is the index of $\equiv_{L(\mathcal{A})}$ and such that $L(\mathcal{A}_{\mathsf{min}}) = L(\mathcal{A})$.

Let $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ be a DTA, we build a deterministic minimal automaton $\mathcal{A}_{\mathsf{min}}$ as in the proof of $3 \Rightarrow 1$ of the previous theorem for $L(\mathcal{A})$ (i.e. Q_{min} is the set of equivalence classes for $\equiv_{L(\mathcal{A})}$).

We build first an equivalence $\approx$ on the states of Q:

- $q \approx_0 q'$ iff $q, q' \in Q^{\mathsf{f}}$ ou $q, q' \in Q \setminus Q^{\mathsf{f}}$.
- $q \approx_{k+1} q'$ iff $q \approx_k q'$ et $\forall f \in \Sigma_n$, $\forall q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n \in Q$ $(1 \leq i \leq n)$,

$$\Delta\big(f(q_1, \ldots, q_{i-1}, q, q_{i+1}, \ldots, q_n)\big) \approx_k \Delta\big(f(q_1, \ldots, q_{i-1}, q', q_{i+1}, \ldots,$$

Let $\approx$ be the fixpoint of this construction, $\approx$ is $\equiv_{L(\mathcal{A})}$, hence $\mathcal{A}_{\mathsf{min}} = (\Sigma, Q_{\mathsf{min}}, Q^{\mathsf{f}}_{\mathsf{min}}, \Delta_{\mathsf{min}})$ with :

- $Q_{\mathsf{min}} = \{[q]_\approx \mid q \in Q\}$,
- $Q^{\mathsf{f}}_{\mathsf{min}} = \{[q^{\mathsf{f}}]_\approx \mid q^{\mathsf{f}} \in Q^{\mathsf{f}}\}$,
- $\Delta_{\mathsf{min}} = \big\{f\big([q_1]_\approx, \ldots, [q_n]_\approx\big) \to \big[f(q_1, \ldots, q_n)\big]_\approx\big\}$.

recognizes $L(\mathcal{A})$. and it is smaller than $\mathcal{A}$.

Algebraic Characterization of Regular Languages

Corollary :

A set $L \subseteq \mathcal{T}(\Sigma)$ is regular iff there exists

- a Σ-algebra $\mathcal{Q}$ of finite domain Q,
- an homomorphism $h : \mathcal{T}(\Sigma) \to \mathcal{A}$,
- a subset $Q^{\mathsf{f}} \subseteq Q$ such that $L = h^{-1}(Q^{\mathsf{f}})$.

operations of $\mathcal{Q}$:
for each $f \in \Sigma_n$, there is a function $f^{\mathcal{Q}} : Q^n \to Q$.

proof:

$\Rightarrow$: deterministic and complete TA $\mathcal{A}$ = finite Σ-algebra with domain $Q_{\mathcal{A}}$, and homomorphism $h : \mathcal{T}(\Sigma) \to Q$. $L(\mathcal{A}) = h^{-1}(Q_{\mathsf{f}})$.

$\Leftarrow$: define $s \equiv t$ iff $h(s) = h(t)$.
It is a congruence of finite index and L is a union of equivalence classes of $\equiv$.
Then we use $2 \Rightarrow 1$ of Th. Myhill Nerode.

Plan

Tree Transformations, Verification

- formalisms for the transformation of terms (languages): rewrite systems, tree homomorphisms, transducers...
  - $=$ transitions in an infinite states system,
  - $=$ evaluation of programs,
  - $=$ transformation of XML documents, updates...
- problem of the type checking:

  given:
  - $L_{\text{in}} \subseteq \mathcal{T}(\Sigma)$, (regular) input language
  - h transformation $\mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$
  - $L_{\text{out}} \subseteq \mathcal{T}(\Sigma')$ (regular) output language

  question: do we have $h(L_{\text{in}}) \subseteq L_{\text{out}}$?

Tree Homomorphisms

Definition :

$$h : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$$
$$h\big(f(t_1, \ldots, t_n)\big) := t_f\big\{x_1 \leftarrow h(t_1), \ldots, x_n \leftarrow h(t_n)\big\}$$
for $f \in \Sigma_n$, with $t_f \in \mathcal{T}\big(\Sigma', \{x_1, \ldots, x_n\}\big)$.

h is called

- *linear* if for all $f \in \Sigma$, t_f is linear,
- *complet* if for all $f \in \Sigma_n$, $vars(t_f) = \{x_1, \ldots, x_n\}$,
- *symbol-to-symbol* if for all $f \in \Sigma_n$, $height(t_f) = 1$.

Homomorphisms: examples

Example : ternary trees $\to$ binary trees

Let $\Sigma = \{a : 0, b : 0, g : 3\}$, $\Sigma' = \{a : 0, b : 0, f : 2\}$ and $h : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$ defined by

- $t_a = a$,
- $t_b = b$,
- $t_g = f(x_1, f(x_2, x_3))$.

$h\big(g(a, g(b, b, b), a)\big) = f(a, f(f(f(b, f(b, b))), a))$

Example : Elimination of the $\wedge$

Let $\Sigma = \{0 : 0, 1 : 0, \neg : 1, \vee : 2, \wedge : 2\}$, $\Sigma' = \{0 : 0, 1 : 0, \neg : 1, \vee : 2\}$ and $h : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$ with $t_\wedge = \neg(\vee(\neg(x_1), \neg(x_2)))$.

Closure of Regular languages under Linear Homomorphisms

Theorem :

If L is regular and h is a linear homomorphism, then $h(L)$ is regular.

This is not true in general for the non-linear homomorphisms.

Example : Non-linear homomorphisms

$\Sigma = \{a : 0, g : 1, f : 1\}$, $\Sigma' = \{a : 0, g : 1, f' : 2\}$,
$h : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$ with $t_a = a$, $t_g = g(x_1)$, $t_f = f'(x_1, x_1)$.
Let $L = \left\{ f\left(g^n(a)\right) \;\middle|\; n \geq 0 \right\}$,
$h(L) = \left\{ f'\left(g^n(a), g^n(a)\right) \;\middle|\; n \geq 0 \right\}$ is not regular.

proof theorem (the construction is in [TATA])
let $\mathcal{A} = (Q, Q^{\mathsf{f}}, \Delta)$ be clean and s.t. $L(\mathcal{A}) = L$, we build $\mathcal{A}' = (Q', Q'_{\mathsf{f}}, \Delta')$ with ε-transitions s.t. $L(\mathcal{A}') = h(L(\mathcal{A}))$.
For each $r = f(q_1, \ldots, q_n) \to q \in \Delta$, with $t_f \in \mathcal{T}(\Sigma', \mathcal{X}_n)$ (linear), let $Q^r = \{q^r_p \mid p \in \mathcal{P}os(t_f)\}$, and Δ_r defined as follows: for all $p \in \mathcal{P}os(t_f)$:

- if $t_f(p) = g \in \Sigma'_m$, then $g(q^r_{p_1}, \ldots, q^r_{p_m}) \to q^r_p \in \Delta_r$,
- if $t_f(p) = x_i$, then $q_i \xrightarrow{\varepsilon} q^r_p \in \Delta_r$,
- $q^r_\epsilon \xrightarrow{\varepsilon} q \in \Delta_r$.

- $Q' = Q \cup \bigcup_{r \in \Delta} Q^r$,
- $Q'_{\mathsf{f}} = Q_{\mathsf{f}}$,
- $\Delta' = \bigcup_{r \in \Delta} \Delta_r$.

$h(L) = L(\mathcal{A}')$.

Closure of Regular Languages under Inverse Homomorphisms

Theorem :

For all regular languages L and all homomorphisms h, $h^{-1}(L)$ is regular.

$\mathcal{A}' = (Q', Q'_{\mathsf{f}}, \Delta')$ complete deterministic such that $L(\mathcal{A}') = L$.
We construct $\mathcal{A} = (Q, Q_{\mathsf{f}}, \Delta)$ with $Q = Q' \cup \{s\}$ where $s \notin Q'$, $Q_f = Q'_{\mathsf{f}}$ and Δ is defined by:

- for $a \in \Sigma_0$, if $t_a \xrightarrow[\mathcal{A}']{*} q$ then $a \to q \in \Delta$;
- for $f \in \Sigma_n$ with $n > 0$, for $p_1, \ldots, p_n \in Q$, if $t_f\{x_1 \leftarrow p_1, \ldots, x_n \leftarrow p_n\} \xrightarrow[\mathcal{A}']{*} q$ then $f(q_1, \ldots, q_n) \to q \in \Delta$ where $q_i = p_i$ if x_i occurs in t_f and $q_i = s$ otherwise;
- for $a \in \Sigma_0$, $a \to s \in \Delta$;
- for $f \in \Sigma_n$ where $n > 0$, $f(s, \ldots, s) \to s \in \Delta$.

$t \xrightarrow[\mathcal{A}]{*} q$ iff $h(t) \xrightarrow[\mathcal{A}']{*} q$.

Closure under Homomorphisms

Theorem :

The class of regular tree languages is the smallest non trivial classe of sets of trees closed under linear homomorphisms and inverse homomorphisms.

A problem whose decidability has been open for 35 years:

INPUT: a TA $\mathcal{A}$, an homomorphism h
QUESTION: is $h(L(\mathcal{A}))$ regular?

Tree Transducers

Definition : Bottom-up Tree Transducers

A *bottom-up tree transducer* (TT) is a tuple $U = (\Sigma, \Sigma', Q, Q^{\mathsf{f}}, \Delta)$ where

- Σ, Σ' are the input, resp. output, signatures,
- Q is a finite set of *states*,
- $Q^{\mathsf{f}} \subseteq Q$ is the subset of final states
- Δ is a set of transduction (rewrite) rules of the form:
  - $f(q_1(x_1), \ldots, q_n(x_n)) \to q(u)$ with $f \in \Sigma_n$ $(n \geq 0)$, $q_1, \ldots, q_n, q \in Q$, $x_1, \ldots, x_n$ pairwise distinct and $u \in \mathcal{T}(\Sigma', \{x_1, \ldots, x_n\})$, or
  - $q(x_1) \to q'(u)$ with $q, q' \in Q$, $u \in \mathcal{T}(\Sigma', \{x_1\})$.

A TT is *linear* if all the u in transduction rules are linear.

The transduction relation of U is the binary relation:

$$L(U) = \left\{ \langle t, t' \rangle \;\middle|\; t \xrightarrow[U]{*} q(t'), t \in \mathcal{T}(\Sigma), t' \in \mathcal{T}(\Sigma'), q \in Q^{\mathsf{f}} \right\}$$

Example 1

$$U_1 = \big(\{f:1, a:0\}, \{g:2, f, f':1, a:0\}, \{q, q'\}, \{q'\}, \Delta_1\big),$$

$$\Delta_1 = \left\{ \begin{array}{rcl} a & \to & q(a) \\ f(q(x_1)) & \to & q(f(x_1)) \mid q(f'(x_1)) \mid q'(g(x_1, x_1)) \end{array} \right\}$$

Example 2

$\Sigma_{in} = \{f : 2, g : 1, a : 0\}$,
$U_2 = \big(\Sigma_{in}, \Sigma_{in} \cup \{f' : 1\}, \{q, q', q_{\mathsf{f}}\}, \{q_{\mathsf{f}}\}, \Delta_2\big)$,

$$\Delta_2 = \left\{ \begin{array}{rcl} a & \to & q(a) \mid q'(a) \\ g(q(x_1)) & \to & q(g(x_1)) \\ g(q'(x_1)) & \to & q'(g(x_1)) \\ f(q'(x_1), q'(x_2)) & \to & q'(f(x_1, x_2)) \\ f(q'(x_1), q'(x_2)) & \to & q_{\mathsf{f}}(f'(x_1)) \end{array} \right\}$$

$$L(U_2) = \left\{ \langle f(t_1, t_2), f'(t_1) \mid t_2 = g^m(a), m \geq 0 \right\}$$

Languages

Theorem :

- The domain of a TT is a regular tree language.
- The image of a regular tree language by a linear TT is a regular tree language.

Transducers and Homomorphisms

An homomorphism is called *delabeling* if it is linear, complete, symbol-to-symbol.

Definition : Bimorphisms

A *bimorphism* is a triple $B = (h, h', L)$ where h, h' are homomorphisms and L is a regular tree language.

$$L(B) = \left\{ \langle h(t), h'(t) \rangle \;\middle|\; t \in L \right\}$$

Theorem :

TT $\equiv$ bimorphisms (h, h', L) where h delabeling.

TATA book http://tata.gforge.inria.fr

Tree Automata Techniques and Applications

chapter 3:

- automata for tuples of trees,
- logic WSkS,
- applications.

Hubert Comon Max Dauchet Rémi Gilleron
Florent Jacquemard Denis Lugiez Christof Löding
Sophie Tison Marc Tommasi

Logic and Automata

- logic for expressing properties of labeled binary trees
  = specification of tree languages, example:

$$t \models \forall x\ a(x) \Rightarrow \exists y\ y > x \wedge b(y)$$

- compilation of formulae into automata
  = decision algorithms.
- equivalence between both formalisms
  [Thatcher & Wright's theorem].

Plan

WSkS: Definition

Automata $\rightarrow$ Logic

Logic $\rightarrow$ Automata

Fragments and Extensions of WSkS

Monadic second-order logic of the infinite binary tree (S2S)

Interpretation Structures

$\mathcal{L} :=$ set of predicate symbols $P_1, \ldots P_n$ with arity.

A structure $\mathcal{M}$ over $\mathcal{L}$ is a tuple

$$\mathcal{M} := \left\langle \mathcal{D}, P_1^{\mathcal{M}}, \ldots, P_n^{\mathcal{M}} \right\rangle$$

where

- $\mathcal{D}$ is the domain of $\mathcal{M}$,
- every $P_i^{\mathcal{M}}$ (interpretation of P_i) is a subset of $\mathcal{D}^{arity(P_i)}$ (relation).

Term as structure

Σ signature, k = maximal arity.

$$\mathcal{L}_\Sigma := \{=, <, S_1, \ldots, S_k, L_a \mid a \in \Sigma\}.$$

to $t \in \mathcal{T}(\Sigma)$, we associate a structure $\underline{t}$ over $\mathcal{L}_\Sigma$

$$\underline{t} := \langle \mathcal{P}os(t), =, <, S_1, \ldots, S_k, L_a^t, L_b^t, \cdots \rangle$$

where

- domain = positions of t $\quad (\mathcal{P}os(t) \subset \{1, \ldots, k\}^*)$
- $=$ equality over $\mathcal{P}os(t)$,
- $<$ prefix ordering over $\mathcal{P}os(t)$,
- $S_i = \{\langle p, p \cdot i \rangle \mid p, p \cdot i \in \mathcal{P}os(t)\}$ (i^{th} successor position),
- $L_a^t = \{p \in \mathcal{P}os(t) \mid t(p) = a\}$.

FOL with k successors

- first order variables $x, y \ldots$
- form $::=$ $x = y \mid x < y$
  $\mid S_1(x, y) \mid \ldots \mid S_k(x, y) \mid L_a(x) \quad a \in \Sigma$
  $\mid$ form $\wedge$ form $\mid$ form $\vee$ form $\mid \neg$form
  $\mid \exists x$ form $\mid \forall x$ form

Notation: $\phi(x_1, \ldots, x_m)$,
where $x_1, \ldots, x_m$ are the free variables of ϕ.

WSkS: syntax

- first order variables $x, y \ldots$
- second order variables $X, Y \ldots$
- form $::=$ $x = y \mid x < y \mid x \in X$
  $\mid S_1(x,y) \mid \ldots \mid S_k(x,y) \mid L_a(x) \quad a \in \Sigma$
  $\mid$ form $\wedge$ form $\mid$ form $\vee$ form $\mid \neg$form
  $\mid \exists x$ form $\mid \exists X$ form $\mid \forall x$ form $\mid \forall X$ form

Notation: $\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)$,
where $x_1, \ldots, x_m$, $X_1, \ldots, X_n$ are the free variables of ϕ.

WSkS: semantics

- $t \in \mathcal{T}(\Sigma)$,
- valuation σ of first order variables into $\mathcal{P}os(t)$,
- valuation δ of second order variables into subsets of $\mathcal{P}os(t)$,
- $\underline{t}, \sigma, \delta \models x = y$ iff $\sigma(x) = \sigma(y)$,
- $\underline{t}, \sigma, \delta \models x < y$ iff $\sigma(x) <_{prefix} \sigma(y)$,
- $\underline{t}, \sigma, \delta \models x \in X$ iff $\sigma(x) \in \delta(X)$,
- $\underline{t}, \sigma, \delta \models S_i(x, y)$ iff $\sigma(y) = \sigma(x) \cdot i$,
- $\underline{t}, \sigma, \delta \models L_a(x)$ iff $t(\sigma(x)) = a$ i.e. $\sigma(x) \in L_a^t$,
- $\underline{t}, \sigma, \delta \models \phi_1 \wedge \phi_2$ iff $\underline{t}, \sigma, \delta \models \phi_1$ and $\underline{t}, \sigma, \delta \models \phi_2$,
- $\underline{t}, \sigma, \delta \models \phi_1 \vee \phi_2$ iff $\underline{t}, \sigma, \delta \models \phi_1$ or $\underline{t}, \sigma, \delta \models \phi_2$,
- $\underline{t}, \sigma, \delta \models \neg\phi$ iff $\underline{t}, \sigma, \delta \not\models \phi$,

WSkS: semantics (quantifiers)

- $\underline{t}, \sigma, \delta \models \exists x\ \phi$ iff $x \notin dom(\sigma)$, x free in ϕ and exists $p \in \mathcal{P}os(t)$ s.t. $\underline{t}, \sigma \cup \{x \mapsto p\}, \delta \models \phi$,
- $\underline{t}, \sigma, \delta \models \forall x\ \phi$ iff $x \notin dom(\sigma)$, x free in ϕ and for all $p \in \mathcal{P}os(t)$, $\underline{t}, \sigma \cup \{x \mapsto p\}, \delta \models \phi$,
- $\underline{t}, \sigma, \delta \models \exists X\ \phi$ iff $X \notin dom(\delta)$, X free in ϕ and exists $P \subseteq \mathcal{P}os(t)$ s.t. $\underline{t}, \sigma, \delta \cup \{X \mapsto P\} \models \phi$,
- $\underline{t}, \sigma, \delta \models \forall X\ \phi$ iff $X \notin dom(\delta)$, X free in ϕ and for all $P \subseteq \mathcal{P}os(t)$, $\underline{t}, \sigma, \delta \cup \{X \mapsto P\} \models \phi$.

WSkS: languages

Definition : WSkS-definability

For $\phi \in$ WSkS closed (without free variables) over $\mathcal{L}_\Sigma$,

$$L(\phi) := \{t \in \mathcal{T}(\Sigma) \mid \underline{t} \models \phi\}.$$

Example :

$\Sigma = \{a : 2, b : 2, c : 0\}$. Language of terms in $\mathcal{T}(\Sigma)$

- containing the pattern $a(b(x_1, x_2), x_3)$:
  $\exists x \exists y\ S_1(x, y) \wedge L_a(x) \wedge L_b(y)$
- such that every a-labelled node has a b-labelled child.
  $\forall x \exists y\ L_a(x) \Rightarrow \bigvee_{i=1}^{2} S_i(x, y) \wedge L_b(y)$
- such that every a-labelled node has a b-labelled descendant.
  $\forall x \exists y\ L_a(x) \Rightarrow x < y \wedge L_b(y)$

WSkS: examples

- root position: $\mathsf{root}(x) \equiv \neg\exists y\ y < x$
- inclusion: $X \subseteq Y \equiv \forall x(x \in X \Rightarrow x \in Y)$
- intersection: $Z = X \cap Y \equiv \forall x\ (x \in Z \Leftrightarrow (x \in X \wedge x \in Y))$
- emptiness: $X = \emptyset \equiv \forall x\ x \notin X$
- finite union:
  $$X = \bigcup_{i=1}^{n} X_i \equiv (\bigwedge_{i=1}^{n} X_i \subseteq X) \wedge \forall x\ (x \in X \Rightarrow \bigvee_{i=1}^{n} x \in X_i)$$
- partition:
  $$X_1, \ldots, X_n \text{ partition } X \equiv X = \bigcup_{i=1}^{n} X_i \wedge \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} X_i \cap X_j = \emptyset$$

WSkS: examples (2)

- singleton:
  $\mathsf{sing}(X) \equiv X \neq \emptyset \wedge \forall Y \ \big(Y \subseteq X \Rightarrow (Y = X \vee Y = \emptyset)\big)$
- $\leq$ (without $<$)

$$x \leq y \quad \equiv \quad \forall X \left(\begin{array}{l} y \in X \\ \wedge \, \forall z \, \forall z' \, (z' \in X \wedge \bigvee_{i \leq k} S_i(z, z')) \Rightarrow z \in X \end{array} \right) \Rightarrow x \in X$$

or

$$x \leq y \quad \equiv \quad \exists X \, \big(\forall z \; z \in X \Rightarrow (\exists z' \bigvee_{i \leq k} S_i(z', z) \wedge z' \in X) \vee z = x\big) \wedge y \in X$$

Thatcher & Wright's Theorem

Theorem : Thatcher and Wright

Languages of WSkS formulae $=$ regular tree languages.

pr.: 2 directions (2 constructions):

- TA $\rightarrow$ WSkS,
- WSkS $\rightarrow$ TA.

Plan

WSkS: Definition

Automata $\rightarrow$ Logic

Logic $\rightarrow$ Automata

Fragments and Extensions of WSkS

Monadic second-order logic of the infinite binary tree (S2S)

Regular languages $\rightarrow$ WSkS languages

Let $\Sigma = \{a_1, \ldots, a_n\}$.

Theorem :

For all tree automaton $\mathcal{A}$ over Σ, there exists $\phi_{\mathcal{A}} \in$ WSkS such that $L(\phi_A) = L(\mathcal{A})$.

$\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ with $Q = \{q_0, \ldots, q_m\}$.
$\phi_{\mathcal{A}}$: existence of an accepting run of $\mathcal{A}$ on $t \in \mathcal{T}(\Sigma)$.

$$\phi_{\mathcal{A}} := \exists Y_0 \ldots \exists Y_m \; \phi_{\mathsf{lab}}(\overline{Y}) \wedge \phi_{\mathsf{acc}}(\overline{Y}) \wedge \phi_{\mathsf{tr}_0}(\overline{Y}) \wedge \phi_{\mathsf{tr}}(\overline{Y})$$

regular languages $\rightarrow$ WSkS languages

$\phi_{\mathsf{lab}}(\overline{Y})$: every position is labeled with one state exactely.

$$\phi_{\mathsf{lab}}(\overline{Y}) \equiv \forall x \quad \bigvee_{0 \leq i \leq m} \quad x \in Y_i \ \wedge \quad \bigwedge_{\substack{0 \leq i,j \leq m \\ i \neq j}} \quad \left(x \in Y_i \Rightarrow \neg x \in Y_j\right)$$

$\phi_{\mathsf{acc}}(\overline{Y})$: the root is labeled with a final state

$$\phi_{\mathsf{acc}}(\overline{Y}) \equiv \forall x_0 \ \mathsf{root}(x_0) \Rightarrow \bigvee_{q_i \in Q^{\mathsf{f}}} x_0 \in Y_i$$

regular languages $\rightarrow$ WSkS languages

$\phi_{\mathsf{tr}_0}(\overline{Y})$: transitions for constants symbols

$$\phi_{\mathsf{tr}_0}(\overline{Y}) \equiv \bigwedge_{a \in \Sigma_0} \left(\forall x \; L_a(x) \Rightarrow \bigvee_{a \rightarrow q_i \in \Delta} x \in Y_i \right)$$

$\phi_{\mathsf{tr}}(\overline{Y})$: transitions for non-constant symbols

$$\begin{aligned} \phi_{\mathsf{tr}}(\overline{Y}) \quad \equiv \quad & \bigwedge_{f \in \Sigma_j, 0 < j \leq k} \forall x \, \forall y_1 \ldots \forall y_j \\ & \big(L_f(x) \wedge S_1(x, y_1) \wedge \ldots \wedge S_j(x, y_j) \big) \\ & \Downarrow \\ & \bigvee_{f(q_{i_1}, \ldots, q_{i_j}) \rightarrow q_i \in \Delta} x \in Y_i \wedge y_1 \in Y_{i_1} \wedge \ldots \wedge y_j \in Y_{i_j} \end{aligned}$$

Plan

WSkS: Definition

Automata $\rightarrow$ Logic

Logic $\rightarrow$ Automata

Fragments and Extensions of WSkS

Monadic second-order logic of the infinite binary tree (S2S)

Theorem Thatcher & Wright

Theorem :

Every WSkS language is regular.

For all formula $\phi \in$ WSkS over Σ (without free variables) there exists a tree automaton $\mathcal{A}_\phi$ over Σ, such that $L(\mathcal{A}_\phi) = L(\phi)$.

Corollary :

WSkS is decidable.

pr.: reduction to emptiness decision for $\mathcal{A}_\phi$.

Theorem Thatcher & Wright

$\mathcal{A}_\phi$ is effectively constructed from ϕ, by induction.

- automata for atoms

  $\Rightarrow$ need of automata for formula with free variables.

  it will characterize
- Boolean closures for Boolean connectors.
- $\exists$ quantifier: projection.

Theorem Thatcher & Wright

When ϕ contains free variables, $\mathcal{A}_\phi$ will characterize both terms AND valuations satisfying ϕ: $L(\mathcal{A}_\phi) \equiv \{\langle t, \sigma, \delta\rangle \mid \underline{t}, \sigma, \delta \models \phi\}$. Below we define the product $\langle t, \sigma, \delta\rangle$.

✓ for free second order variables:

$$\frac{\begin{array}{c} t \in \mathcal{T}(\Sigma) \\ \delta : \{X_1, \ldots, X_n\} \to 2^{\mathcal{P}os(t)} \end{array}}{\quad \mapsto \quad t \times \delta \in \mathcal{T}(\Sigma \times \{0,1\}^n)}$$

arity of $\langle a, \overline{b}\rangle$ in $\Sigma \times \{0,1\}^n$ = arity of a in Σ.

for all $p \in \mathcal{P}os(t)$, $(t \times \delta)(p) = \langle t(p), b_1, \ldots, b_n\rangle$ where for all $i \leq n$,

- $b_i = 1$ if $p \in \delta(X_i)$,
- $b_i = 0$ otherwise.

✓ free first order variables are interpreted as singletons.

WSkS$_0$

We consider a simplified language (wlog).

- no first order variables,
- only second order variables $X, Y \ldots$,
- form ::= $X \subseteq Y \mid Y = X \cdot 1 \mid \ldots \mid Y = X \cdot k$
  $\mid X \subseteq L_a \quad a \in \Sigma$
  $\mid$ form $\wedge$ form $\mid$ form $\vee$ form $\mid \neg$form
  $\mid \exists X$ form $\mid \forall X$ form

interpretation $Y = X \cdot i$: $X = \{x\}$, $Y = \{y\}$ and $y = x \cdot i$.

ex: singleton

$$\text{singleton}(X) \equiv \exists Y \quad \big(Y \subseteq X \wedge Y \neq X \wedge \neg\exists Z \, (Z \subseteq X \wedge Z \neq X \wedge Z \neq Y)\big)$$

WSkS $\rightarrow$ WSkS$_0$

Lemma :

For all formula $\phi(x_1, \ldots, x_m, X_1, \ldots, X_n) \in$ WSkS,
there exists a formula $\phi'(X'_1, \ldots, X'_m, X_1, \ldots, X_n) \in$ WSkS$_0$
s.t. $\underline{t}, \sigma, \delta \models \phi(x_1, \ldots, x_m, X_1, \ldots, X_n)$
iff $\underline{t}, \sigma' \cup \delta \models \phi'(X'_1, \ldots, X'_m, X_1, \ldots, X_n)$, with $\sigma' : X'_i \mapsto \{\sigma(x_i)\}$.

pr.: several steps of formula rewriting:

1. elimination of $<$,
2. elimination of $S_i(x, y)$ $(i \leq k)$, $L_a(x)$ $(a \in \Sigma)$,
   elimination of first order variables (use singleton(X)).

compilation of WSkS$_0$ into automata

notation: $\Sigma_{[m]} := \Sigma \times \{0,1\}^m$.

For all $\phi(X_1, \ldots, X_n) \in$ WSkS$_0$ and $m \geq n$,
we construct a tree automaton $[\![\phi]\!]_m$ over $\Sigma_{[m]}$ recognizing

$$\left\{ t \times \delta \mid \delta : \{X_1, \ldots, X_m\} \to 2^{\mathcal{P}os(t)},\ \underline{t}, \delta \models \phi(X_1, \ldots, X_n) \right\}$$

projection, cylindrification

projection

$proj_n$: $\bigcup_{m \geq n} \mathcal{T}(\Sigma_{[m]}) \to \mathcal{T}(\Sigma_{[n]})$
delete components $n+1, \ldots, m$.

Lemma : projection

For all $n \leq m$, if $L \subseteq \mathcal{T}(\Sigma_{[m]})$ is regular then $proj_n(L)$ is regular.

cylindrification $(m \geq n)$

$cyl_{n,m} : L \subseteq \mathcal{T}(\Sigma_{[n]}) \mapsto \{t \in \mathcal{T}(\Sigma_{[m]}) \mid proj_n(t) \in L\}$

Lemma : cylindrification

For all $n \leq m$, if $L \subseteq \mathcal{T}(\Sigma_{[n]})$ is regular, then $cyl_{n,m}(L)$ is regular.

compilation: $X_1 \subseteq X_2$

Automaton $[\![X_1 \subseteq X_2]\!]_2$:

- signature $\Sigma_{[2]} = \Sigma \times \{0,1\}^2$.
- states: q_0
- final states: q_0
- transitions:

$$\begin{array}{rcl} \langle a,0,0\rangle(q_0,\ldots,q_0) & \to & q_0 \\ \langle a,0,1\rangle(q_0,\ldots,q_0) & \to & q_0 \\ \langle a,1,1\rangle(q_0,\ldots,q_0) & \to & q_0 \end{array}$$

For $m \geq 2$,

$$[\![X_1 \subseteq X_2]\!]_m := cyl_{2,m}\big([\![X_1 \subseteq X_2]\!]_2\big)$$

compilation: $X_1 = X_2 \cdot 1$

Automaton $[\![X_1 = X_2 \cdot 1]\!]_2$:

- signature $\Sigma_{[2]} = \Sigma \times \{0,1\}^2$.
- states: q_0, q_1, q_2
- final states: q_2
- transitions:

$$\begin{array}{lcl}
\langle a,0,0\rangle(q_0,\ldots,q_0) & \to & q_0 \\
\langle a,1,0\rangle(q_0,\ldots,q_0) & \to & q_1 \\
\langle a,0,1\rangle(q_1,q_0,\ldots,q_0) & \to & q_2 \\
\langle a,0,0\rangle(q_0,\ldots,q_0,q_2,q_0,\ldots,q_0) & \to & q_2
\end{array}$$

For $m \geq 2$,

$$[\![X_2 = X_1 \cdot 1]\!]_m := cyl_{2,m}\big([\![X_2 = X_1 \cdot 1]\!]_2\big)$$

compilation: $X_1 \subseteq L_a$

Automate $[\![X_1 \subseteq L_a]\!]_1$:

- signature $\Sigma_{[2]} = \Sigma \times \{0,1\}^2$.
- states: q_0
- final states: q_0
- transitions:

$$\begin{array}{lcll} \langle a,0\rangle(q_0,\ldots,q_0) & \to & q_0 & \\ \langle b,0\rangle(q_0,\ldots,q_0) & \to & q_0 & (b \neq a) \\ \langle a,1\rangle(q_0,\ldots,q_0) & \to & q_0 & \end{array}$$

For $m \geq 1$,

$$[\![X_1 \subseteq L_a]\!]_m := cyl_{1,m}\big([\![X_1 \subseteq L_a]\!]_1\big)$$

compilation: Boolean connectors

- $[\![\phi(X_1, \ldots, X_n) \vee \phi(X_1, \ldots, X_{n'})]\!]_m :=$
  $[\![\phi(X_1, \ldots, X_n)]\!]_m \cup [\![\phi(X_1, \ldots, X_{n'})]\!]_m$
  with $m \geq \max(n, n')$
- $[\![\phi(X_1, \ldots, X_n) \wedge \phi(X_1, \ldots, X_{n'})]\!]_m :=$
  $[\![\phi(X_1, \ldots, X_n)]\!]_m \cap [\![\phi(X_1, \ldots, X_{n'})]\!]_m$
  with $m \geq \max(n, n')$
- $[\![\neg\phi(X_1, \ldots, X_n)]\!]_m := \mathcal{T}(\Sigma_{[m]}) \setminus [\![\phi(X_1, \ldots, X_n)]\!]_m$
  for $m \geq n$.

compilation: quantifiers

- $[\![\exists X_{n+1}\, \phi(X_1, \ldots, X_{n+1})]\!]_n := proj_n\big([\![\phi(X_1, \ldots, X_{n+1})]\!]_{n+1}\big)$
- NB: this construction does not preserve determinism.
- $[\![\exists X_{n+1}\, \phi(X_1, \ldots, X_{n+1})]\!]_m :=$
  $cyl_{n,m}\big([\![\exists X_{n+1}\, \phi(X_1, \ldots, X_{n+1})]\!]_n\big)$ for $m \geq n$.
- $\forall = \neg\exists\neg$

Theorem Thatcher & Wright

Theorem :

For all formula $\phi \in \mathrm{WS}k\mathrm{S}_0$ over Σ without free variables, there exists a tree automaton $\mathcal{A}_\phi$ over Σ, such that $L(\mathcal{A}_\phi) = L(\phi)$.

$\mathcal{A}_\phi = [\![\phi]\!]_0$ can be computed explicitely!

Corollary :

For all formula $\phi \in \mathrm{WS}k\mathrm{S}$ over Σ without free variables there exists a tree automaton $\mathcal{A}_\phi$ over Σ, such that $L(\mathcal{A}_\phi) = L(\phi)$.

using translation of $\mathrm{WS}k\mathrm{S}$ into $\mathrm{WS}k\mathrm{S}_0$ first.

Size of $\mathcal{A}_\phi$

Theorem : Stockmeyer and Meyer 1973

For all n there exists $\exists x_1 \neg \exists y_1 \exists x_2 \neg \exists y_2 \ldots \exists x_n \neg \exists y_n \ \phi \in$ FOL such that for every automaton $\mathcal{A}$ recognizing the same language

$$\left.\begin{array}{l} \\ \\ \end{array}\right\} n \quad \text{size}(\mathcal{A}) \geq 2^{2^{\cdots^{2^{\text{size}(\phi)}}}}$$

Plan

WSkS: Definition

Automata $\rightarrow$ Logic

Logic $\rightarrow$ Automata

Fragments and Extensions of WSkS

Monadic second-order logic of the infinite binary tree (S2S)

WSkS and FO

Using the 2 directions of the Thatcher & Wright theorem:

$$\mathsf{WS}k\mathsf{S} \ni \phi \mapsto \mathcal{A} \mapsto \exists Y_1 \ldots \exists Y_n \, \psi$$

with $\psi \in \mathsf{FOL}$.

Corollary :

Every WSkS formula is equivalent to a formula $\exists Y_1 \ldots \exists Y_n \, \psi$ with ψ first order.

FO $\subsetneq$ WSkS

Proposition :

The language L of terms with an even number of nodes labeled by a is regular (hence WSkS-definable) but not FO-definable.

pr.: with Ehrenfeucht-Fraïssé games.

Ehrenfeucht-Fraïssé games

goal: prove FO equivalence of finite structures
(wrt finite set of predicates $\mathcal{L}$).

Definition

for two finite $\mathcal{L}$-structures $\mathfrak{A}$ and $\mathfrak{B}$ $\mathfrak{A} \equiv_m \mathfrak{B}$ iff for all ϕ closed, of quantifier depth m, $\mathfrak{A} \models \phi$ iff $\mathfrak{B} \models \phi$

Ehrenfeucht-Fraïssé games

$\mathcal{G}_m(\mathfrak{A}, \mathfrak{B})$

1 Spoiler chooses $a_1 \in dom(\mathfrak{A})$ or $b_1 \in dom(\mathfrak{B})$

$1'$ Duplicator chooses $b_1 \in dom(\mathfrak{B})$ or $a_1 \in dom(\mathfrak{A})$

$\vdots$

m' Duplicator chooses $b_m \in dom(\mathfrak{B})$ or $a_m \in dom(\mathfrak{A})$

Duplicator wins if $\{a_1 \mapsto b_1, \ldots, a_m \mapsto b_m\}$ is an injective partial function compatible with the relations of $\mathfrak{A}$ and $\mathfrak{B}$ ($\forall P \in \mathcal{P}$, $P^{\mathfrak{A}}(a_{i_1}, \ldots, a_{i_n})$ iff $P^{\mathfrak{B}}(b_{i_1}, \ldots, b_{i_n})$)
$=$ partial isomorphism.
Otherwise Spoiler wins.

Theorem : Ehrenfeucht-Fraïssé

$\mathfrak{A} \equiv_m \mathfrak{B}$ iff Duplicator has a winning strategy for $\mathcal{G}_m(\mathfrak{A}, \mathfrak{B})$.

Ehrenfeucht-Fraïssé Theorem

more generally: equivalence of finite structures + valuation of n free variables.

for two finite $\mathcal{L}$-structures $\mathfrak{A}$ and $\mathfrak{B}$ and $\alpha_1, \ldots, \alpha_n \in dom(\mathfrak{A})$, $\beta_1, \ldots, \beta_n \in dom(\mathfrak{B})$, $m \geq 0$,

$$\mathfrak{A}, \alpha_1, \ldots, \alpha_n \equiv_m \mathfrak{B}, \beta_1, \ldots, \beta_n$$

iff for all $\phi(x_1, \ldots, x_n)$ of quantifier depth m,

$$\mathfrak{A}, \sigma_a \models \phi(\overline{x}) \text{ iff } \mathfrak{B}, \sigma_b \models \phi(\overline{x})$$

where $\sigma_a = \{x_1 \mapsto \alpha_1, \ldots, x_n \mapsto \alpha_n\}$,
$\sigma_b = \{x_1 \mapsto \beta_1, \ldots, x_n \mapsto \beta_n\}$.

Games: the partial isomorphisms must extend $\{\alpha_1 \mapsto \beta_1, \ldots, \alpha_n \mapsto \beta_n\}$.

FO $\subsetneq$ WSkS

let $\Sigma = \{a : 1, \bot : 0\}$.

Lemma :

For all $m \geq 3$ and all $i, j \geq 2^m - 1$,
Duplicator has a winning strategy for $\mathcal{G}_m(a^i(\bot), a^j(\bot))$.

Corollary :

The language $L \subseteq \mathcal{T}(\Sigma)$ of terms with an even number of nodes labeled by a is not FO-definable.

- Star-free languages $=$ FO definable holds for words [McNaughtonPapert] but not for trees.
- It is an active field of research to characterize regular tree languages definable in FO.
  e.g. [Benedikt Segoufin 05] $\approx$ locally threshold testable.

Restriction to antichains

Definition :

An antichain is a subset $P \subseteq \mathcal{P}os(t)$ s.t. $\forall p, p' \in P$, $p \not< p'$ and $p \not> p'$.

antichain-WSkS: second-order quantifications are restricted to antichains.

Theorem :

If $\Sigma_1 = \emptyset$, the classes of antichain-WSkS languages and regular languages over Σ conincide.

Theorem :

chain-WSkS is strictly weaker than WSkS.

MSO on Graphs

Weak second-order monadic theory of the grid

Σ finite alphabet,

$$\mathcal{L}_{\text{grid}} := \{=, S_{\rightarrow}, S_{\uparrow}, L_a \mid a \in \Sigma\}$$

Grid $G : \mathbb{N} \times \mathbb{N} \rightarrow \Sigma$; Interpretation structure:

$$\underline{G} := \langle \mathbb{N} \times \mathbb{N}, =, x+1, y+1, L_a^G, L_b^G, \ldots \rangle.$$

Proposition :

The weak monadic second-order theory of the grid is undecidable.

csq: weak MSO of graphs is undecidable.

MSO on Graphs (remarks)

- algebraic framework [Courcelle]:
  MSO decidable on graphs generated by a hedge replacement graph grammar $=$ least solutions of equational systems based on graph operations: $\| : 2$, $exch_{i,j} : 1$, $forget_i : 1$, $edge : 0$, $ver : 0$.
- related notion: graphs with bounded *tree width*.
- FO-definable sets of graphs of bounded degree $=$ locally threshold testable graphs (some local neighborhood appears n times with $n <$ threshold - fixed).

Undecidable Extensions

Left concatenation: new predicate

$$S_1' = \{\langle p, 1 \cdot p\rangle \mid p, 1 \cdot p \in \mathcal{P}os(t)\}$$

Proposition :

WS2S + left concatenation predicate is undecidable.

Predicate of equal length.

Proposition :

WS2S + $|x| = |y|$ is undecidable.

Plan

WSkS: Definition

Automata $\rightarrow$ Logic

Logic $\rightarrow$ Automata

Fragments and Extensions of WSkS

Monadic second-order logic of the infinite binary tree (S2S)

S2S

Σ finite alphabet (all symbols have arity 2),

$$\mathcal{L}_\Sigma := \{=, <, S_1, S_2, L_a \mid a \in \Sigma\}.$$

infinite binary tree $t : \{1,2\}^* \to \Sigma$.
Set of infinite binary trees over Σ is denoted $T^\omega(\Sigma)$.

Interpretation structure associated to $t \in T^\omega(\Sigma)$:

...

formulae: same syntax as WS2S.
S2S: interpretation (and quantification) sur des ensembles infinis.

S2S

- $x <_{lex} y \equiv x \leq y \vee \exists z\,(z \cdot 0 \leq x \wedge z \cdot 1 \leq y)$
- $\mathsf{fini}(X) \equiv$
  $\forall Y\,\big(Y \subseteq X \wedge Y \neq \emptyset \Rightarrow$
  $(\exists y\, y \,\text{minimal pour}\, <_{lex} \,\text{dans}\, Y \wedge y \,\text{maximal pour}\, <_{lex} \,\text{dans}\, Y)\big)$

Proposition :

WS2S $\subset$ S2S

WS2S langages: exemples

- $T_0 =$ arbres de $T^\omega(\{0,1\})$ avec un chemin contenant un nombre infini de 1.

  $$\phi_0(X_1) \equiv \exists X \left(\mathsf{path}(X) \wedge \forall x \left(x \in X \Rightarrow \exists y \left(x < y \wedge y \in X \wedge y \in X_1\right)\right)\right)$$

- $T_1 =$ arbres de $T^\omega(\{0,1\})$ dont tous les chemins contiennent un nombre fini de 1.

  $$\phi_1(X_1) \equiv \neg\phi_0(X_1) \equiv$$
  $$\forall X \left(\mathsf{path}(X) \Rightarrow \exists x \left(x \in X \wedge \forall y \left(x < y \wedge y \in X \Rightarrow y \notin X_1\right)\right)\right)$$

Automates de Büchi

$\mathcal{T}^\omega(\Sigma)$ est l'ensemble des arbres binaires infinis étiquetés par Σ (alphabet fini).

Definition : Automate d'arbres de Büchi

Un automate de Büchi sur l'alphabet Σ est un tuple $\mathcal{A} = (Q, q^{\mathsf{i}}, F, \Delta)$ où Q est un ensemble fini d'états, $q^{\mathsf{i}} \in Q$ est l'état initial, $F \subseteq Q$ est le sous-ensemble des états finaux et Δ est un ensemble de règles de transition de la forme: $q \to d(q_0, q_1)$ avec $d \in \Sigma$ et $q, q_0, q_1 \in Q$.

Un *calcul* de $\mathcal{A}$ sur un arbre $t \in T^\omega(\Sigma)$ est un arbre $r \in \mathcal{T}^\omega(Q)$ respectant Δ: pour toute position $p \in \{0,1\}^*$ de r, $r(p) \to t(p)(r(p0), r(p1)) \in \Delta$.

Le calcul r est *acceptant* ssi pour tout chemin (infini) π partant de la racine de r, $Inf(\pi) \cap F \neq \emptyset$

$$Inf(\pi) := \left\{ q \in Q \;\middle|\; \exists^\infty i, \pi(i) = q \right\}$$

Langages de Büchi: exemples

Soit $\Sigma = \{a, b\}$.

1. arbres de $T^\omega(\Sigma)$ avec un nombre infini de a sur chaque chemin,
2. arbres de $T^\omega(\Sigma)$ avec un chemin contenant un nombre infini de a.

Automates de Rabin

Definition : Automate de Rabin

Un automate de Rabin sur l'alphabet Σ est un tuple $\mathcal{A} = (Q, q^{\mathsf{i}}, \Omega, \Delta)$ où Q est un ensemble fini d'états, $q^{\mathsf{i}} \in Q$ est l'état initial, $\Omega = \{(L_1, U_1), \ldots, (L_n, U_n)\}$ $(\forall i \leq n,\ L_i, U_i \subseteq Q)$ et Δ est un ensemble de règles de transition de la forme: $q \to d(q_0, q_1)$ avec $d \in \Sigma$ et $q, q_0, q_1 \in Q$.

Un calcul r est *acceptant* ssi pour tout chemin infini π de r, il existe $i \leq n$ t.q. $Inf(\pi) \cap L_i = \emptyset$ et $Inf(\pi) \cap U_i \neq \emptyset$.

Theorem : Théorème de Rabin

Pour tout automate de Rabin $\mathcal{A}$, il existe un automate de Rabin reconnaissant $\mathcal{T}^{\omega}(\Sigma) \setminus L(\mathcal{A})$.

Rabin vs Büchi

Soit $\Sigma = \{a, b\}$.
Le langage des arbres de $T^\omega(\Sigma)$ dont tous les chemins contiennent un nombre fini de a

- est Rabin-reconnaissable,
- n'est pas Büchi-reconnaissable.

S2S vs WS2S

Proposition :

Un sous-ensemble de $\mathcal{T}^\omega\big(\{0,1\}^n\big)$ est un langage de Büchi ssi il est défini par une formule de S2S de la forme $\exists Y_1 \ldots \exists Y_m \, \phi(Y_1, \ldots, Y_m, X_1, \ldots, X_n)$ avec $\phi \in$ WS2S.

Corollary :

S2S $\not\subset$ WS2S

car les automates (d'arbres) de Büchi ne sont pas clos par complément.

Proposition :

Un sous-ensemble $L \subseteq \mathcal{T}^\omega\big(\{0,1\}^n\big)$ est défini en WS2S ssi L et $\mathcal{T}^\omega\big(\{0,1\}^n\big) \setminus L$ sont des langages de Büchi.

Theorem :

Toute formule de S2S de la forme $\phi(x_1, \ldots, x_m)$ se traduit en WS2S.

Decidable Extensions

- SkS
- SωS
- théorie du second ordre monadique de fonctions d'arité 1 (sur un domaine quelconque dénombrable).

Part IV

Unranked Ordered Labeled Trees

Hedge Automata (HA)

TATA book http://tata.gforge.inria.fr

chapter 8: Automata for Unranked Trees

Tree Automata Techniques and Applications

Hubert Comon Max Dauchet Rémi Gilleron
Florent Jacquemard Denis Lugiez Christof Löding
Sophie Tison Marc Tommasi

- ranked terms = first order terms over a signature
  - every symbols has a fixed arity

→ automated deduction, program analysis, evaluation strategies in functional languages...

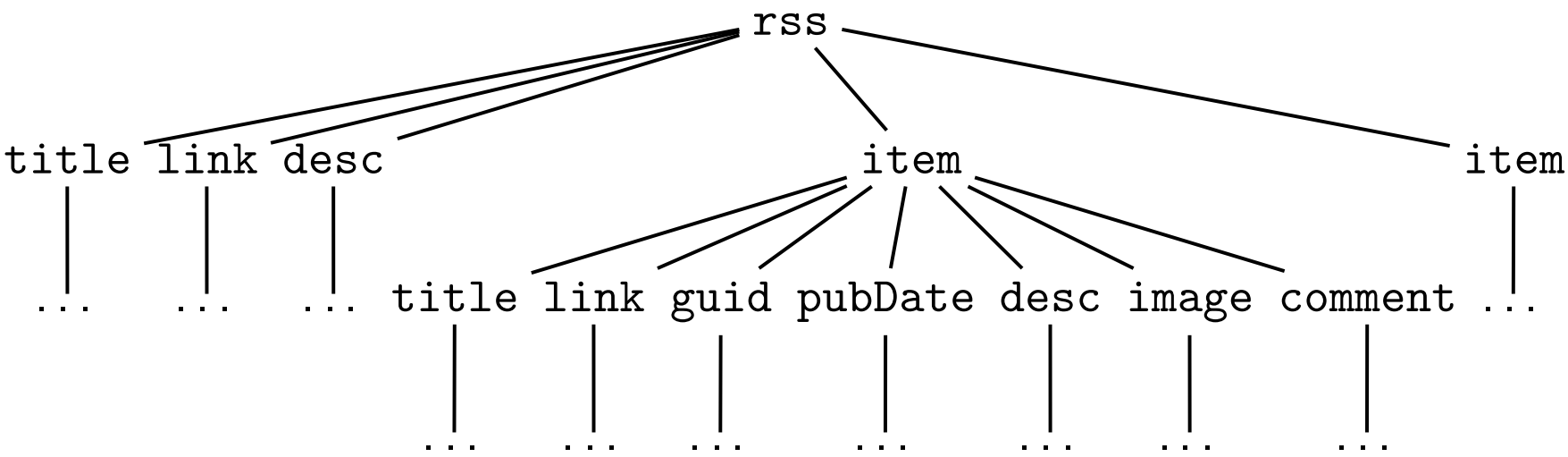
- unranked terms = finite trees (directed, rooted) labelled over a finite alphabet
  - one node can have arbitrarily (though finitely) many childrens
  - the number of children of a node does not depend on its label

→ Web data

Web data (XML Document)

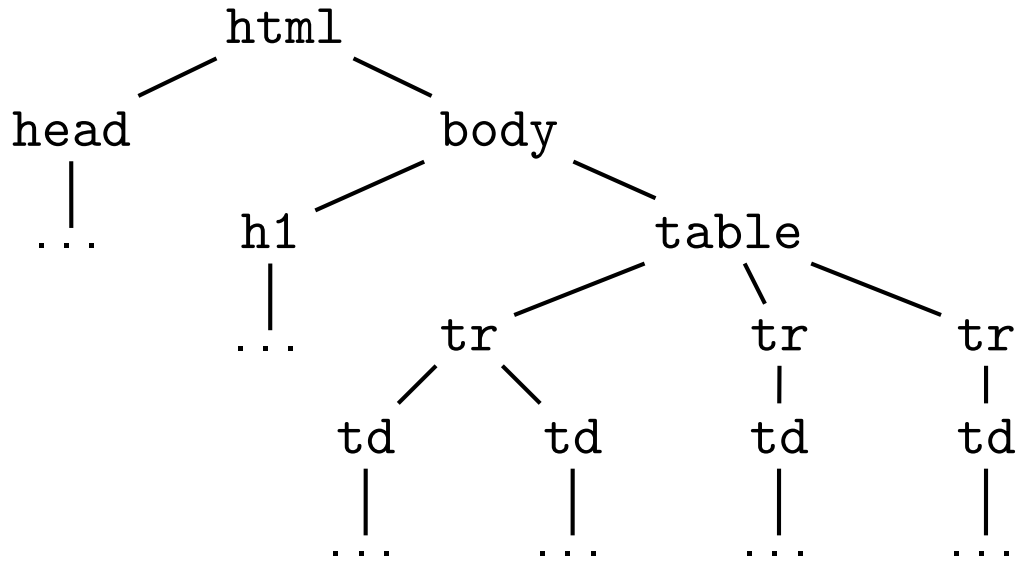
```
<rss version="2.0">
    <title>Mon blog</title>
    <link>http://myblog.blogspot.com</link>
    <description>bla bla bla</description>
    <item>
      <title>Concert</title>
      <link>http://myblog.blogspot.com/me/Mon blog/...</link>
      <guid>5f7da0aa-a593-4a2e</guid>
      <pubDate>Fri, 21 Mar 2009 14:40:02 +0100</pubDate>
      <description>...</description>
      <image href="..."></image>
      <comment link="..." count="0" enabled="0">...</comment>
    </item>
    <item>
      <title>Journée de surf</title>
      ...
    </item>
</rss>
```

Web data


rss
title
link
desc
item
item
...
...
...
title
link
guid
pubDate
desc
image
comment
...
...
...
...
...
...
...
...

HTML Document

```
<html>
 <head>...</head>
 <body>
  <h1>...</h1>
  <table>
   <tr>
    <td>...</td>
    <td>...</td>
   </tr>
   <tr>
    <td>...</td>
   </tr>
   <tr>
    <td>...</td>
   </tr>
  </table>
 </body>
</html>
```



XML Documents

- class of documents with a predefined structure (valid documents)

  ex:

```
<table>
  <tr> <td> c11 </td> <td> c12 </td> </tr>
  <tr> <td> c21 </td> <td> c22 </td> </tr>
</table>
```

- defined by DTD, XML schema... $=$ tree language

XML Documents

ranked	unranked
tree automata	schemas (DTD, XML schema)
membership	validation
emptiness	query satisfiability
inclusion	schema entailment
tree transducers, rewrite systems	transformation languages (XSLT)
rewrite closure	type inference

Plan

Unranked Ordered Trees

Hedge Automata, Determinism

Decision Problems

Binary Encodings

Boolean Closure

Minimization

Weak Second Order Monadic Logic MSO($\rightarrow, \downarrow$)

HA Variants, Regular Languages modulo Associativity

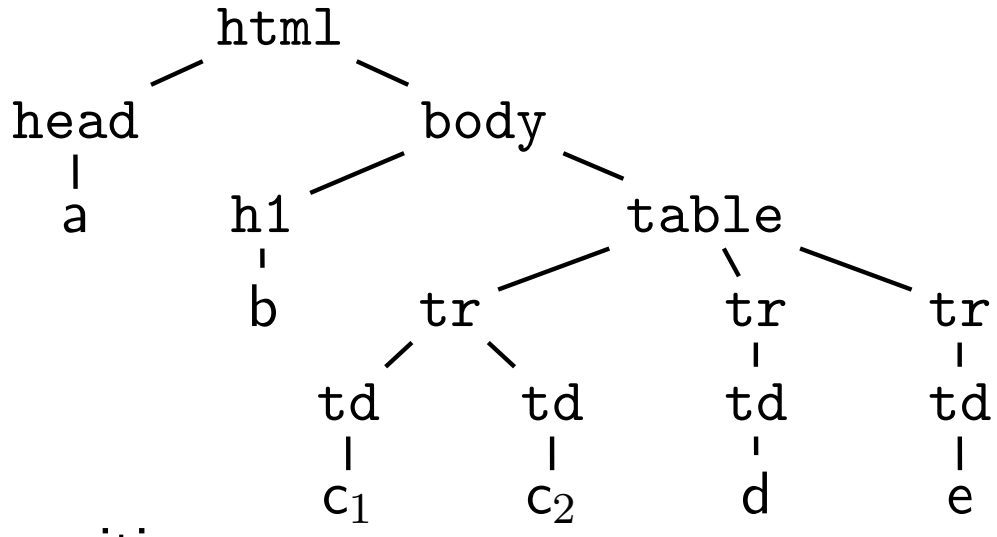
Unranked ordered trees

- Σ is a finite alphabet.

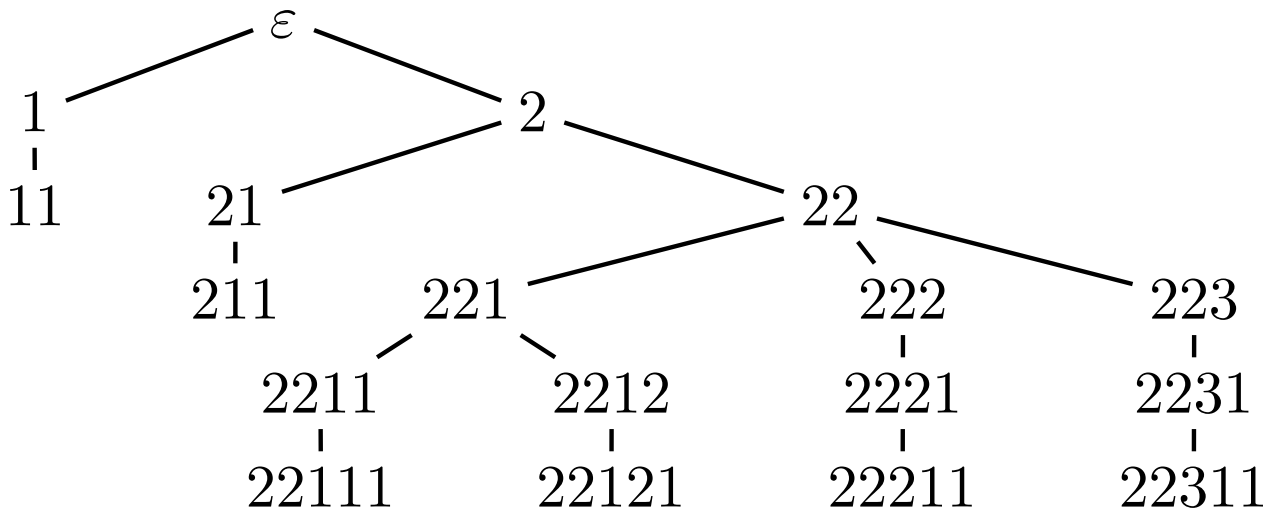
$$\begin{array}{rcll} \text{tree} & := & a(\text{hedge}) & (a \in \Sigma) \\ \text{hedge} & := & \text{tree}^* & \end{array}$$

- a hedge can be empty. $a()$ is denoted by a.
- The set of all unranked ordered trees over Σ is denoted $\mathcal{U}(\Sigma)$.
- The set of hedges over Σ is denoted $\mathcal{H}(\Sigma)$.
- set of positions $\subset \mathbb{N}^*$: as for terms.

Example: tree of $\mathcal{U}(\Sigma)$



positions:



Example: language $\subseteq \mathcal{U}(\Sigma)$

- $\Sigma = \{a, b\}$.
- $L :=$ terms of $\mathcal{U}(\Sigma)$
  - height at most 1,
  - root is labelled by a,
  - even number of leaves, all leaves labelled by b.
- $L = \{a, a(bb), a(bbbb), ...\}$.
- finite description: $L = a\big((bb)^*\big)$

Plan

Unranked Ordered Trees

Hedge Automata, Determinism

Decision Problems

Binary Encodings

Boolean Closure

Minimization

Weak Second Order Monadic Logic MSO($\rightarrow, \downarrow$)

HA Variants, Regular Languages modulo Associativity

Hedge Automata (HA)

Definition : Hedge Automata

A Hedge Automaton (HA) over an alphabet Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ where Q is a finite set of states, $Q^{\mathsf{f}} \subseteq Q$ is the subset of final states and Δ is a set of transition rules of the form: $a(L) \to q$ with $a \in \Sigma$ and $L \subseteq Q^*$ is a regular language.

A run of $\mathcal{A}$ on $t \in \mathcal{U}(\Sigma)$ is a tree $r \in \mathcal{U}(Q)$ such that

- r and t have the same domain,
- for all $p \in \mathcal{P}os(t)$, with $t(p) = a$, $r(p) = q$, there exists $a(L) \to q \in \Delta$ such that $r(p1) \dots r(pn) \in L$, where n is the number of successors of p in $\mathcal{P}os(t)$.

The run r is accepting (successful) iff $r(\varepsilon) \in Q^{\mathsf{f}}$.

HA Languages

- language of $\mathcal{A}$: $L(\mathcal{A})$ is the set of terms on which there exists an accepting run of $\mathcal{A}$,
- language of $\mathcal{A}$ in state $q \in Q$: $L(\mathcal{A}, q)$ is the set of terms t such that there exists a run r of $\mathcal{A}$ on t with $r(\varepsilon) = q$,
- $L(\mathcal{A}) = \bigcup_{q \in Q_f} L(\mathcal{A}, q)$.

- equivalently, $L(\mathcal{A}, q)$ is the smallest set of terms $a(t_1, \ldots, t_n) \in \mathcal{U}(\Sigma)$ $(n \geq 0)$ such that there exists a transition $a(L) \to q$, and states $q_1, \ldots, q_n \in Q$ with $t_i \in L(\mathcal{A}, q_i)$ s.t. $i \leq n$ and $q_1 \ldots q_n \in L$.

HA language: Example 1

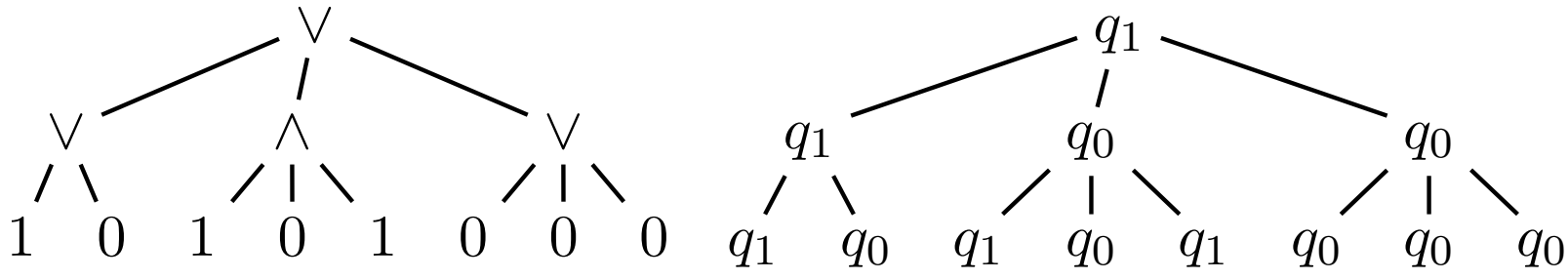
- $\Sigma = \{a, b\}$.
- $L :=$ terms of $\mathcal{U}(\Sigma)$
  - height 1,
  - root is labelled by a,
  - even number of leaves, all leaves labelled by b.
- $L = \{a, a(bb), a(bbbb), ...\}$.
- finite description: $L = L(\mathcal{A})$ with
  $\mathcal{A} := \big(\Sigma, \{q_a, q_b\}, \{q_a\}, \{b \to q_b, a((q_b q_b)^*) \to q_a\}\big)$.

Boolean expressions with variadic $\wedge$ and $\vee$

- $\Sigma = \{\wedge, \vee, 0, 1\}$,
- states $\{q_0, q_1\}$,
- transitions:

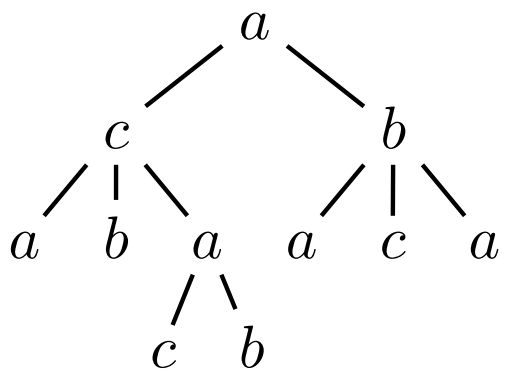
$$
\begin{array}{rcl}
0 & \to & q_0 \\
\wedge(q_1^*\, q_0\, (q_0 \mid q_1)^*) & \to & q_0 \\
\vee(q_0\, q_0^*) & \to & q_0 \\
\neg(q_0) & \to & q_1
\end{array}
\qquad
\begin{array}{rcl}
1 & \to & q_1 \\
\wedge(q_1\, q_1^*) & \to & q_1 \\
\vee(q_0^*\, q_1\, (q_0 \mid q_1)^*) & \to & q_1 \\
\neg(q_1) & \to & q_0
\end{array}
$$

- example: Boolean expression and associated run



HA language: Example 3

- $\Sigma = \{a, b, c\}$.
- $L :=$ terms of $\mathcal{U}(\Sigma)$
  - with 2 b's at positions p_1 and p_2, and
  - one c on the smallest common ancestor of p_1 and p_2.

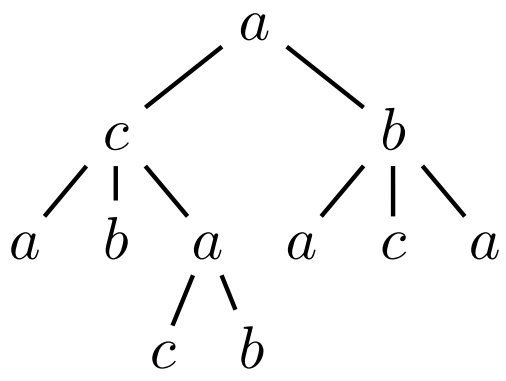


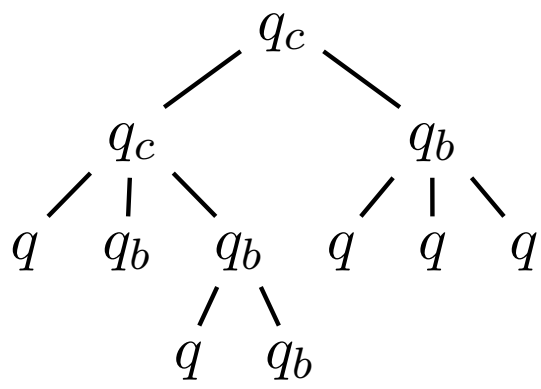
HA language: Example 3

- $\Sigma = \{a, b, c\}$.
- $L :=$ terms of $\mathcal{U}(\Sigma)$
  - with 2 b's at positions p_1 and p_2, and
  - one c on the smallest common ancestor of p_1 and p_2.

$\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$, avec $Q = \{q, q_b, q_c\}$, $Q^{\mathsf{f}} = \{q_c\}$, $\Delta =$

$$
\begin{array}{llll}
a(Q^*) \to q & a(Q^* q_b Q^*) \to q_b & a(Q^* q_c Q^*) \to q_c \\
b(Q^*) \to q_b & c(Q^* q_b Q^*) \to q_b & b(Q^* q_c Q^*) \to q_c \\
c(Q^*) \to q & c(Q^* q_b Q^* q_b Q^*) \to q_c & c(Q^* q_c Q^*) \to q_c
\end{array}
$$





Normalized Hedge Automata

Definition :

A HA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ over Σ is called normalized if for all $a \in \Sigma$ and $q \in Q$, there is at most one transition of the form $a(L) \to q$ in Δ.

When $\mathcal{A}$ is normalized, we denote $a(L_{a,q}) \to q$ the unique transition with a and q.

Proposition :

For all HA $\mathcal{A}$, there exista a normalized HA $\mathcal{A}_{\mathsf{n}}$ recognizing the same language.

The size of $\mathcal{A}_{\mathsf{n}}$ is linear in the size of $\mathcal{A}$.

Complete and deterministic hedge automata

Semantical definitions

Definition :

A HA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ over Σ is complete if for all $t \in \mathcal{U}(\Sigma)$, there exists at least one state $q \in Q$ s.t. $t \in L(\mathcal{A}, q)$.

Definition :

A HA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ over Σ is deterministic if for all $t \in \mathcal{U}(\Sigma)$, there exists at most one state $q \in Q$ s.t. $t \in L(\mathcal{A}, q)$.

Complete and deterministic hedge automata

Syntactical definitions

Definition :

A HA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ over Σ is complete if
for all $a \in \Sigma$ and all finite sequence $q_1, \ldots, q_n \in Q^*$,
there exists a transition $a(L) \to q \in \Delta$ with $q_1 \ldots q_n \in L$.

Definition :

A HA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ over Σ is deterministic if
for all transitions $a(L_1) \to q_1$ and $a(L_2) \to q_2$ in Δ,
either $L_1 \cap L_2 = \emptyset$, or $q_1 = q_2$.

Determinism: examples

HA for Boolean expressions evaluation : deterministic.
$\Sigma = \{\wedge, \vee, 0, 1\}$, $Q = \{q_0, q_1\}$ et Δ:

$$\begin{array}{rcl} 0 & \to & q_0 \\ \wedge(q_1^*\, q_0\, (q_0 \mid q_1)^*) & \to & q_0 \\ \vee(q_0 q_0^*) & \to & q_0 \end{array} \qquad \begin{array}{rcl} 1 & \to & q_1 \\ \wedge(q_1 q_1^*) & \to & q_1 \\ \vee(q_0^*\, q_1\, (q_0 \mid q_1)^*) & \to & q_1 \end{array}$$

Language with 2 b's and common ancestor c: not deterministic.
$\Sigma = \{a, b, c\}$, $Q = \{q, q_b, q_c\}$, $Q^{\mathsf{f}} = \{q_c\}$, Δ:

$$\begin{array}{rcl} a(Q^*) & \to & q \\ b(Q^*) & \to & q_b \\ c(Q^*) & \to & q \end{array} \qquad \begin{array}{rcl} a(Q^* q_b Q^*) & \to & q_b \\ c(Q^* q_b Q^*) & \to & q_b \\ c(Q^* q_b Q^* q_b Q^*) & \to & q_c \end{array} \qquad \begin{array}{rcl} a(Q^* q_c Q^*) & \to & q_c \\ b(Q^* q_c Q^*) & \to & q_c \\ c(Q^* q_c Q^*) & \to & q_c \end{array}$$

HA completion

Proposition :

For all HA $\mathcal{A}$, there exists a complete HA $\mathcal{A}_c$ recognizing the same language.

The size of $\mathcal{A}_c$ is linear is the size of $\mathcal{A}$.

pr.: add a *trash* state $q_\perp$ and transitions :

$$a\Big(\bigcap_{q\in Q} Q^* \setminus L_{a,q} \cup Q_\perp^* q_\perp Q_\perp^*\Big) \to q_\perp$$

HA determinization

Proposition :

For all HA $\mathcal{A}$, there exists a deterministic HA $\mathcal{A}_\mathsf{d}$ recognizing the same language.

The size of $\mathcal{A}_\mathsf{d}$ is exponential in the size of $\mathcal{A}$ (lower bound).

pr.: subset construction

Plan

Unranked Ordered Trees

Hedge Automata, Determinism

Decision Problems

Binary Encodings

Boolean Closure

Minimization

Weak Second Order Monadic Logic MSO($\rightarrow, \downarrow$)

HA Variants, Regular Languages modulo Associativity

HA: membership decision

Proposition : $\in$

The problem of membership is decidable in polynomial time for the HAs whose languages $L_{a,q}$ are given by NFAs.

- linear time for DHA whose languages $L_{a,q}$ are given by DFA,
- NP-complete if the languages $L_{a,q}$ are given by alternating automata.

HA: emptiness decision

Proposition : $\emptyset$

The problem of emptiness is decidable in polynomial time for HAs whose languages $L_{a,q}$ are given by NFAs.

pr.: state marking. $M \subseteq Q$,

- initialy, $M = \emptyset$.
- at each step , if $a(L_{a,q}) \to q \in \Delta$ and $L_{a,q} \cap M^* \neq \emptyset$ then $M := M \cup \{q\}$.

- PSPACE-complete if the languages $L_{a,q}$ are given by alternating automata.

HA: decision of inclusion and equivalence

Proposition : $\subseteq$, $\equiv$

The problem of inclusion (resp. equivalence) is EXPTIME-complete for HAs whose languages $L_{a,q}$ are given by NFAs.

pr.: $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff $L(\mathcal{A}_1) \cap (\mathcal{U}(\Sigma) \setminus L(\mathcal{A}_2)) = \emptyset$.

- in PTIME for DHAs whose languages $L_{a,q}$ are given by DFAs.
- PSPACE-complete for DHAs whose languages $L_{a,q}$ are given by alternating automata.

Plan

Unranked Ordered Trees

Hedge Automata, Determinism

Decision Problems

Binary Encodings

Boolean Closure

Minimization

Weak Second Order Monadic Logic MSO$(\rightarrow, \downarrow)$

HA Variants, Regular Languages modulo Associativity

Currying

Transformation into binary trees with @ and constants.

We associate the following signature to an alphabet Σ:

$$\Sigma_@ := \{a : 0 \mid a \in \Sigma\} \cup \{@ : 2\}$$

The function curry : $\mathcal{U}(\Sigma) \to \mathcal{T}(\Sigma_@)$, is defined recursively:

- $\mathsf{curry}(a) := a$,
- $\mathsf{curry}\big(a(t_1, \ldots, t_n)\big) := @\big(\mathsf{curry}\big(a(t_1, \ldots, t_{n-1})\big), \mathsf{curry}(t_n)\big)$.

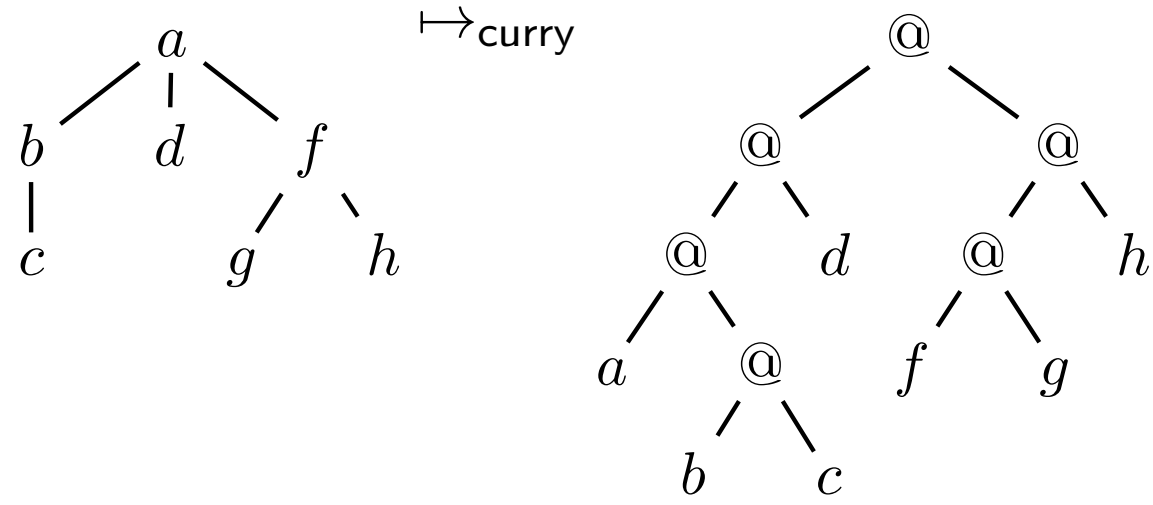
binary operator on $\mathcal{U}(\Sigma)$

$$a(t_1 \ldots t_n) \mathbin{@} t_{n+1} = a(t_1 \ldots t_{n+1})$$

$$\mathsf{curry}(t \mathbin{@} t') = @\big(\mathsf{curry}(t), \mathsf{curry}(t')\big)$$

Currying: example 1

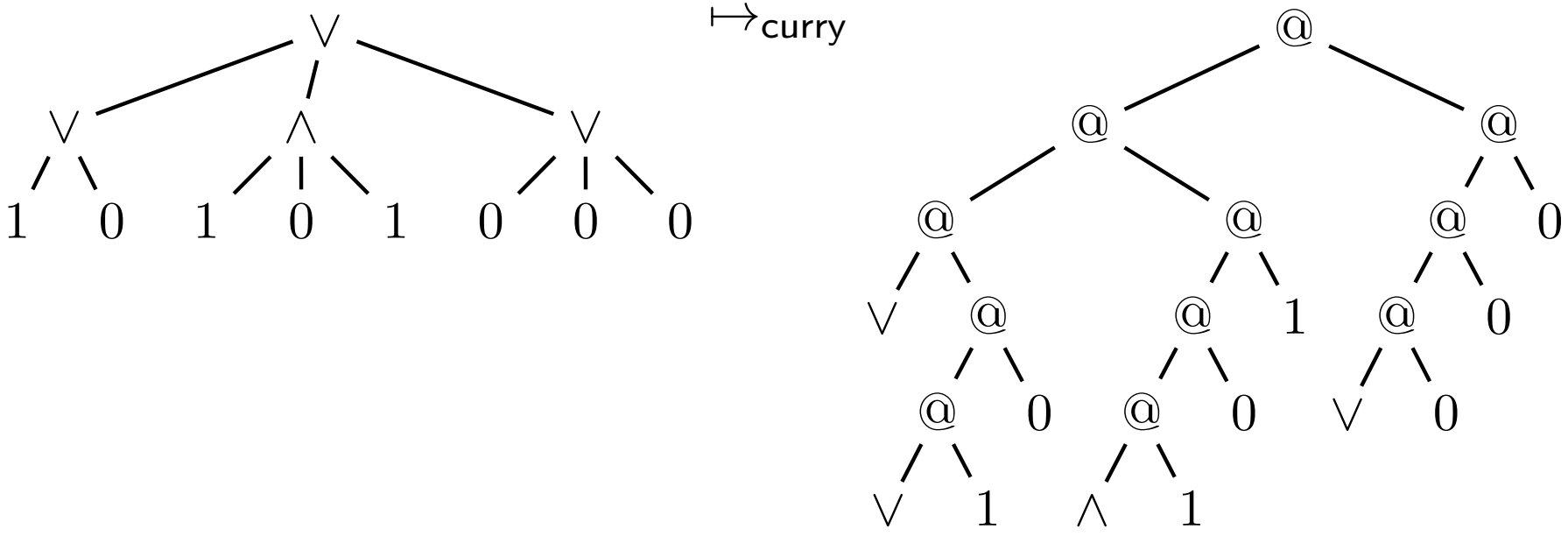
$$\mathsf{curry}\big(a(t_1,\ldots,t_n)\big) := @\big(\mathsf{curry}\big(a(t_1,\ldots,t_{n-1})\big),\mathsf{curry}(t_n)\big)$$



Currying: example 2

$\mathsf{curry}\big(a(t_1,\ldots,t_n)\big) := @\big(\mathsf{curry}\big(a(t_1,\ldots,t_{n-1})\big), \mathsf{curry}(t_n)\big)$

image of unranked Boolean expressions.



Currying: properties

Lemma :

curry is a bijection from $\mathcal{U}(\Sigma)$ into $\mathcal{T}(\Sigma_@)$.

Proposition :

$L \subseteq \mathcal{U}(\Sigma)$ is a HA language iff curry(L) is regular.

Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta \rangle$ be normalized and let $B_{a,q}$ be the NFA recognizing L in the (unique) transition $a(L) \to q$ of Δ associated to $a \in \Sigma$ and $q \in Q$ (its input alphabet is Q). Let P be the union (assumed disjoint) for all transitions in Δ of the state sets of the automata $B_{a,q}$.

We construct an automaton $\mathcal{A}'$ whose transitions, when computing on curry(t) for some $t \in \mathcal{U}(\Sigma)$, will simulate both the transitions of $\mathcal{A}$ (vertical transitions) and the transitions of the NFAs $B_{a,q}$ (horizontal transitions).

Let $\mathcal{A}' = \langle Q \cup P, \Sigma, F, \Delta' \rangle$ where Δ' contains the transitions:

- $a \to q$ if $B_{a,q}$ recognizes the empty word,
- $a \to \text{init}_{a,q}$ for all $q \in Q$, where $init_{a,q}$ is the initial state of $B_{a,q}$,
- $@(p, q) \to p'$ if there is a transition $p \xrightarrow{q} p'$ in some $B_{a',q'}$, and
- $@(p, q) \to q'$ if there is a transition $p \xrightarrow{q} p'$ in some $B_{a',q'}$, where p' is a final state of $B_{a',q'}$.

We can show by induction on $t \in \mathcal{U}(\Sigma)$ that $t \in L(\mathcal{A})$ iff curry$(t) \in L(\mathcal{A}')$.

Note that $\mathcal{A}'$ is non deterministic.

Plan

- Unranked Ordered Trees
- Hedge Automata, Determinism
- Decision Problems
- Binary Encodings
- Boolean Closure
- Minimization
- Weak Second Order Monadic Logic MSO$(\rightarrow, \downarrow)$
- HA Variants, Regular Languages modulo Associativity

HA: Boolean operations

Proposition :

The class of HA languages is closed under union, intersection and complement.

HA: closure under morphisms

projection $h : \mathcal{U}(\Sigma) \to \mathcal{U}(\Sigma')$, defined by extension to trees of an application $h : \Sigma \to \Sigma'$.

$$h(L) = \{h(t) \mid t \in L\} \quad \text{and} \quad h^{-1}(L') = \{t \in \mathcal{U}(\Sigma) \mid h(t) \in L'\}$$

Proposition :

The class of HA languages is closed under projections and inverse projections.

Plan

Unranked Ordered Trees

Hedge Automata, Determinism

Decision Problems

Binary Encodings

Boolean Closure

Minimization

Weak Second Order Monadic Logic MSO$(\rightarrow, \downarrow)$

HA Variants, Regular Languages modulo Associativity

Minimization

2 questions must be addressed

1. for which definition of determinism?
   (minimization makes sense only for deterministic automata)
2. what to minimize?

For ranked tree automata, the answer to both questions is clear:

1. ranked DTA: every step of computation is deterministic

2. we want to minimize the number of states.

For unranked tree automata, this is not so clear:

1. even for DHA[DFA] (DHA whose horizontal languages are defined by DFA), if we have $a(L) \to q$ and $a(L') \to q'$ and $L \cap L' = \emptyset$, in configuration $a(q_1 \dots q_n)$ we must test both $q_1 \dots q_n \in L$ and $q_1 \dots q_n \in L'$ before firing the right transition. In the construction of a TA for curry(HA), we choose ND.

2. there are states for the DHA and for the DFAs for the horizontal languages.

Minimization of DHA

First approach: we ignore the formalism for horizontal languages, i.e. we chose

1. DFAs (whatever for the horizontal automata)
2. number of states of the DFA

Congruence of a language $L \subseteq \mathcal{U}(\Sigma)$:

$$s \equiv_L t \quad \text{iff} \quad \forall C\ C[s] \in L \Leftrightarrow C[t] \in L$$

Minimal DHA for the HA language L:

- states: $\left\{[t]_{\equiv_L} \mid t \in \mathcal{U}(\Sigma)\right\}$,
- final states: $\left\{[t]_{\equiv_L} \mid t \in L\right\}$ (we simply write $[t]$ below),
- transitions:
  $\left\{a(L_{a,[t]}) \to [t] \mid L_{a,[t]} = \{[t_1] \dots [t_n] \mid a(t_1 \dots t_n) \equiv_L t\}\right\}$,

Minimization of DHA (2)

2 drawbacks for the first approach

- the complexity of the effective construction depends on the formalism for horizontal languages.
- no analogous of Myhill-Nerode theorem for DFA or DTA (ranked):

$$L \text{ is an HA language} \quad \Rightarrow \quad \equiv_L \text{ has finite index}$$
$$L \text{ is an HA language} \quad \not\Leftarrow \quad \equiv_L \text{ has finite index}$$

Minimization of DHA[DTA]

Second approach: we consider both vertical and horizontal states and transitions, i.e. we chose

1. DFA[DTA] (DHA whose horizontal language are defined by disjoint DFAs)
2. number of states of the DFA + number of states of the horizontal (disjoint) DTAs

Minimization of DHA[DTA] (2)

first idea: use the curry encoding and minimize the ranked TA.

problem: the TA associated to an HA wrt curry is not deterministic; we have $a \to \text{init}_{a,q}$ for all $q \in Q$.
other question: uniqueness of minimal automaton?

$\to$ stepwise automata: one unique transition from each $a \in \Sigma$ to the start state of a deterministic machine that will read the state sequence below a and output a state.
Moreover, vertical states $=$ horizontal states.

Deterministic Stepwise Automata

Definition : stepwise automata

A deterministic stepwise hedge automaton (DSHA) is a tuple $\mathcal{A} = (\Sigma, Q, Q_f, \delta_0, \delta)$, where Σ, Q, and Q_f are as usual, $\delta_0 : \Sigma \to Q$ is a function assigning to each letter of the alphabet an initial state, and $\delta : Q \times Q \to Q$ is the transition function.

For $a \in \Sigma$,
$$\begin{aligned} \delta_a : & \quad Q^* \to Q \\ \delta_a(\varepsilon) &= \delta_0(a) \\ \delta_a(w \cdot q) &= \delta(\delta_a(w), q) \end{aligned}$$

A run of $\mathcal{A}$ on $t \in \mathcal{U}(\Sigma)$ is a tree $r \in \mathcal{U}(Q)$ such that

- r and t have the same domain,
- for all $p \in \mathcal{P}os(t)$, $r(p) = \delta_{t(p)}(r(p1) \dots r(pn))$, where n is the number of successors of p in $\mathcal{P}os(t)$.

The run r is accepting (successful) iff $r(\varepsilon) \in Q^{\mathsf{f}}$.

Stepwise Automata & Ranked TA

stepwise DSHA $\mathcal{A}$	ranked DTA curry($\mathcal{A}$)
$\delta_0(a) = q$	$a \to q$
$\delta(q_1, q_2) = q$	$@(q_1, q_2) \to q$

Lemma :

For all $t, t' \in \mathcal{U}(\Sigma)$ and $q, q' \in Q$,
if $t \in L(\mathcal{A}, q)$ and $t' \in L(\mathcal{A}, q')$, then $t \,@\, t' \in L(A, \delta(q, q'))$.

Lemma :

For all DSHA $\mathcal{A}$, curry$(L(\mathcal{A})) = L($curry$(\mathcal{A}))$.

Minimal Stepwise Automata

Corollary :

DSHA recognize all HA unranked tree languages.

Corollary :

For each HA language $L \subseteq \mathcal{U}(\Sigma)$ there is a unique (up to renaming of states) minimal DSHA accepting L.

Plan

Unranked Ordered Trees

Hedge Automata, Determinism

Decision Problems

Binary Encodings

Boolean Closure

Minimization

Weak Second Order Monadic Logic MSO($\rightarrow, \downarrow$)

HA Variants, Regular Languages modulo Associativity

Logic and Automata

- logic expresses properties of labeled trees
  = specification of language,
- compilation of formulae into automata
  = decision algorithms.
- equivalence between both formalisms
  Thatcher & Wright's theorem for ranked trees and ranked TA.
  analogous for unranked trees and HA.

Unranked term as structure

Σ finite alphabet

$$\mathcal{L}_\Sigma := \{=, \downarrow, \rightarrow, L_a \mid a \in \Sigma\}.$$

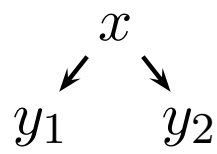
to $t \in \mathcal{U}(\Sigma)$, we associate a structure $\underline{t}$ over $\mathcal{L}_\Sigma$

$$\underline{t} := \langle \mathcal{P}os(t), =^{\underline{t}}, \downarrow^{\underline{t}}, \rightarrow^{\underline{t}}, L_a^{\underline{t}}, L_b^{\underline{t}}, \cdots \rangle$$

$$\begin{array}{rcl}
\text{domain} & := & \text{positions of } t \quad (\mathcal{P}os(t) \subset \mathbb{N}^*), \\
=^{\underline{t}} & := & \text{equality over } \mathcal{P}os(t), \\
\downarrow^{\underline{t}} & := & \left\{ \langle p, p \cdot i \rangle \mid i \in \mathbb{N}, p, p \cdot i \in \mathcal{P}os(t) \right\} \text{ (parent relation)}, \\
\rightarrow^{\underline{t}} & := & \left\{ \langle p \cdot i, p \cdot i + 1 \rangle \mid i \in \mathbb{N}, p, p \cdot i, p \cdot i + 1 \in \mathcal{P}os(t) \right\} \\
 & & \text{(next sibling)}, \\
L_a^{\underline{t}} & := & \{p \in \mathcal{P}os(t) \mid t(p) = a\}.
\end{array}$$

MSO$(\rightarrow, \downarrow)$: syntaxe

Rappel: WS2S, $S_1(x, y_1) \wedge S_2(x, y_2)$ for



formulas of MSO$(\rightarrow, \downarrow)$:

- first order variables $x \ldots$
- second order variables $X \ldots$
- term ::= x
- form ::= $x = y \mid x \downarrow y \mid x \rightarrow y \mid x \in X \mid L_a(x) \quad a \in \Sigma$
  $\mid$ form $\wedge$ form $\mid$ form $\vee$ form $\mid \neg$form
  $\mid \exists x$ form $\mid \exists X$ form $\mid \forall x$ form $\mid \forall X$ form

MSO($\rightarrow, \downarrow$): semantics

- $t \in \mathcal{U}(\Sigma)$,
- valuation σ of first order variables into $\mathcal{P}os(t)$,
- valuation δ of second order variables into subsets of $\mathcal{P}os(t)$,
- $\underline{t}, \sigma, \delta \models x = y$ iff $\sigma(x) = \sigma(y)$,
- $\underline{t}, \sigma, \delta \models x \downarrow y$ iff $\exists i \in \mathbb{N}$, $\sigma(y) = \sigma(x) \cdot i$,
- $\underline{t}, \sigma, \delta \models x \rightarrow y$ iff $\exists p_0 \in \mathbb{N}^*$, $i \in \mathbb{N}$, $\sigma(x) = p_0 \cdot i$ and $\sigma(y) = p_0 \cdot i + 1$,
- $\underline{t}, \sigma, \delta \models x \in X$ iff $\sigma(x) \in \delta(X)$,
- $\underline{t}, \sigma, \delta \models L_a(x)$ iff $t(\sigma(x)) = a$,
- $\underline{t}, \sigma, \delta \models \phi_1 \wedge \phi_2$ iff $\underline{t}, \sigma, \delta \models \phi_1$ and $\underline{t}, \sigma, \delta \models \phi_2$,
- $\underline{t}, \sigma, \delta \models \phi_1 \vee \phi_2$ iff $\underline{t}, \sigma, \delta \models \phi_1$ or $\underline{t}, \sigma, \delta \models \phi_2$,
- $\underline{t}, \sigma, \delta \models \neg\phi$ iff $\underline{t}, \sigma, \delta \not\models \phi$,

MSO($\rightarrow, \downarrow$): semantics (quantifiers)

- $\underline{t}, \sigma, \delta \models \exists x\ \phi$ iff $x \notin dom(\sigma)$, x free in ϕ and exists $p \in \mathcal{P}os(t)$ s.t. $\underline{t}, \sigma \cup \{x \mapsto p\}, \delta \models \phi$,
- $\underline{t}, \sigma, \delta \models \forall x\ \phi$ iff $x \notin dom(\sigma)$, x free in ϕ and for all $p \in \mathcal{P}os(t)$, $\underline{t}, \sigma \cup \{x \mapsto p\}, \delta \models \phi$,
- $\underline{t}, \sigma, \delta \models \exists X\ \phi$ iff $X \notin dom(\delta)$, X free in ϕ and exists $P \subseteq \mathcal{P}os(t)$ s.t. $\underline{t}, \sigma, \delta \cup \{X \mapsto P\} \models \phi$,
- $\underline{t}, \sigma, \delta \models \forall X\ \phi$ iff $X \notin dom(\delta)$, X free in ϕ and for all $P \subseteq \mathcal{P}os(t)$, $\underline{t}, \sigma, \delta \cup \{X \mapsto P\} \models \phi$.

Second order monadic logic: examples

- root:

  $x = \varepsilon \equiv \forall y \, x \downarrow^* y$, or $\neg \exists y \, y \downarrow x$,
- leaf:

  $\mathsf{leaf}(x) \equiv \neg \exists y \, x \downarrow y$
- first child:

  $\mathsf{first}(x) \equiv \neg \exists y \, y \rightarrow x$
- last child:

  $\mathsf{last}(x) = \neg \exists y \, x \rightarrow y$

Second order monadic logic: examples (2)

- prefixe ordering $=$ transitive closure of $\downarrow$:

  $x \downarrow^* y \equiv$
  $\forall X \left(y \in X \wedge \forall z \, \forall z' \, (z \downarrow z' \wedge z' \in X \Rightarrow z \in X) \right) \Rightarrow x \in X$

  or $x \downarrow^* y \equiv \exists X \, x \in X \wedge y \in X \wedge$
  $\forall z \left[(z \in X \wedge z \neq x) \Rightarrow \exists z' \, (z' \in X \wedge z' \downarrow z) \right]$

- transitive closure of $\rightarrow$:

  $x \xrightarrow{*} y \equiv$
  $\forall X \left(y \in X \wedge \forall z \, \forall z' \, (z \rightarrow z' \wedge z' \in X \Rightarrow z \in X) \right) \Rightarrow x \in X$

  or $x \xrightarrow{*} y \equiv \exists X \, x \in X \wedge y \in X \wedge$
  $\forall z \left[(z \in X \wedge z \neq x) \Rightarrow \exists z' \, (z' \in X \wedge z' \rightarrow z) \right]$

Defining languages of $\mathcal{U}(\Sigma)$ in MSO$(\rightarrow, \downarrow)$

$\Sigma := \{a_1, \ldots, a_n\}$,

Definition : MSO$(\rightarrow, \downarrow)$-definability

For $\phi \in$ MSO$(\rightarrow, \downarrow)$ without free variables over $\mathcal{L}_\Sigma$,
$L(\phi) := \left\{ t \in \mathcal{U}(\Sigma) \;\middle|\; \underline{t} \models \phi \right\}$.

Theorem :

A subset of $\mathcal{U}(\Sigma)$ is a HA language iff it is definable in MSO$(\rightarrow, \downarrow)$.

HA $\to$ MSO$(\to, \downarrow)$

Let $\Sigma = \{a_1, \ldots, a_n\}$.

Theorem :

For all hedge automaton $\mathcal{A}$ over Σ, there exists $\phi_{\mathcal{A}} \in \mathrm{MSO}(\to, \downarrow)$ such that $L(\phi_A) = L(\mathcal{A})$.

Let $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$, normalized, with $Q = \{q_1, \ldots, q_m\}$, and $P = \{p_1, \ldots, p_h\}$ disjoint union of state sets of NFAs for the horizontal languages $L_{a,q} \subseteq Q^*$, $a \in \Sigma$, $q \in Q$.
$\phi_{\mathcal{A}}$: existence of an accepting run of $\mathcal{A}$ on $t \in \mathcal{U}(\Sigma)$.

$$\phi_{\mathcal{A}} := \exists Y_1 \ldots \exists Y_m \, \exists Z_0 \ \ldots \exists Z_h \, \phi(\overline{Y}, \overline{Z})$$

HA $\rightarrow$ MSO$(\rightarrow, \downarrow)$

$$
\begin{aligned}
&\phi_{\mathcal{A}} \equiv Y_1, \ldots, Y_m \text{ partition } \mathcal{P}os(t) \\
&\wedge Z_0 = \{\varepsilon\} \wedge Z_0, Z_1, \ldots, Z_h \text{ partition } \mathcal{P}os(t) \\
&\wedge \bigvee_{q_i \in Q^{\mathsf{f}}} \varepsilon \in Y_i \\
&\wedge \forall x \bigvee_{a(L) \rightarrow q_i \in \Delta} a(x) \wedge x \in Y_i \wedge \\
&\forall y \left(\begin{array}{ll}
x \downarrow y \wedge \mathsf{first}(y) & \Rightarrow \bigvee_{p_k = \mathsf{initial}(L)} y \in Z_k \\
\wedge \exists y'\ x \downarrow y \wedge y \rightarrow y' & \Rightarrow \\
& \bigvee_{p_k \xrightarrow{q_j} p_{k'} \in L} (y \in Z_k \wedge y \in Y_j \wedge y' \in Z_{k'}) \\
\wedge (x \downarrow y \wedge \mathsf{last}(y)) & \Rightarrow \\
& \bigvee_{p_{k'} \in \mathsf{final}(L)} \bigvee_{p_k \xrightarrow{q_j} p_{k'} \in L} (y \in Z_k \wedge y \in Y_j)
\end{array} \right)
\end{aligned}
$$

MSO$(\rightarrow, \downarrow) \rightarrow$ HA

Theorem :

Every subset of $\mathcal{U}(\Sigma)$ definable in MSO$(\rightarrow, \downarrow)$ is a HA language.

For all formula $\phi \in$ MSO$(\rightarrow, \downarrow)$ over Σ (without free variables) there exists a hedge automaton $\mathcal{A}_\phi$ over Σ, such that $L(\mathcal{A}_\phi) = L(\phi)$.

Corollary :

MSO$(\rightarrow, \downarrow)$ is decidable.

pr.: reduction to emptiness decision for $\mathcal{A}_\phi$.

Operator "first child next sibling"

Correspondence with algebras of ranked terms (symbols of arity 0 or 2): representation of unranked terms as binary terms with pointers and lists.

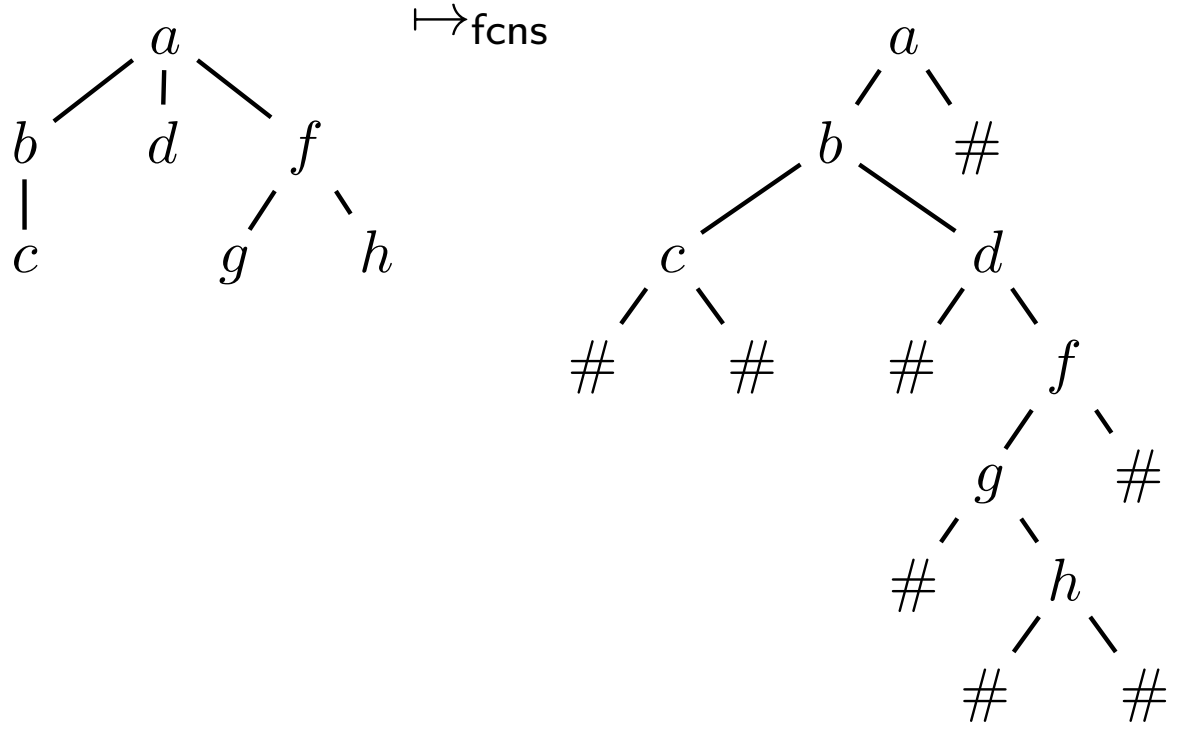
We associate the following signature to an alphabet Σ:

$$\Sigma_{\#} := \{a : 2 \mid a \in \Sigma\} \cup \{\# : 0\}$$

The operator fcns : $\mathcal{H}(\Sigma) \to \mathcal{T}(\Sigma_{\#})$, is defined recursively:

- $\mathsf{fcns}(a) := a(\#, \#)$,
- $\mathsf{fcns}\big(a(t_1, \ldots, t_n)\big) := a\big(\mathsf{fcns}(t_1, \ldots, t_n), \#\big)$,
- $\mathsf{fcns}(t_1, \ldots, t_n) := \mathsf{fcns}(t_1)[\mathsf{fcns}(t_2, \ldots, t_n)]_2$ if $n \geq 2$.

Operator "first child next sibling": example



Operator "first child next sibling": properties

Lemma :

fcns is a bijection from $\mathcal{H}(\Sigma)$ into $\mathcal{T}(\Sigma_{\#})$.

For all $t \in \mathcal{T}(\Sigma_{\#})$, $\text{fcns}^{-1}(t) \in \mathcal{U}(\Sigma)$ iff $t|_2 = \#$.

Proposition :

If $L \subseteq \mathcal{U}(\Sigma)$ is a HA language, then $\text{fcns}(L)$ is regular.

Proposition :

If $L \subseteq \mathcal{T}(\Sigma_{\#})$ is regular, then $\text{fcns}^{-1}(L) \cap \mathcal{U}(\Sigma)$ is a HA language.

Let $\mathcal{A} = \langle Q, Q_\mathsf{f}, \Delta \rangle$ a HA over Σ, s.t. every $L_{a,q}$ ($a \in \Sigma$, $q \in Q$) recognized by $\mathcal{B}_{a,q} = \langle P_{a,q}, \mathsf{init}_{a,q}, F_{a,q}, R_{a,q} \rangle$. $P = \bigcup_{a,q} P_{a,q}$ and $F = \bigcup_{a,q} F_{a,q}$. We construct $\mathcal{A}' \langle Q', Q'_\mathsf{f}, \Delta' \rangle$, a TA over $\Sigma_\#$

- $Q' = P \cup \{q_\mathsf{f}, q_\#\}$,
- $Q'_\mathsf{f} = \{q_\mathsf{f}\}$,
- Δ' contains
  - $\# \to q_\#$,
  - $\# \to p$ for all $p \in F$,
  - $b(\mathsf{init}_{b,q'}, p') \to p$ if $\exists a, q \; p \xrightarrow[\Delta_{a,q}]{q'} p'$,
  - $a(\mathsf{init}_{a,q}, q_\#) \to q_\mathsf{f}$ if $q \in Q_\mathsf{f}$.

$t_i \xrightarrow[\mathcal{A}]{*} q_i$, $p \xrightarrow[\mathcal{B}_{a,q}]{q_1 \dots q_n} p' \in F_{a,q}$ iff $\mathsf{fcns}(t_1 \dots t_n) \xrightarrow[\mathcal{A}']{*} p$ (induction on the hedge).

MSO$(\rightarrow, \downarrow)$ $\rightarrow$ HA

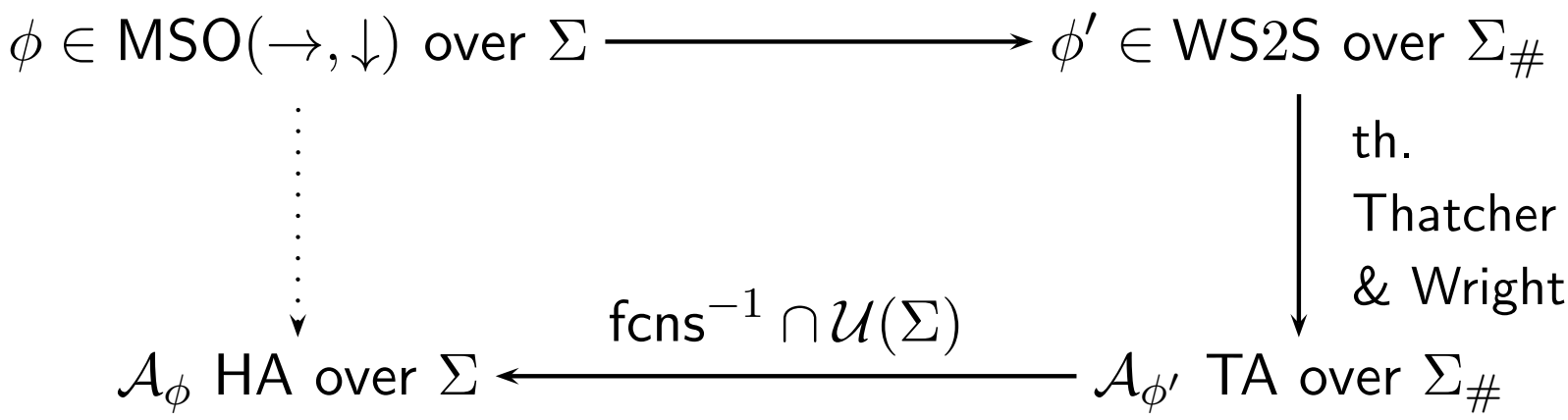
$$\begin{array}{ccc} \phi \in \mathsf{MSO}(\rightarrow, \downarrow) \text{ over } \Sigma & \mapsto & \phi' \in \mathsf{WS2S} \text{ over } \Sigma_{\#} \\ \mathsf{fcns}(L(\phi)) & = & L(\phi') \end{array}$$

$$\phi' \equiv L_{\#}(\varepsilon \cdot 2) \wedge \psi$$

ψ obtained from ϕ by replacements of atoms:

ϕ	ψ
$x \downarrow y$	$y \in x \cdot 1 \cdot 2^*$
$x \rightarrow y$	$y = x \cdot 2$

$$y \in x{\cdot}1{\cdot}2^* \equiv \exists X \, (x{\cdot}1 \in X \wedge y \in X \wedge \forall z \, (z \in X \wedge z \neq y \Rightarrow z{\cdot}2 \in X))$$



Plan

Unranked Ordered Trees

Hedge Automata, Determinism

Decision Problems

Binary Encodings

Boolean Closure

Minimization

Weak Second Order Monadic Logic MSO$(\rightarrow, \downarrow)$

HA Variants, Regular Languages modulo Associativity

Extensions of HA

Definition : CF-HA

A CF-HA is a tuple (Σ, Q, Q_f, Δ), where Q, Q_f are as for HA and the transitions of Δ have the form $a(L) \to q$ with $a \in \Sigma$, $q \in Q$, and $L \subseteq Q^*$ is a context-free language.

Definition : CS-HA

A CS-HA is a tuple (Σ, Q, Q_f, Δ) where Q, Q_f are as for HA and the transitions of Δ have the form $a(L) \to q$ with $a \in \Sigma$, $q \in Q$, and $L \subseteq Q^*$ is a context-sensitive language.

CF-HA: example

$\Sigma = \{a, b, f\}$, language L of trees of $\mathcal{U}(\Sigma)$:

- whose internal nodes are labeled by f,
- with the same number of leaves a than leaves b under every node.

language of the CF-HA $(Q, Q_{\mathsf{f}}, \Delta)$ with $Q = \{q, q_a, q_b\}$ and

$$\Delta = \{a \to q_a, \quad b \to q_b, \quad f(L) \to q\}$$

L is the language generated by the context-free grammar

$$\begin{array}{lll} N & := & \varepsilon \mid \text{ all permutations of } NN_aN_b \mid q \\ N_a & := & q_a \quad N_b := q_b \end{array}$$

Rem.: L is not a HA language.

Regular Languages modulo A (ATA)

Signature $\Sigma = \Sigma_\emptyset \uplus \Sigma_{\mathsf{A}}$.
The symbols of Σ_{A} are binary and follow the associativity axiom:

$$a(x_1, a(x_2, x_3)) = a(a(x_1, x_2), x_3) \qquad \text{(A)}$$

Given a TA $\mathcal{B}$ over Σ, we note

$$\mathsf{A}\big(L(\mathcal{B})\big) := \big\{ t \in \mathcal{T}(\Sigma) \;\big|\; t \overset{*}{\underset{\mathsf{A}}{\leftrightarrow}} s \in L(\mathcal{B}) \big\}$$

(ATA language)

Proposition :

- the class of regular tree languages is strictly included in the class of ATA languages.
- The class of ATA languages is not closed under intersection.

Correspondences $\mathcal{T}(\Sigma) \leftrightarrow \mathcal{U}(\Sigma)$

Let $\Sigma_{\mathsf{A}} = \{a\}$.

$$\begin{array}{rrcl} \mathit{flat}: & \mathcal{T}(\Sigma) & \to & \mathcal{U}(\Sigma) \\ \mathit{hflat}: & \mathcal{T}(\Sigma)^* & \to & \mathcal{H}(\Sigma) \\ \mathit{flat}^{-1}: & \mathcal{U}(\Sigma) & \to & \mathcal{T}(\Sigma) \end{array}$$

Definitions ($g \in \Sigma_n \setminus \Sigma_{\mathsf{A}}$):

$$\begin{array}{rcl} \mathit{flat}\big(g(t_1,\ldots,t_n)\big) & = & g(\mathit{flat}(t_1)\ldots\mathit{flat}(t_n)) \\ \mathit{flat}\big(a(t_1,t_2)\big) & = & a(\mathit{hflat}(t_1\,t_2)) \\ \\ \mathit{hflat}\big(g(s_1,\ldots,s_n)\,t_2\ldots t_m\big) & = & \mathit{flat}\big(g(s_1,\ldots,s_n)\big)\,\mathit{hflat}(t_2\ldots t_m) \\ \mathit{hflat}\big(a(s_1,s_2)\,t_2\ldots t_m\big) & = & \mathit{hflat}(s_1 s_2\,t_2\ldots t_m) \\ \\ \mathit{flat}^{-1}\big(g(t_1\ldots t_n)\big) & = & g(\mathit{flat}^{-1}(t_1),\ldots,\mathit{flat}^{-1}(t_n)) \\ \mathit{flat}^{-1}\big(a(t_1\ldots t_m)\big) & = & a(\mathit{flat}^{-1}(t_1), a(\mathit{flat}^{-1}(t_2),\ldots, \\ & & a(\mathit{flat}^{-1}(t_{m-1}),\mathit{flat}^{-1}(t_m)))) \\ & & (m \geq 2) \end{array}$$

CF-HA $\leftrightarrow$ ATA

Proposition :

CF-HA $\equiv$ ATA

- $\subseteq$ for all CF-HA $\mathcal{A}$ there exists a TA $\mathcal{B}$ such that $L(\mathcal{B}) = \mathit{flat}^{-1}\big(L(\mathcal{A})\big)$.
- $\supseteq$ for all TA $\mathcal{B}$ there exists a CF-HA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathit{flat}\big(\mathrm{A}(L(\mathcal{B}))\big)$.

CF-HA: Results

Proposition :

The class of CF-HA languages is not closed under intersection and complementation.

Proposition : $\emptyset$

The emptiness problem is decidable in polynomial time for CF-HA.

Proposition : $\in$

The membership problem is decidable in polynomial time for CF-HA.

pr.: Consequences of the correspondence with ATA and results for these languages.

Generalized Tree Automata (GTA)

Definition : GTA

A *generalized Tree Automata* (GTA) over a signature Σ is a tuple $\mathcal{B} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ where Q is a finite set of *states*, $Q^{\mathsf{f}} \subseteq Q$ is the subset of final states and Δ is a set of transition rules of the form $f(q_1, \ldots, q_n) \to q$ or $f(q_1, \ldots, q_n) \to f(q'_1, \ldots, q'_n)$ with $f \in \Sigma_n$ $(n \geq 0)$ and $q_1, \ldots, q_n, q'_1, \ldots, q'_n \in Q$.

For a TA or GTA $\mathcal{B}$, we define the languages:

$$\begin{array}{rcl} L(\mathcal{B}) & := & \{t \in \mathcal{T}(\Sigma) \;\big|\; t \xrightarrow[\Delta]{*} q \in Q^{\mathsf{f}}\} \\ L_{\mathsf{A}}(\mathcal{B}) & := & \{t \in \mathcal{T}(\Sigma) \;\big|\; t \xrightarrow[\Delta/\mathsf{A}]{*} q \in Q^{\mathsf{f}}\} \end{array}$$

where $\xrightarrow[\Delta/\mathsf{A}]{*}$ is rewriting modulo A: $\xrightarrow[\Delta/\mathsf{A}]{*} := \xleftrightarrow[\mathsf{A}]{*} \circ \xrightarrow[\Delta]{} \circ \xleftrightarrow[\mathsf{A}]{*}$

Languages of TA and GTA modulo A

Proposition :

For all GTA $\mathcal{B}$, there exists a TA $\mathcal{B}'$ such that $L(\mathcal{B}') = L(\mathcal{B})$.

Proposition :

For all TA $\mathcal{B}$, $L_{\mathsf{A}}(\mathcal{B}) = \mathsf{A}\big(L(\mathcal{B})\big)$.

Csq: the emptiness of $L_{\mathsf{A}}(\mathcal{B})$ for a TA $\mathcal{B}$ is decidable.

Proposition :

For a GTA $\mathcal{B}$, in general $L_{\mathsf{A}}(\mathcal{B}) \neq \mathsf{A}\big(L(\mathcal{B})\big)$.

CS-HA $\leftrightarrow$ GTA modulo A

Proposition :

CS-HA $\equiv$ AGTA

- $\subseteq$ for all CS-HA $\mathcal{A}$ there exists a GTA $\mathcal{B}$ such that $L_{\mathsf{A}}(\mathcal{B}) = \mathsf{A}\big(\mathit{flat}^{-1}\big(L(\mathcal{A})\big)\big)$.
- $\supseteq$ for all GTA $\mathcal{B}$ there exists a CS-HA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathit{flat}\big(L_{\mathsf{A}}(\mathcal{B})\big)$.

GTA & CS-HA: decision results

Proposition : $\in$ CS-HA

The membership problem is PSPACE-complete for CS-HA.

Proposition : $\in$ GTA

The membership problem $t \in L_{\mathsf{A}}(\mathcal{B})$ given a GTA $\mathcal{B}$ is PSPACE-complete.

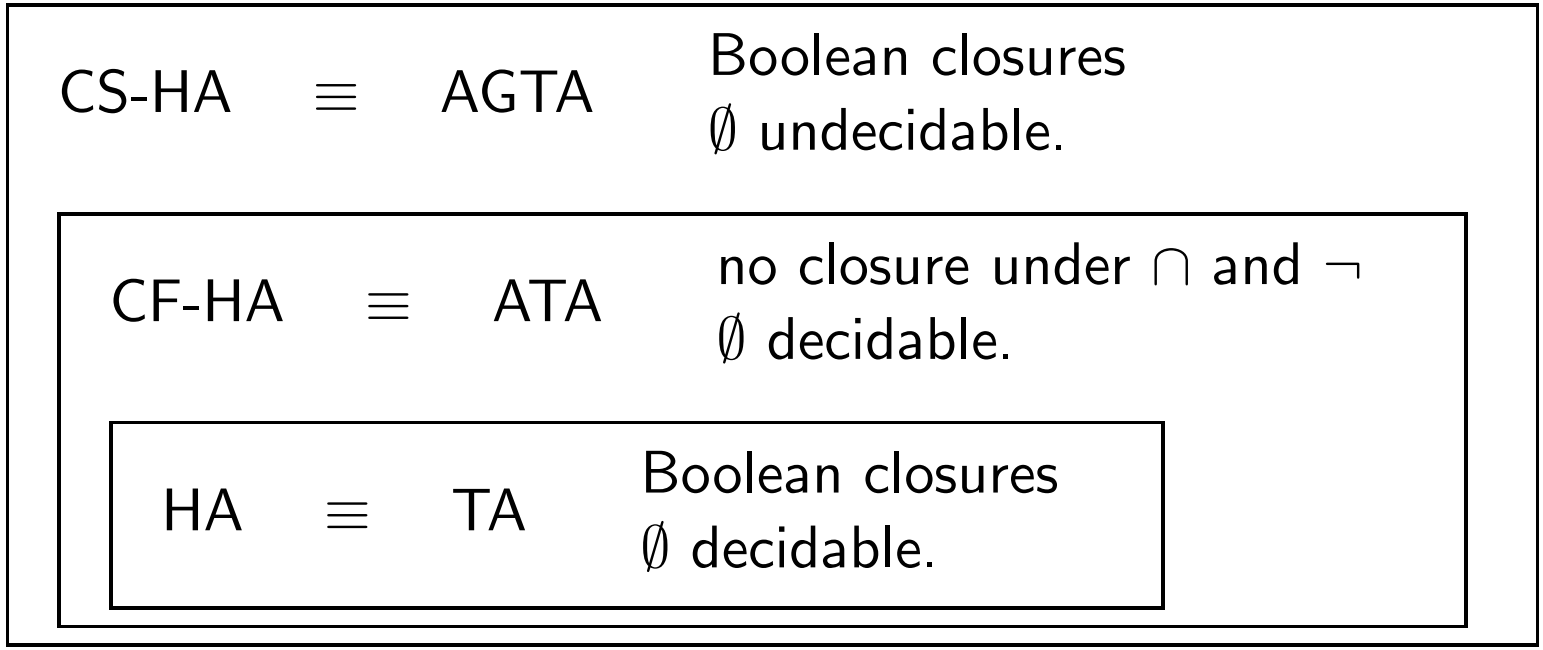
Proposition : $\emptyset$ CS-HA

The emptiness problem is undecidable for CS-HA.

Proposition : $\emptyset$ GTA

The emptiness problem $L_{\mathsf{A}}(\mathcal{B})$ given a GTA $\mathcal{B}$ is undecidable.

Summary


CS-HA ≡ AGTA
Boolean closures
∅ undecidable.
CF-HA ≡ ATA
no closure under ∩ and ¬
∅ decidable.
HA ≡ TA
Boolean closures
∅ decidable.

Part V

Unranked Unordered Labeled Trees

Presburger Automata (PA)

- previous lecture: unranked ordered trees
  - XML documents
  - hedge automata (HA), $\mathrm{MSO}(\rightarrow, \downarrow)$.
  - cf. TATA chap. 8.
- this lecture: unranked unordered trees
  - web data
  - Presburger automata (PA), PMSO.
  - cf. article "Numerical Document Queries", Helmut Seidl, Thomas Schwentick, Anca Muscholl, 2003.

Plan

Unranked Unordered Trees

Presburger Arithmetic

Presburger Automata (PA)

Determinism

Boolean Closure

Decision Problems

Regular Languages modulo Associativity and Commutativity

Weak Second Order Monadic Logic PMSO

Unranked Unordered Trees

- Σ is a finite alphabet.

$$\begin{array}{rcl} \mathsf{tree} & := & a(\mathsf{multiset}) \quad (a \in \Sigma) \\ \mathsf{multiset} & := & \{\mathsf{tree}, \ldots, \mathsf{tree}\} \end{array}$$

- $a\big(\{t_1, \ldots, t_n\}\big)$ is denoted $a(t_1, \ldots, t_n)$.
- rem: the multiset can be empty. $a(\emptyset)$ is denoted a.
- The set of unranked unordered trees over Σ is denoted $\mathcal{U}_2(\Sigma)$.

Examples of languages of trees of $\mathcal{U}_2(\Sigma)$

- $\Sigma = \{a, b\}$.
- terms of $\mathcal{U}_2(\Sigma)$
  - of height 1,
  - with one b at the root,
  - the leaves are a or b:

  i. with an even number of a (HA),

  ii. with the same even number of a than b ($\neg$ HA).

Plan

Unranked Unordered Trees

Presburger Arithmetic

Presburger Automata (PA)

Determinism

Boolean Closure

Decision Problems

Regular Languages modulo Associativity and Commutativity

Weak Second Order Monadic Logic PMSO

Presburger Arithmetic

Presburger Formulae:

$$
\begin{array}{lll}
\text{term} & ::= & x \quad \text{(first order variables)} \\
 & | & n \quad \text{(natural number)} \\
 & | & \text{term} + \text{term} \\
\text{form} & ::= & \text{term} = \text{term} \\
 & | & \neg\text{form} \mid \text{form} \vee \text{form} \mid \text{form} \wedge \text{form} \\
 & | & \forall x\, \text{form} \mid \exists x\, \text{form}
\end{array}
$$

Interpretation in the domain of natural numbers.
$(n_1, \ldots, n_p) \models \phi(x_1, \ldots, x_p)$ ($x_1, \ldots, x_p$ free variables)
iff $\phi(n_1, \ldots, n_p)$ is evaluated to *true*.

Presburger Arithmetic

- notation: terms $n\,x$ for $\underbrace{x + \ldots + x}_{n}$,
- the natural can be restricted to 0 and 1,
- the atoms can be restricted to $x = n$ and $x = y + z$.

Examples:

- $x \leq y \equiv \exists x'\, y = x + x'$.
- $odd(x) \equiv \exists y\, x = y + y + 1$.

Presburger Arithmetic

Theorem : Presburger Arithmetic

Presburger Arithmetic is decidable.

- lower bound 2-EXPTIME (Fischer and Rabin 1974)
- upper bound 3-EXPTIME (Klaedtke 2004, with an automata construction)
- NP-complete for the existential fragment.

Presburger Arithmetic

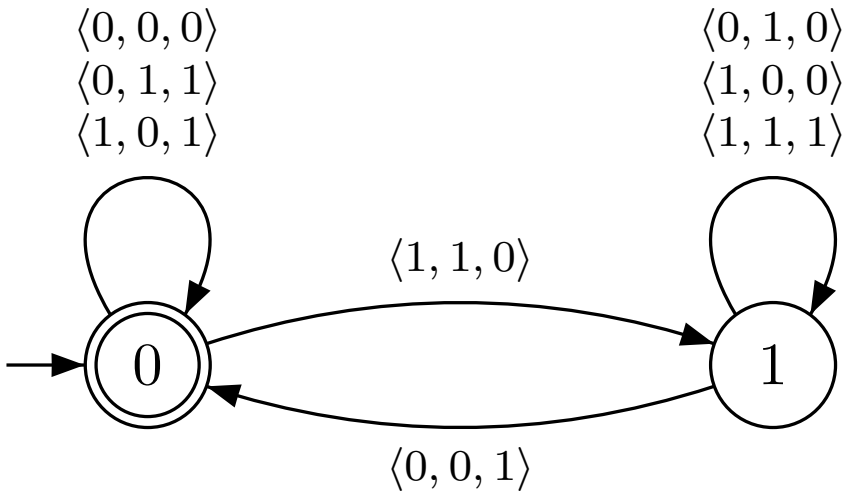
Decidability of Presburger Arithmetic.

We can associate to a formula $\phi(x_1, \ldots, x_p)$ a finite automaton over the alphabet $\{0,1\}^p$ recognizing the set of $\langle b_{1,1}, \ldots, b_{p,1}\rangle \ldots \langle b_{1,k}, \ldots, b_{p,k}\rangle$ such that $b_{1,1} \ldots b_{1,k}$, $\ldots$, $b_{p,1} \ldots b_{p,k}$ are the binary representations of integers $n_1, \ldots, n_p$ satisfying ϕ.

Hence we can decide wether there exists $n_1, \ldots, n_p$ such that $(n_1, \ldots, n_p) \models \phi(x_1, \ldots, x_p)$.

Presburger Arithmetic and Automata

finite automaton for $x_1 + x_2 = x_3$



Semi-linear sets

Definition :

- A *linear* set is a subset of $\mathbb{N}^p$ of the form $\left\{\overline{v_0} + \overline{v_1} + \ldots + \overline{v_m} \;\middle|\; m \geq 0, \overline{v_1}, \ldots, \overline{v_m} \in B\right\}$, for $\overline{v_0} \in \mathbb{N}^p$ and $B \subset \mathbb{N}^p$ finite (fixed).
- A *semi-linear* set is a finite union of linear sets.

models of Presburger formulae $\equiv$ semi-linear sets.

Projection and Parikh theorem

$\Sigma = \{a_1, \ldots, a_p\}$.

Definition : Parikh Projection

The Parikh projection of a word $w \in \Sigma^*$ is the tuple $\#(w) := (m_1, \ldots, m_p)$ where m_i $(i \leq p)$ is the number of occurrences of a_i in w.

For a set $L \subseteq \Sigma^*$, we denote $\#(L) := \{\#(w) \mid w \in L\}$.

Theorem :

For all context-free language $L \subseteq \Sigma^*$, there exists a Presburger formula $\phi(x_1, \ldots, x_p)$ such that $\#(L) := \{(n_1, \ldots, n_p) \models \phi(x_1, \ldots, x_p)\}$.

When L is regular, the Presburger formula is computed in linear time (in the size of the NFA defining L).

Conversely, given a Presburger formula $\phi(x_1, \ldots, x_p)$, one can build a NFA $\mathcal{A}$ such that
$\#\big(L(\mathcal{A})\big) = \{(n_1, \ldots, n_k) \models \phi(x_1, \ldots, x_k)\}$.

Plan

- Unranked Unordered Trees
- Presburger Arithmetic
- Presburger Automata (PA)
- Determinism
- Boolean Closure
- Decision Problems
- Regular Languages modulo Associativity and Commutativity
- Weak Second Order Monadic Logic PMSO

Presburger Automata (PA)

Definition : Presburger Automata

A *Presburger Automaton* (PA) over an alphabet Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ where $Q = \{q_1, \ldots, q_p\}$ is a finite set od *states*, $Q^{\mathsf{f}} \subseteq Q$ is the subset of final states and Δ is a set of transition rules of the form: $a(\phi) \to q$ with $a \in \Sigma$, $q \in Q$, and $\phi = \phi(x_1, \ldots, x_p)$ is a Presburger formula with one free variable x_i for each state q_i.

The language of $\mathcal{A}$ in state $q \in Q$, denoted $L(\mathcal{A}, q)$, is the smallest subset of terms $a(t_1, \ldots, t_n) \in \mathcal{U}_2(\Sigma)$ such that

- there exists $i_1, \ldots, i_n \leq p$ such that for all $j \leq n$, $t_j \in L(\mathcal{A}, q_{i_j})$,
- there exists a transition $a(\phi) \to q \in \Delta$ such that $\#(q_{i_1}, \ldots, q_{i_n}) \models \phi(x_1, \ldots, x_p)$.

The language of $\mathcal{A}$ is $L(\mathcal{A}) = \bigcup_{q \in Q_{\mathsf{f}}} L(\mathcal{A}, q)$.

PA: example 1

$\Sigma = \{a, b, f\}$.
Set of trees of $\mathcal{U}_2(\Sigma)$ where all the a and b label the leaves:

$$\mathcal{A} = \big(\{q\}, \{q\}, \{a(x_q = 0) \to q, b(x_q = 0) \to q, f(true) \to q\}\big)$$

PA: example 2

$\Sigma = \{a, b, c\}$.
Set of trees of $\mathcal{U}_2(\Sigma)$ with the same number of a and of b under each node.

$$\mathcal{A} = \big(\{q_a, q_b, q\}, \{q_a, q_b, q\}, \{a(\phi) \to q_a, b(\phi) \to q_b, c(\phi) \to q\}\big)$$

with $\phi \equiv x_{q_a} = x_{q_b}$.

PA: example 3

$\Sigma = \{a, b\}$.
Set of trees of $\mathcal{U}_2(\Sigma)$ where all internal node are labeled by a and every node labeled by a has at least as much sons without b than sons containing b.

$Q = Q_{\mathsf{f}} = \{q_a, q_b\}$.
The state q_b accepts the trees containing a b and q_a accepts the others.

$$\Delta = \left\{ \begin{array}{llll} a(x_{q_a} \geq x_{q_b} = 0) & \to & q_a & a(x_{q_a} \geq x_{q_b} > 0) \to q_b \\ & & & b(x_{q_a} = x_{q_b} = 0) \to q_b \end{array} \right\}$$

Normalized Presburger automata

Definition :

A PA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ sur Σ is called *normalized* if for all $a \in \Sigma$ and $q \in Q$, there is at most one transition of the form $a(\phi) \to q$ in Δ.

When $\mathcal{A}$ is normalized, we note $a(\phi_{a,q}) \to q$ the unique transition with a and q.

Proposition :

For all PA $\mathcal{A}$, there exists a normalized PA $\mathcal{A}_n$ recognizing the same language.

The size of $\mathcal{A}_n$ is linear in the size of $\mathcal{A}$.

Plan

Unranked Unordered Trees

Presburger Arithmetic

Presburger Automata (PA)

Determinism

Boolean Closure

Decision Problems

Regular Languages modulo Associativity and Commutativity

Weak Second Order Monadic Logic PMSO

Complete and deterministic Presburger automata

Definition : Deterministic PA

A PA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ over Σ is *deterministic* if for all $a \in \Sigma$ and all $q_{i_1}, \ldots, q_{i_n} \in Q^*$, if $a(\phi) \to q \in \Delta$ and $a(\phi') \to q' \in \Delta$ are such that $\#(q_{i_1}, \ldots, q_{i_n}) \models \phi$ and $\#(q_{i_1}, \ldots, q_{i_n}) \models \phi'$, then $q = q'$.

Lemma :

The determinism of PA is decidable.

Definition : PA complets

A PA $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ over Σ is *complete* if for all $a \in \Sigma$ and all $q_{i_1}, \ldots, q_{i_n} \in Q^*$ there exists at least one transition $a(\phi) \to q \in \Delta$ such that $\#(q_{i_1}, \ldots, q_{i_n}) \models \phi$.

Completion of PA

Proposition : Completion

For all PA $\mathcal{A}$, there exists a complete PA $\mathcal{A}_c$ recognizing the same language.

The size of $\mathcal{A}_c$ is linear in the size of $\mathcal{A}$.

pr.: Let $\mathcal{A} = (Q, Q_f, \Delta)$ be a PA (normalized).
We add the state $q_\perp$: $\mathcal{A}_c = \big(Q \cup \{q_\perp\}, Q_f, \Delta_c\big)$ with

$$\begin{array}{rcll} \Delta_c & := & a\big(\phi_{a,q} \wedge x_{q_\perp} = 0\big) & \rightarrow \quad q \text{ s.t. } a(\phi_{a,q}) \rightarrow q \in \Delta \\ & \cup & a\big(\bigwedge_{q \in Q} \neg\phi_{a,q} \vee x_{q_\perp} > 0\big) & \rightarrow \quad q_\perp \end{array}$$

Determinization of PA

Proposition : Determinization

For all PA $\mathcal{A}$, there exists a deterministic PA $\mathcal{A}_d$ recognizing the same language.

The size of $\mathcal{A}_d$ is exponential in the size of $\mathcal{A}$ (lower bound).

pr.: Let $\mathcal{A} = (Q, Q_\mathsf{f}, \Delta)$, normalized with $Q = \{q_1, \ldots, q_b\}$.

$$\mathcal{A}_d = \left(2^Q, \{S \subseteq Q \mid S \cap Q_\mathsf{f} \neq \emptyset\}, \Delta_d\right)$$

Presburger formulae in Δ_d: a free variable x_S for each $S \subseteq Q$.

ε-transitions and elimination

Remark :

PA with transitions $q \to q' \equiv$ PA.

Plan

Unranked Unordered Trees

Presburger Arithmetic

Presburger Automata (PA)

Determinism

Boolean Closure

Decision Problems

Regular Languages modulo Associativity and Commutativity

Weak Second Order Monadic Logic PMSO

PA: operations Boolean

Proposition :

The class of languages of PA is closed under intersection and complementation.

- $\cup$ disjoint union (linear) or product (quadratic, preserves determinism).
- $\cap$ de Morgan law or product (quadratic).
- $\neg$ complete, determinize, invert final states (exponentiel).

Plan

Unranked Unordered Trees

Presburger Arithmetic

Presburger Automata (PA)

Determinism

Boolean Closure

Decision Problems

Regular Languages modulo Associativity and Commutativity

Weak Second Order Monadic Logic PMSO

PA: decision of membership

Proposition : $\in$

The problem of membership is decidable for PA.

- in polynomial time for DPA,
- NP-complete for NPA.

PA: emptiness decision

Proposition : $\emptyset$

The problem of emptiness is decidable in polynomial time for PA.

pr.: states marking, construction of $M_i \subseteq Q$.

- initially, $M_0 = \emptyset$.
- at each step, if for $q \in Q \setminus M_i$, $\bigwedge_{p \in Q \setminus M_i} x_p = 0 \wedge \bigvee_{a \in \Sigma} \phi_{a,q}$ is satisfiable, then $M_{i+1} := M_i \cup \{q\}$.

PA: decision of inclusion, equivalence

Proposition : $\subseteq$, $\equiv$

The problems of inclusion and equivalence are decidable for PA.

pr.: $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff $L(\mathcal{A}_1) \cap (\mathcal{U}_2(\Sigma) \setminus L(\mathcal{A}_2)) = \emptyset$.

Plan

Unranked Unordered Trees

Presburger Arithmetic

Presburger Automata (PA)

Determinism

Boolean Closure

Decision Problems

Regular Languages modulo Associativity and Commutativity

Weak Second Order Monadic Logic PMSO

Correspondence between PA, HA and AC-TA

There is a correspondence between:

- the languages of PA
- the languages of CF-HA whose (CF) languages in transitions are closed under permutation
- the class of closure of regular tree languages modulo AC

Rappel: CF-HA

Definition : CF-HA

CF-HA: tuple $(\Sigma, Q, Q_{\mathsf{f}}, \Delta)$ like a HA whose transitions are of the form $a(L) \to q$ with $a \in \Sigma$, $q \in Q$, and $L \subseteq Q^*$ is CF.

Regular languages modulo AC (AC-TA)

Signature $\Sigma = \Sigma_\emptyset \uplus \{a\}$.
The symbol a is binary and follows the axioms of associativity and commutativity:

$$a(x_1, a(x_2, x_3)) = a(a(x_1, x_2), x_3) \qquad \text{(A)}$$

$$a(x_1, x_2)) = a(x_2, x_1) \qquad \text{(C)}$$

For a TA $\mathcal{B}$ over Σ, we note
$\mathsf{AC}\big(L(\mathcal{B})\big) := \big\{t \in \mathcal{T}(\Sigma) \;\big|\; t =_{\mathsf{AC}} s \in L(\mathcal{B})\big\}$ (language of AC-TA)

Proposition :

- the class of regular tree languages is strictly included in the class of AC-TA languages.
- La class of AC-TA languages is closed under Boolean operations.

Correspondence between PA, HA and AC-TA

There is a correspondence between:

- the languages of PA
- the languages of CF-HA whose (CF) languages in transitions are closed under permutation
- the class of closure of regular tree languages modulo AC

Correspondences $\mathcal{T}(\Sigma) \leftrightarrow \mathcal{U}_2(\Sigma)$

We assume $\Sigma_{\mathsf{AC}} = \{a\}$.

$$\begin{array}{lrcl} \mathit{flat}: & \mathcal{T}(\Sigma) & \to & \mathcal{U}_2(\Sigma) \\ \mathit{hflat}: & \mathcal{T}(\Sigma)^* & \to & \mathcal{H}(\Sigma) \\ \mathit{flat}^{-1}: & \mathcal{U}_2(\Sigma) & \to & \mathcal{T}(\Sigma) \end{array}$$

Definitions ($g \in \Sigma_n \setminus \{a\}$):

$$\begin{array}{rcl} \mathit{flat}\big(g(t_1,\ldots,t_n)\big) & = & g(\mathit{flat}(t_1)\ldots\mathit{flat}(t_n)) \\ \mathit{flat}\big(a(t_1,t_2)\big) & = & a(\mathit{hflat}(t_1\,t_2)) \\ \mathit{hflat}\big(g(s_1,\ldots,s_n)\,t_2\ldots t_m\big) & = & \mathit{flat}\big(g(s_1,\ldots,s_n)\big)\,\mathit{hflat}(t_2\ldots t_m) \\ \mathit{hflat}\big(a(s_1,s_2)\,t_2\ldots t_m\big) & = & \mathit{hflat}(s_1 s_2\,t_2\ldots t_m) \\ \mathit{flat}^{-1}\big(g(t_1\ldots t_n)\big) & = & g(\mathit{flat}^{-1}(t_1),\ldots,\mathit{flat}^{-1}(t_n)) \\ \mathit{flat}^{-1}\big(a(t_1\ldots t_m)\big) & = & a(t_1,a(t_2,\ldots,a(t_{m-1},t_m))) \\ & & (m \geq 2) \end{array}$$

Plan

Unranked Unordered Trees

Presburger Arithmetic

Presburger Automata (PA)

Determinism

Boolean Closure

Decision Problems

Regular Languages modulo Associativity and Commutativity

Weak Second Order Monadic Logic PMSO

Weak Second Order Monadic Logic of Presburger

Syntax of formulae of PMSO.

- first order variables $x \ldots$
- second order variables $X \ldots$
-

$$
\begin{array}{rcl}
\text{form} & ::= & x = y \mid x \downarrow y \mid a(x) \mid x \in X \mid x/\text{pres} \quad (a \in \Sigma) \\
& \mid & \text{form} \wedge \text{form} \mid \text{form} \vee \text{form} \mid \neg\text{form} \\
& \mid & \exists x\, \text{form} \mid \exists X\, \text{form} \mid \forall x\, \text{form} \mid \forall X\, \text{form} \\
\text{pres} & ::= & \text{term} = \text{term} \\
& \mid & \neg\text{pres} \mid \text{pres} \vee \text{pres} \mid \text{pres} \wedge \text{pres} \\
& \mid & \forall z\, \text{pres} \mid \exists z\, \text{pres} \\
\text{term} & ::= & z \quad \text{(first order integer variable)} \\
& \mid & n \quad \text{(natural number)} \\
& \mid & [X] \quad (X \text{ second order var.}) \\
& \mid & \text{term} + \text{term}
\end{array}
$$

formulas pres such that the variables z are bounded.

rem. no atoms $x \to y$ as for ordered trees.

Weak Second Order Monadic Logic of Presburger

Semantics of PMSO.

- interpretation domain : set $\|t\|$ of nodes of a tree $t \in \mathcal{U}_2(\Sigma)$,
- σ: first order variables $\to \|t\|$,
- ρ: second order variable $\to 2^{\|t\|}$,
- $t, \sigma, \rho \models x = y$ iff $\sigma(x)$ is $\sigma(y)$,
- $t, \sigma, \rho \models x \downarrow y$ iff $\sigma(x)$ is the father of $\sigma(y)$ in t,
- $t, \sigma, \rho \models a(x)$ iff $\sigma(x)$ labeled by a in t,
- $t, \sigma, \rho \models x \in X$ iff $\sigma(x) \in \rho(X)$,
- $t, \sigma, \rho \models x/\phi$ iff $(n_1, \ldots, n_p) \models \phi$ where n_i is the number of sons (in t) of $\sigma(x)$ in $\rho(X_i)$ (with $dom(\rho) = \{X_1, \ldots, X_p\}$),
- $t, \sigma, \rho \models \psi_1 \vee \psi_2$ iff $t, \sigma, \rho \models \psi_1$ or $t, \sigma, \rho \models \psi_2$,
- $t, \sigma, \rho \models \psi_1 \wedge \psi_2$ iff $t, \sigma, \rho \models \psi_1$ and $t, \sigma, \rho \models \psi_2$,
- $t, \sigma, \rho \models \neg\psi$ iff $u, \sigma \not\models \psi$,
- $t, \sigma, \rho \models \exists x\, \psi$ iff there exists $p \in \|t\|$ s.t. $t, \sigma \cup \{p \to x\}, \rho \models \psi$,
- $t, \sigma, \rho \models \exists X \psi$ iff the exists $P \subseteq \|t\|$ s.t. $t, \sigma, \rho \cup \{P \to X\} \models \psi$,

PMSO: examples

- root:

  $x = \varepsilon \equiv \neg\exists y\, y \downarrow x$

- leaf:

  $\mathsf{leaf}(x) \equiv \neg\exists y\, x \downarrow y$

- $x \downarrow y \equiv \exists Y\, Y = \{y\} \wedge x/_{[Y]=1}$

- prefix ordering $=$ transitive cloture of $\downarrow$:

  $x \downarrow^* y \equiv \forall X \left(x \in X \wedge \forall z \forall z' \left(z \in X \wedge z \downarrow z' \Rightarrow z' \in X\right)\right) \Rightarrow y \in X$

PMSO languages

Definition : language

The language defined by the closed PMSO formula ψ over Σ is the set of terms $t \in \mathcal{U}_2(\Sigma)$ s.t. $t \models \psi$.

language of PMSO: example 1

The set of trees of the form $f(a, \ldots, a, b, \ldots, b)$ with the same number of a and b.

$$\begin{aligned} \exists X_a \, \exists X_b \, f(\varepsilon) \quad & \wedge \forall y \, (y \in X_a \Leftrightarrow a(y)) \wedge (y \in X_b \Leftrightarrow b(y)) \\ & \wedge \forall y \, \varepsilon \downarrow y \Rightarrow (\mathsf{leaf}(y) \wedge (y \in X_a \vee y \in X_b)) \\ & \wedge \varepsilon /_{[X_a]=[X_b]} \end{aligned}$$

PMSO: PA example 2

$\Sigma = \{a, b\}$.
The set of trees of $\mathcal{U}_2(\Sigma)$ s.t. every internal node is labeled by a and every node labeled by a has at least as much sons without b than sons containing b.

$$\begin{array}{lll} \exists X_a\, \exists X_b\, \forall x & & (b(x) \Rightarrow \mathsf{leaf}(x)) \\ & \wedge & (a(x) \wedge x/_{[X_a] \geq [X_b] > 0} \Rightarrow x \in X_b) \\ & \wedge & (a(x) \wedge x/_{[X_a] \geq [X_b] = 0} \Rightarrow x \in X_a) \\ & \wedge & (b(x) \wedge x/_{[X_a] = [X_b] = 0} \Rightarrow x \in X_b) \end{array}$$

$Q = Q_{\mathsf{f}} = \{q_a, q_b\}$.
The state q_b accepts the trees containing a b and q_a accepts the others.

$$\Delta = \left\{ \begin{array}{llll} a(x_{q_a} \geq x_{q_b} = 0) & \rightarrow & q_a & a(x_{q_a} \geq x_{q_b} > 0) \rightarrow q_b \\ & & & b(x_{q_a} = x_{q_b} = 0) \rightarrow q_b \end{array} \right\}$$

PMSO: examples of queries

Base of clients of an online music store, stored in an unordered unranked tree.

A client is a subtree:

- root labeled by `client`
- informations in the sons (purchase, labeled at root by the kind).

Query for clients x who have purchased more jazz than blues:

$$\begin{aligned} query_1(x) \equiv \quad & (\exists X_{\texttt{jazz}}\, \forall y\, y \in X_{\texttt{jazz}} \Leftrightarrow \texttt{jazz}(y)) \\ \wedge \quad & (\exists X_{\texttt{blues}}\, \forall y\, y \in X_{\texttt{blues}} \Leftrightarrow \texttt{blues}(y)) \\ \wedge \quad & \texttt{client}(x) \wedge x/_{[X_{\texttt{jazz}}]>[X_{\texttt{blues}}]} \end{aligned}$$

Query for clients x who have purchased more jazz than anything else:

$$\begin{aligned} query_1(x) \equiv \quad & (\exists X_{\texttt{jazz}}\, \forall y\, y \in X_{\texttt{jazz}} \Leftrightarrow \texttt{jazz}(y)) \\ \wedge \quad & (\exists X_{\texttt{other}}\, \forall y\, y \in X_{\texttt{other}} \Leftrightarrow \neg\texttt{jazz}(y)) \\ \wedge \quad & \texttt{client}(x) \wedge x/_{[X_{\texttt{jazz}}]>[X_{\texttt{other}}]} \end{aligned}$$

PMSO and PA

Theorem

$L \subseteq \mathcal{U}(\Sigma)$ is definable in PMSO iff L is a PA language.

Part VI

Presburger Constraints and Unranked Ordered Trees

Presburger Hedge Automata (PHA)

Presburger Hedge Automata (PHA)

Definition : Presburger Hedge Automata

A *Presburger Hedge Automaton* (PHA) over an alphabet Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ where $Q = \{q_1, \ldots, q_p\}$ is a finite set of *states*, $Q^{\mathsf{f}} \subseteq Q$ is the subset of final states and Δ is a set of transition rules of the form: $a\big(\bigvee_i (L_i \wedge \phi_i)\big) \to q$ with $a \in \Sigma$, $q \in Q$, $L_i \subseteq Q^*$ regular language and $\phi_i = \phi(x_1, \ldots, x_p)$ is a Presburger formula with a free variable for each state.

for all $w \in Q^*$ we define $w \models L_i \wedge \phi_i$ if $w \in L_i$ and $\#(w) \models \phi_i$.

PHA: languages

The language $L(\mathcal{A}, q)$ of $\mathcal{A}$ in state $q \in Q$ is the smallest set of ordered unranked trees $a(t_1, \ldots, t_n) \in \mathcal{U}(\Sigma)$ s.t.

- there exists $i_1, \ldots, i_n \leq p$ such that for all $j \leq n$, $t_j \in L(\mathcal{A}, q_{i_j})$,
- there exists a transition $a\big(\bigvee_i (L_i \wedge \phi_i)\big) \to q \in \Delta$ such that $q_{i_1} \ldots q_{i_n} \models \bigvee_i (L_i \wedge \phi_i)$, i.e. there exists i s.t.
  - $q_{i_1} \ldots q_{i_n} \in L_i$,
  - $\#(q_{i_1} \ldots q_{i_n}) \models \phi_i(x_1, \ldots, x_p)$.

The language of $\mathcal{A}$ is $L(\mathcal{A}) = \bigcup_{q \in Q_\mathsf{f}} L(\mathcal{A}, q)$.

PHA: proprerties

Proposition :

- The class of PHA languages is closed under union and intersection,
- The class of PHA languages is not closed under complementation.

- $\cup$, $\cap$: product
- csq undecidability of the problem of universality.

Proposition :

DPHA $\not\equiv$ NPHA.

PHA: decision problems

Lemma :

Given a finite set Q, $L \subseteq Q^*$, regular, and $\phi = \phi(x_1, \ldots, x_p)$ a formula of Presburger ($p = |Q|$), it is decidable whether there exists $w \in L$ such that $\#(w) \models \phi$.

Proposition : $\in$

The membership is decidable in PTIME for PHA.

Proposition : $\emptyset$

The emptiness is decidable for PHA.

Proposition : $\forall$

Universality is undecidable for PHA.

PHA: logique

Theorem :

A set of trees of $\mathcal{U}(\Sigma)$ is recognizable by a PHA iff it is defined by a PMSO formula of the form $\exists X_1 \dots \exists X_k \, \phi$ where ϕ is first order.

Corollary :

- EPMSO (existential fragment) is decidable in $\mathcal{U}(\Sigma)$.
- PMSO is undecidable over $\mathcal{U}(\Sigma)$.

Part VII

Mixed Labeled Trees (Unordered and Ordered)

Mixed Trees

- $\Sigma = \Sigma_A \cup \Sigma_{AC}$.

$$\begin{array}{rcll} \text{tree} & := & a(\text{hedge}) \mid c(\text{multiset}) & (a \in \Sigma_A, c \in \Sigma_{AC}) \\ \text{hedge} & := & \text{tree}, \ldots, \text{tree} & \\ \text{multiset} & := & \{\text{tree}, \ldots, \text{tree}\} & \end{array}$$

- $c\big(\{t_1, \ldots, t_n\}\big)$ is denoted $c(t_1, \ldots, t_n)$.
- The set of mixed unranked trees over Σ is denoted $\mathcal{M}(\Sigma)$.

Presburger m-Tree Automata (PMA)

Definition : Presburger m-Tree Automata

A *Presburger m-Tree Automaton* (PMA) over an alphabet $\Sigma = \Sigma_A \cup \Sigma_{AC}$ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$ where $Q = \{q_1, \ldots, q_p\}$ is a finite set of *states*, $Q^{\mathsf{f}} \subseteq Q$ is the subset of final states and Δ is a set of transition rules of the form:

- $a\big(L\big) \to q$ with $a \in \Sigma_A$, $q \in Q$, $L \subseteq Q^*$ is a regular language or
- $c\big(\phi\big) \to q$ with $c \in \Sigma$, $q \in Q$, $\phi = \phi(x_1, \ldots, x_p)$ is a Presburger formula with one free variable for each state.

PMA: languages

The language $L(\mathcal{A}, q)$ of the PMA $\mathcal{A}$ in state $q \in Q$, is the smallest set of mixed trees

- $a(t_1, \ldots, t_n)$, $a \in \Sigma_A$, such that
  - there exists $i_1, \ldots, i_n \leq p$ with $t_j \in L(\mathcal{A}, q_{i_j})$ for all $j \leq n$,
  - there exists a transition $a(L) \to q \in \Delta$ such that $q_{i_1} \ldots q_{i_n} \in L$,
- or $c(t_1, \ldots, t_n)$, $c \in \Sigma_{AC}$, such that
  - there exists $i_1, \ldots, i_n \leq p$ with $t_j \in L(\mathcal{A}, q_{i_j})$ for all $j \leq n$,
  - there exists a transition $c(\phi) \to q \in \Delta$ such that $\#(q_{i_1} \ldots q_{i_n}) \models \phi(x_1, \ldots, x_p)$.

The language of $\mathcal{A}$ is $L(\mathcal{A}) = \bigcup_{q \in Q_\mathsf{f}} L(\mathcal{A}, q)$.

PMA: properties

Proposition :

The class of PMA languages is closed under all Boolean operations.

Proposition :

DPMA $\equiv$ NPMA.

PMA: decision problems

Proposition : $\in$

Membership is decidable for PMA.

Proposition : $\emptyset$

Emptiness is decidable for PMA.

Consequences of the analogous results for PHA ($\mathrm{PMA} \subseteq \mathrm{PHA}$).

PMA: logic

Theorem :

The class of languages of $\mathcal{M}(\Sigma)$ definable by PMSO formulae is the class of PMA languages.

Corollary

PMSO over $\mathcal{M}(\Sigma)$ is decidable.

Part VIII

Alternating Tree Automata

Tree Automata Techniques and Applications

Hubert Comon Max Dauchet Rémi Gilleron
Florent Jacquemard Denis Lugiez Christof Löding
Sophie Tison Marc Tommasi

Chandra Kozen Stockmeyer. Alternation. J. ACM vol. 28, 1981

Top-Down Tree Automata (rappel)

Context: Finite ranked terms of a signature Σ (every symbol has a fixed arity).

Definition : Top-Down Tree Automata

A top-down tree automaton over a signature Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\text{init}}, \Delta)$ where Q is a finite set of *states*, $Q^{\text{init}} \subseteq Q$ is the subset of initial states and Δ is a set of transition rules of the form: $q \to f(q_1, \ldots, q_n)$ with $f \in \Sigma_n$ $(n \geq 0)$ and $q_1, \ldots, q_n, q \in Q$.

A ground term $t \in \mathcal{T}(\Sigma)$ is accepted by $\mathcal{A}$ in the state q iff $q \xrightarrow[\Delta]{*} t$.

The language of $\mathcal{A}$ starting from the state q is $L(\mathcal{A}, q) := \left\{ t \in \mathcal{T}(\Sigma) \;\middle|\; q \xrightarrow[\Delta]{*} t \right\}$.

The language of $\mathcal{A}$ is $L(\mathcal{A}) := \bigcup_{q \in Q^{\text{init}}} L(Q, q)$.

Top-Down Tree Automata: example

Terms in $\mathcal{T}\big(\{f{:}2, g{:}2, a{:}0\}\big)$ containing exactly one g:
$Q = \{q_0, q_1\}$, $Q^{\mathsf{init}} = \{q_1\}$,

$$\begin{array}{lcllcl} q_1 & \to & f(q_0, q_1) & q_1 & \to & f(q_1, q_0) \\ q_1 & \to & g(q_0, q_0) & & & \\ q_0 & \to & f(q_0, q_0) & q_0 & \to & a \end{array}$$

Terms in $\mathcal{T}\big(\{f{:}2, g{:}2, a{:}0\}\big)$ containing at least one g:
$Q = \{q_0, q_1\}$, $Q^{\mathsf{init}} = \{q_1\}$,

$$\begin{array}{lcllcl} q_1 & \to & f(q_0, q_1) & q_1 & \to & f(q_1, q_0) \\ q_1 & \to & g(q_0, q_0) & q_0 & \to & g(q_0, q_0) \\ q_0 & \to & f(q_0, q_0) & q_0 & \to & a \end{array}$$

Top-Down Tree Automata (rappels, ctnd)

expressiveness:

$$\text{D top-down} \subsetneq \text{ND top-down} = \text{ND bottom-up} = \text{D bottom-up}$$

Non determinism and alternation

Non determinism:

$$\begin{array}{lcl} q & \rightarrow & f(q_{1,1}, \ldots, q_{1,n}), \\ & \vdots & \\ q & \rightarrow & f(q_{k,1}, \ldots, q_{k,n}) \end{array}$$

can be summarized into one single disjunctive transition:

$$q, f \rightarrow (q_{1,1}, \ldots, q_{1,n}) \vee \ldots \vee (q_{k,1}, \ldots, q_{k,n})$$

or also

$$q, f \rightarrow \bigvee_{i=1}^{k} \bigwedge_{j=1}^{n} \langle q_{i,j}, j \rangle$$

Alternating Ranked Tree Automata

Definition : Alternating Ranked Tree Automata

An alternating tree automaton over the ranked signature Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\text{init}}, \delta)$ where Q is a finite set of states, $Q^{\text{init}} \subseteq Q$ is the subset of initial states and δ is a function from $Q \times \Sigma$ into $\mathcal{B}^+(Q \times \mathbb{N})$ such that $\delta(q, f) \in \mathcal{B}^+\big(Q \times \{1..arity(f)\}\big)$.

$\mathcal{B}^+ =$ positive Boolean combinaisons i.e. built with $\wedge$ and $\vee$ only

Alternating Tree Automata: example

Terms in $\mathcal{T}\big(\{f{:}2, g{:}2, a{:}0\}\big)$ containing exactly one g:

$$
\begin{array}{lcllcl}
q_1 & \to & f(q_0, q_1) & q_1 & \to & f(q_1, q_0) \\
q_1 & \to & g(q_0, q_0) & & & \\
q_0 & \to & f(q_0, q_0) & q_0 & \to & a
\end{array}
$$

$$\equiv$$

$$
\begin{array}{lcl}
q_1, f & \to & \big(\langle q_0, 1\rangle \wedge \langle q_1, 2\rangle\big) \vee \big(\langle q_1, 1\rangle \wedge \langle q_0, 2\rangle\big) \\
q_1, g & \to & \langle q_0, 1\rangle \wedge \langle q_0, 2\rangle \\
q_0, f & \to & \langle q_0, 1\rangle \wedge \langle q_0, 2\rangle \\
q_0, g & \to & \mathit{false} \\
q_0, a & \to & \mathit{true} \\
q_1, a & \to & \mathit{false}
\end{array}
$$

Alternating Tree Automata: runs

A run of an alternating tree automaton $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{init}}, \delta)$ on a term $t \in \mathcal{T}(\Sigma)$ is a tree r labelled by $Q \times \mathbb{N}^*$ such that:

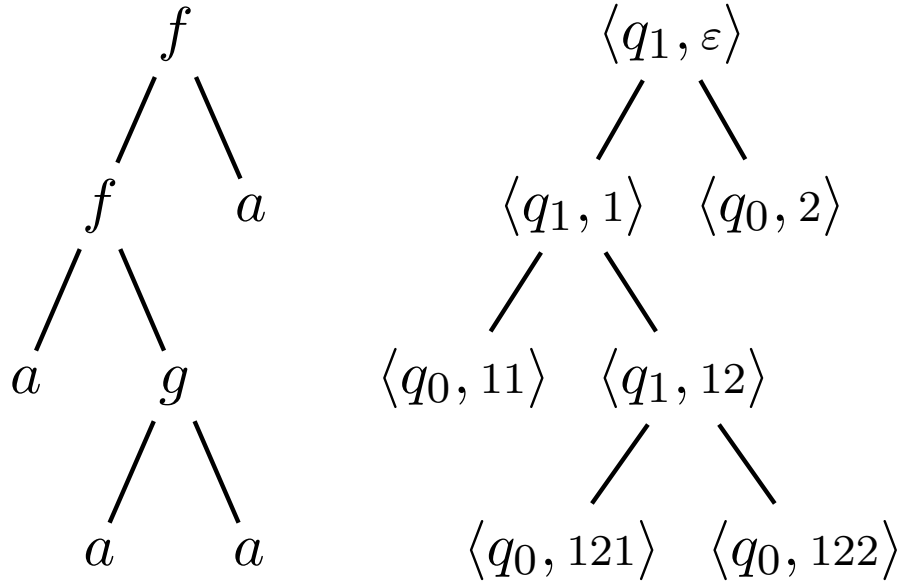
- $r(\varepsilon) = \langle q, \varepsilon \rangle$ for $q \in Q$,
- if $r(\pi) = \langle q, p \rangle$ and $t(p) = f$, then there exists $S = \{\langle q_1, i_1 \rangle, \ldots, \langle q_k, i_k \rangle\} \subseteq Q \times \{1..arity(f)\}$ such that $S \models \delta(q, f)$, and $r(\pi \cdot j) = \langle q_j, p \cdot i_j \rangle$ for all $j \in \{1..k\}$.

$L(\mathcal{A}, q) = \{t \mid \exists \text{ run } r \text{ of } \mathcal{A} \text{ on } t \text{ with } r(\varepsilon) = \langle q, \varepsilon \rangle\}$.

$L(\mathcal{A}) := \bigcup_{q \in Q^{\mathsf{init}}} L(\mathcal{A}, q)$ ($\exists$ an accepting run).

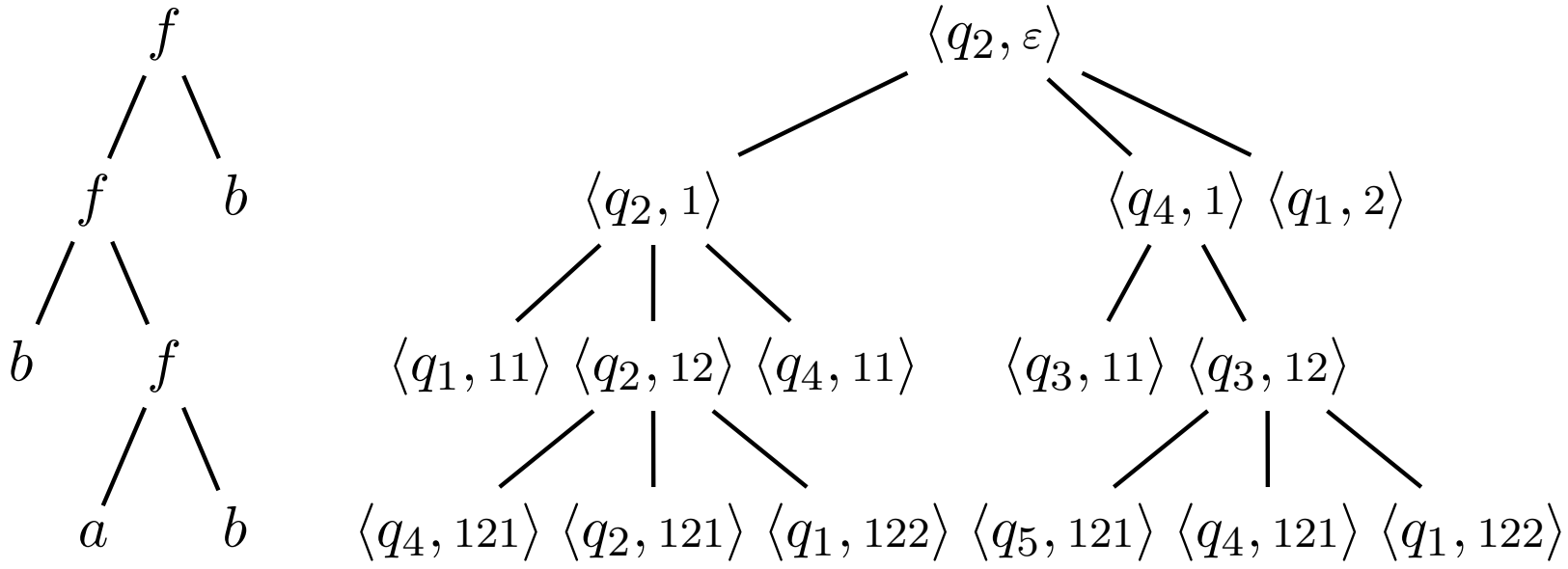
Run of Alternating Tree Automata: example 1

$$
\begin{array}{lcl}
q_1, f & \rightarrow & \big(\langle q_0, 1\rangle \wedge \langle q_1, 2\rangle\big) \vee \big(\langle q_1, 1\rangle \wedge \langle q_0, 2\rangle\big) \\
q_0, f & \rightarrow & \langle q_0, 1\rangle \wedge \langle q_0, 2\rangle \\
q_1, g & \rightarrow & \langle q_0, 1\rangle \wedge \langle q_0, 2\rangle \\
q_0, g & \rightarrow & \mathit{false} \\
q_0, a & \rightarrow & \mathit{true} \\
q_1, a & \rightarrow & \mathit{false}
\end{array}
$$



Run of Alternating Tree Automata: example 2

q_2,f	$\rightarrow$	$\Big[(\langle q_1,1\rangle \wedge \langle q_2,2\rangle) \vee (\langle q_1,2\rangle \wedge \langle q_2,1\rangle)\Big] \wedge \langle q_4,1\rangle$	$q_2,a \rightarrow true$	$q_2,b \rightarrow false$
q_1,f	$\rightarrow$	$(\langle q_2,1\rangle \wedge \langle q_2,2\rangle) \vee (\langle q_1,2\rangle \wedge \langle q_1,1\rangle)$	$q_1,a \rightarrow false$	$q_1,b \rightarrow true$
q_4,f	$\rightarrow$	$(\langle q_3,1\rangle \wedge \langle q_3,2\rangle) \vee (\langle q_4,1\rangle \wedge \langle q_4,2\rangle)$	$q_4,a \rightarrow true$	$q_4,b \rightarrow true$
q_3,f	$\rightarrow$	$\Big[(\langle q_3,1\rangle \wedge \langle q_2,2\rangle) \vee (\langle q_4,1\rangle \wedge \langle q_1,2\rangle)\Big] \wedge \langle q_5,1\rangle$	$q_3,a \rightarrow false$	$q_3,b \rightarrow true$
q_5,f	$\rightarrow$	$false$	$q_5,a \rightarrow true$	$q_5,b \rightarrow false$



Boolean Closure

Proposition :

Given two alternating tree automata $\mathcal{A}_1$ and $\mathcal{A}_2$ over Σ, one can construct in linear time some alternating tree automata recognizing $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, and $\overline{L(\mathcal{A}_1)}$.

Expressiveness

Proposition :

Given an alternating tree automaton $\mathcal{A}$ over Σ, one can construct in exponential time a deterministic bottom-up tree automaton $\mathcal{A}'$ recognizing the same language.

The exponential blowup cannot be avoided.

Decision Results

Proposition :

The problem of membership is decidable in polynomial time for alternating tree automata.

Proposition :

The problems of emptiness and universality are both EXPTIME-complete for alternating tree automata.

Alternating Unranked Tree Automata

Context: Finite unranked ordered terms of an alphabet Σ.

Definition : Alternating unranked tree automata

An alternating tree automaton over the unranked alphabet Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q^{\text{init}}, \delta)$ where Q is a finite set of states, $Q^{\text{init}} \subseteq Q$ is the subset of initial states and δ is a function from Q into

- $true$, $false$
- a, $\overline{a}$, with $a \in \Sigma$
- $q \wedge q'$, $q \vee q'$, with $q, q' \in Q$
- $\langle q, \downarrow_1 \rangle$, $\langle q, \rightarrow \rangle$, with $q \in Q$
- $is\ leaf$, $\overline{is\ leaf}$, $is\ last$, $\overline{is\ last}$

Alternating Unranked Tree Automata: runs

A run of an alternating unranked tree automaton $\mathcal{A} = (\Sigma, Q, Q^{\text{init}}, \delta)$ on a term $t \in \mathcal{U}(\Sigma)$ is a tree r labelled by $Q \times \mathbb{N}^*$ such that:

- $r(\varepsilon) = \langle q, \varepsilon \rangle$, if $q \in Q^{\text{init}}$, the run is called successful
- if $r(\pi) = \langle q, p \rangle$ then π is a leaf in r and
  - $\delta(q) = true$
  - $\delta(q) = a$ and $t(p) = a$
  - $\delta(q) = \overline{a}$ and $t(p) \neq a$
  - if $\delta(q) = is\ leaf$ then p is a leaf in t
  - if $\delta(q) = \overline{is\ leaf}$ then p is not a leaf in t
  - if $\delta(q) = is\ last$ then $p = \varepsilon$ or $p = p' \cdot i$ and $p' \cdot (i+1) \notin \mathcal{P}os(t)$ (p is a last sibling in t)
  - if $\delta(q) = \overline{is\ last}$ $p = p' \cdot i$ and $p' \cdot (i+1) \in \mathcal{P}os(t)$

Alternating Unranked Tree Automata: runs (contd)

- if $r(\pi) = \langle q, p \rangle$ then π is not a leaf in r and
  - if $\delta(q) = q_1 \wedge q_2$, then π has 2 children in r, with $r(\pi \cdot 1) = \langle q_1, p \rangle$ and $r(\pi \cdot 2) = \langle q_2, p \rangle$,
  - if $\delta(q) = q_1 \vee q_2$, then π has 1 child in r, with $r(\pi \cdot 1) = \langle q_1, p \rangle$ or $r(\pi \cdot 1) = \langle q_2, p \rangle$,
  - if $\delta(q) = \langle q_1, \downarrow_1 \rangle$, then p is not a leaf in t and π has 1 child in r, with $r(\pi \cdot 1) = \langle q_1, p \cdot 1 \rangle$
  - if $\delta(q) = \langle q_1, \rightarrow \rangle$, then $p = p' \cdot i$, $p' \cdot (i+1) \in \mathcal{P}os(t)$ and π has 1 child in r, with $r(\pi \cdot 1) = \langle q_1, p' \cdot (i+1) \rangle$

Part IX

Tree Automata defined as Sets of Horn Clauses

Definition of tree automata as set of first order (universal) clauses.
Languages = Herbrand models.

- \+ uniforme formalism for the definition of several classes of automata (alternating, 2-ways, with constraints...)
- \+ enables the use of techniques and tools from automatic déduction in order to solve the classical decision problems
- – complexity
- – model is not operational
  (not easy to construct a witness, a run...)

Clauses: syntax

- terms in $\mathcal{T}(\Sigma, \mathcal{X})$ over signature Σ (Σ_n: symbol of arity n)
- finite set $\mathcal{P}$ of predicate symbols P, Q,... (notation $\mathcal{P}_n$)
  basically we will only considerer predicates of arity 0 or 1.
- literals positive: $P(t)$, noté $+P(t)$
  négative: $\neg P(t)$, noté $-P(t)$
- clause: disjunction of literals $\pm P_1(t_1) \vee \ldots \vee \pm P_k(t_k)$
  empty clause ($k = 0$), denoted $\bot$.
- Horn clause: at most one positive literal
  $-P_1(t_1) \vee \ldots \vee -P_k(t_k) \vee +P(t)$, denoted
  $P_1(t_1), \ldots, P_k(t_k) \Rightarrow P(t)$.
- goal = negative clause
  $-P_1(t_1) \vee \ldots \vee -P_k(t_k)$, denoted $P_1(t_1), \ldots, P_k(t_k) \Rightarrow \bot$.

Clauses: semantics

- interpretation structure $\mathcal{M}$ of domain D:
  $\mathcal{M} = \langle D, Q^{\mathcal{M}} \subseteq D^n \mid Q \in \mathcal{P}_n, f^{\mathcal{M}} : D^n \to D \mid f \in \Sigma_n \rangle$
  for $Q \in \mathcal{P}_0$, $Q^{\mathcal{M}} \in \{true, false\}$.
- interpretation $\rho : \mathcal{X} \to D$
- valuation of terms $[\![\_]\!]_{\mathcal{M},\rho} : \mathcal{T}(\Sigma, \mathcal{X}) \to D$
  - $[\![x]\!]_{\mathcal{M},\rho} := \rho(x)$,
  - $[\![f(t_1, \ldots, t_n)]\!]_{\mathcal{M},\rho} := f^{\mathcal{M}}([\![t_1]\!]_{\mathcal{M},\rho}, \ldots, [\![t_1]\!]_{\mathcal{M},\rho})$
- $\mathcal{M}, \rho \models C$ iff
  - there exists $+P(t) \in C$ such that $[\![t]\!]_{\mathcal{M},\rho} \in P^{\mathcal{M}}$
    or
  - there exists $-P(t) \in C$ such that $[\![t]\!]_{\mathcal{M},\rho} \notin P^{\mathcal{M}}$
- $\mathcal{M}$ is a model of a set of clauses S (denoted $\mathcal{M} \models S$) iff for all $C \in S$, for all interpretation ρ, $\mathcal{M}, \rho \models C$.
- S is called satisfiable iff it admits a model.

Herbrand Models

- a Herbrand structure $\mathcal{H}$ has domain $\mathcal{T}(\Sigma)$ and fonctions $f^{\mathcal{H}}(t_1, \ldots, t_n) := f(t_1, \ldots, t_n)$ (ground term with the symbol f at the root).
- $\mathcal{H}$ is completely defined by the set of ground atoms $P(t)$ such that $\mathcal{H} \models P(t)$.

Theorem :

A set of clauses S is satisfiable iff it admits a Herbrand model.

Smallest Herbrand Models

Theorem :

Every satisfiable set S of Horn clauses admits a smallest (wrt inclusion) Herbrand model $\mathcal{H}_S$.

A set of Horn clauses S defines the following operator T_S over sets of ground atoms

$$\begin{array}{rcl} T_S(L) & = & \left\{ P(t\sigma) \;\middle|\; \begin{array}{l} t\sigma \text{ ground}, P_1(t_1), \ldots, P_n(t_n) \Rightarrow P(t) \in S, \\ P_1(t_1\sigma), \ldots, P_n(t_n\sigma) \in L \end{array} \right\} \\ & \cup & \{\bot\} \text{ if } \begin{array}{l} P_1(t_1), \ldots, P_n(t_n) \Rightarrow \bot \in S, \\ P_1(t_1\sigma), \ldots, P_n(t_n\sigma) \in L \end{array} \end{array}$$

The smallest fixpoint of T_S is $\bigcup_{n \geq 1} T_S^n(\emptyset)$,

- if it contains $\bot$, then S is not satisfiable,
- otherwise, it is the smallest Herbrand model of S.

Languages and automata

The language of a satisfiable set S of Horn clauses for a predicate Q is:

$$L(S, Q) = \{t \mid Q(t) \in \mathcal{H}_S\}$$

Let $\mathcal{A} = (\Sigma, \{q_1, \ldots, q_k\}, F, \Delta)$ be a bottom-up tree automaton.
Let $\mathcal{P} = \{Q_1, \ldots, Q_k\}$ be a set of unary predicates.
We associate to $\mathcal{A}$ the (satisfiable) set of Horn clauses

$$S_{\mathcal{A}} := \left\{ \begin{array}{r} Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \\ \mid f(q_1, \ldots, q_n) \to q \in \Delta \end{array} \right\}$$

Lemma :

For all state q, $L(\mathcal{A}, q) = L(S_{\mathcal{A}}, Q)$.

Clauses/classes of automata

clauses of standard automate ($x_1, \ldots, x_n$ pairwise distinct)

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg)}$$

ε-transitions

$$Q_1(x) \Rightarrow Q(x) \qquad (\varepsilon)$$

alternating clauses

$$Q_1(x), \ldots, Q_n(x) \Rightarrow Q(x) \qquad \text{(alt)}$$

2-ways (bidirectional) clauses ($x_1, \ldots, x_n$ pairwise distinct)

$$Q\big(f(x_1, \ldots, x_n)\big) \Rightarrow Q_i(x_i) \qquad \text{(bidi)}$$

Decision problems, satisfiability

Let S be a satisfiable set of Horn clauses and let Q be a predicate.

- membership: the ground term $t \in L(S, Q)$ iff $S \cup \{Q(t) \Rightarrow \bot\}$ is unsatisfiable
- emptiness: $L(S, Q) \neq \emptyset$ iff $S \cup \{Q(x) \Rightarrow \bot\}$ is unsatisfiable.
- membership of instance: there exists σ such that $t\sigma \in L(S, Q)$ iff $S \cup \{Q(t) \Rightarrow \bot\}$ is unsatisfiable.
- emptiness of intersection: $L(S, Q_1) \cap \ldots \cap L(S, Q_p) \neq \emptyset$ iff $S \cup \{Q_1(x), \ldots, Q_p(x) \Rightarrow \bot\}$ is unsatisfiable.

$\Rightarrow$ we are interested in automated deduction techniques for deciding the satisfiability when S represents an automaton.

Resolution

$$\frac{C \vee +Q(s) \quad D \vee -Q(t)}{C\sigma \vee D\sigma}$$

where σ is the most general unifier (*mgu*) of s and t.

Horn clauses:

$$\frac{P_1(s_1), \ldots, P_m(s_m) \Rightarrow Q_1(s) \quad Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t)}{P_1(s_1\sigma), \ldots, P_m(s_m\sigma), Q_2(t_2\sigma), \ldots, Q_n(t_n\sigma) \Rightarrow Q(t\sigma)}$$

where σ is the *mgu* of s and t_1.

Theorem : correction, completeness

A set S of Horn clauses is unsatisfiable iff one can derive $\perp$ by resolution starting from S.

Terminaison of resolution

The application of the resolution rule to automata clauses (reg) does not terminate.

$$\frac{P_1(x_1), \ldots, P_m(x_m) \Rightarrow Q_1(g(\overline{x})) \quad Q_1(y_1), \ldots, Q_n(y_n) \Rightarrow Q(f(\overline{y}))}{P_1(x_1), \ldots, P_m(x_m), Q_2(y_2), \ldots, Q_n(y_n) \Rightarrow Q\big(f(g(\overline{x}), y_2, \ldots, y_n)\big)}$$

Complete strategies for resolution

$$\frac{\begin{array}{cc} C & D \\ P_1(s_1), \ldots, P_m(s_m) \Rightarrow Q_1(s) & Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t) \end{array}}{P_1(s_1\sigma), \ldots, P_m(s_m\sigma), Q_2(t_2\sigma), \ldots, Q_n(t_n\sigma) \Rightarrow Q(t\sigma)}$$

ordered resolution for $\succ$:

- $Q_1(s)$ maximal for $\succ$ in C,
- $Q_1(t_1)$ maximal for $\succ$ in D.

ordered resolution with selection :
selection function : clause $\mapsto$ subset of negative literals.

- no literal is selected in C,
- $Q_1(s)$ is maximal for $\succ$ in C,
- $Q_1(t_1)$ is selected in D or
- no literal is selected in D and $Q_1(t)$ is maximal in D.

Completeness of ordered resolution

Theorem :

Ordered resolution with selection is complete for Horn clauses.

Starting from any unsatisfiable set S, we shall derive $\bot$.

Choice of an ordered strategy with selection

- ordering $\succ$ s.t. $P(s) \succ Q(t)$ iff $s > t$ for the subterm ordering $>$.
- selection function sel:
  - negative literals $-Q(t)$ where t is not a variable.

Lemma :

Every tree automaton (finite set clauses of type (reg)) is saturated under resolution ordered by $\succ$.

Resolution ordered by $\succ$ and with selection by sel cannot be applied between automata clauses (reg) like in

$$\frac{P_1(x_1), \ldots, P_m(x_m) \Rightarrow Q_1(g(\overline{x})) \quad Q_1(y_1), \ldots, Q_n(y_n) \Rightarrow Q(f(\overline{y}))}{P_1(x_1), \ldots, P_m(x_m), Q_2(y_2), \ldots, Q_n(y_n) \Rightarrow Q\big(f(g(\overline{x}), y_2, \ldots, y_n)\big)}$$

because no literal are selected in the clauses and $Q_1(y_1)$ is not maximal in $\{Q_1(y_1), \ldots, Q_n(y_n), Q(f(\overline{y}))\}$ $(Q(f(\overline{y})) \succ Q_1(y_1))$.

Transformation of alternating automata

Proposition :

Given an alternating tree automaton nt $\mathcal{A}$ over Σ, we can construct in exponential time a deterministic bottom-up tree automaton $\mathcal{A}'$ recognizing the same language.

Construction by application of ordered resolution with selection, following an appropriated strategy.

Transformation of alternating automata

pr.: (proposition alternating automata $\rightarrow$ bottom-up deterministic automata).

- start from a set $\mathcal{A}$ of clauses of the form (reg) and (alt).
- saturate with resolution ordered (by $\succ$) with selection (by sel).
- all the clauses produced belong to a type containing an exponential number of clauses
- hence saturation terminates with a set $\mathcal{A}''$
- the application of resolution to $\mathcal{A}'' \cup \{Q(t) \Rightarrow \bot\}$ (for a ground term t) only involves clauses of the form (reg).
- hence, for all Q, $L(\mathcal{A}'', Q) = L(\mathcal{A}''|_{\mathsf{reg}}, Q)$.

Transformation of bidirectional alternating automata

In order to generalize the previous result (reg + alt $\rightarrow$ reg) to reg + alt + bidi $\rightarrow$ reg, we use the same principle with

- other ordering and selection function for defining the resolution stratégy,
- and a new rule called ε-splitting.

Proposition :

Given a bidirectional alternating tree automaton $\mathcal{A}$ over Σ, we can construct in exponential time a bottom-up deterministic tree automaton $\mathcal{A}'$ recognizing the same language.

ε-splitting

An ε-bloc $B(x)$ is a set of negative literals $-P_1(x) \vee \ldots \vee -P_n(x)$.

$$\frac{B(x), Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t)}{B(x) \Rightarrow q_B \quad q_B, Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t)}$$

- q_B is a nullary predicate associate in a unique way to $B(x)$,
- $x \notin Q_1(t_1), \ldots, Q_n(t_n), Q(t)$.

Theorem :

Ordered resolution with selection and ε-splitting is complete for Horn clauses.

Another choice of ordered strategy with selection

- ordering $\succ$ s.t. $P(s) \succ Q(t)$ iff $s > t$ for the subterm ordering $>$ and $P(s) \succ q_B$ for all P and q_B.
- selection function sel (by order of priority):
  - splitting literals (q_B).
  - negative literals $-Q(t)$ where t is not a variable.

Transformation of bidirectional alternating automata

Proposition :

Given a bidirectional alternating automaton $\mathcal{A}$ over Σ, one can construct in exponential time a bottom-up deterministic tree automaton $\mathcal{A}'$ recognizing the same language.

pr.: same principle as for alternating automata, with the resolution and ε-splitting rules.

Decision of instance membership

Theorem :

The application of resolution ordered by $\succ$ with selection by sel and ε-splitting terminates on the union of a set of clauses (reg) and a goal clause of the form $Q(t) \Rightarrow \bot$.

invariant: the resolution only produces clauses of the following 2 types:

$$P_1(s_1), \ldots, P_m(s_m), q_1, \ldots, q_k \Rightarrow [\, q \,] \qquad \text{(gs)}$$

where $m, k \geq 0$, and $s_1, \ldots, s_m$ are subterms of t.

$$P_1(y_{i_1}), \ldots, P_k(y_{i_k}), \underline{P'_1(f(y_1, \ldots, y_n))}, \ldots, \underline{P'_m(f(\overline{y}))} \Rightarrow [\, q \,] \qquad \text{(gf)}$$

where $k, m \geq 0$, $k + m > 0$, $i_1, \ldots, i_k \leq n$, and $y_1, \ldots, y_n$ are distinct.

Part X

Tree Automata with Equality Constraints

Testing equality between brother subterms

in standard tree automata clauses, the variables $x_1, \ldots, x_n$ are pairwise distinct

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg)}$$

with variable sharing in

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(brother)}$$

we force equalities between brother subterm.

example:

$$\begin{array}{rcl} & \Rightarrow & Q(a), \\ Q(x_1), Q(x_2) & \Rightarrow & Q\big(f(x_1, x_2)\big), \\ Q(x), Q(x) & \Rightarrow & Q_{\mathsf{f}}\big(f(x, x)\big) \end{array}$$

Testing equality between brother subterms

expressiveness: tree automata with equality tests between brother subterms $\supsetneq$ bottom-up tree automata.

Theorem :

The emptiness problem is EXPTIME-complete for tree automata with equality tests between brother subterms.

Bogaert-Tison automates

Bruno Bogaert, Sophie Tison, 1992.
Tests of $=$ and $\neq$ between brother subterms (see chapter 4 TATA).

- determinizable in exponentiel time
- all Boolean closures
- emptiness decidable in PTIME for deterministic
- emptiness EXPTIME-complete for non-deterministic

Arbitrary equality tests

clauses with equality

$$Q_1(x_1), \ldots, Q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \tag{test}$$

where $k \geq 0$, $u_1, v_1, \ldots, u_k, v_k \in \mathcal{T}\big(\Sigma, \{x_1, \ldots, x_n\}\big)$.

without restrictions, emptiness is undecidable.

Arbitrary equality tests: decidable class

We distinguish some predicates call test predicates, and assume a partial ordering $\succ$ over predicates such that
for all Q, test predicate and Q_0 non-test, $Q \succ Q_0$.

$$Q_1(x_1), \ldots, Q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(test)}$$

where Q is a test predicate, and for all Q_i test predicate, $Q \succ Q_i$.

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg}')$$

where either all $Q, Q_1, \ldots, Q_n$ are not test predicates
or Q is a test predicate and at most one $Q_i = Q$ and the others are not test.

Arbitrary equality tests: decidable class

example: stuttering lists

$$\begin{array}{rlrcl} \Rightarrow & Q_0(0) & Q_0(x) & \Rightarrow & Q_0(s(x)) \\ \Rightarrow & Q_1([\,]) & Q_0(x), Q_1(y) & \Rightarrow & Q_1(cons(x,y)) \\ & & Q_0(x), Q_2(y) & \Rightarrow & Q_2(cons(x,y)) \end{array}$$

$$Q_0(x), Q_1(y), y = cons(x, y') \Rightarrow Q_2(cons(x, y))$$

Arbitrary equality tests: decidable class

Theorem :

The satisfiability of a set of clauses of type (test) and (reg') and a goal clause $Q(t) \Rightarrow \bot$ is decidable.

pr.: Saturation by ordered paramodulation with selection and ϵ-splitting.

Extension to langages (and equality tests) modulo equational theories, by adding clauses $\Rightarrow \ell = r$ (of a restricted form).

Part XI

Automates d'arbres à mémoire

Automates à pile

Extension des automates de mots finis (NFA) avec une mémoire non bornée = pile.

$\mathcal{A} = (\Sigma, \Gamma, Q, Q^{\mathsf{i}}, Q^{\mathsf{f}}, \delta)$.

- Σ: alphabet d'entrée,
- Γ: alphabet de pile.

$$\delta \subseteq \underbrace{Q \times \Sigma \times Q \times \Gamma}_{(push)} \cup \underbrace{Q \times \Sigma \times \Gamma \times Q}_{(pop)} \cup \underbrace{Q \times \Sigma \cup \{\varepsilon\} \times Q}_{(internal)}$$

Example :

$\Sigma = \{a, b\}$, $\Gamma = \{\alpha\}$.

$$\begin{aligned} push : &\quad q^{\mathsf{i}} \xrightarrow{a} q^{\mathsf{i}}, \alpha \\ internal : &\quad q^{\mathsf{i}} \xrightarrow{b} q^{\mathsf{f}} \\ pop : &\quad q^{\mathsf{f}} \xrightarrow{a,\alpha} q^{\mathsf{f}} \end{aligned}$$

reconnaît: $\{a^n b a^n \mid n \geq 0\}$.

Automates à pile (propriétés)

- expressivité $\supsetneq$ NFA
- même expressivité que les grammaires hors-contexte ($N := N_1 N_2$, $N := a$)
- vide décidable
- pas clos par intersection, complément
- la restriction visibly: $\Sigma = \Sigma_{push} \uplus \Sigma_{pop} \uplus \Sigma_{int}$ a les clôtures Booléennes.

Lemma :

Le langage des piles accessibles est régulier.

Automates d'arbres à pile

Automates d'arbres à pile [Guessarian 83]:

- Σ: signature d'entrée; en lecture: termes de $\mathcal{T}(\Sigma)$,
- Γ: alphabet de pile; mémoire auxiliaire $=$ pile de Γ^*,
- transitions descendantes:

$$\begin{array}{lrcl}
push & q(y) & \to & f\big(q_1(e_1(y)), \ldots, q_n(e_n(y))\big) \\
pop & q(e(y)) & \to & f\big(q_1(y), \ldots, q_n(y)\big) \\
pop & q(\bot) & \to & f\big(q_1(\bot), \ldots, q_n(\bot)\big) \\
int & q(y) & \to & f\big(q_1(y), \ldots, q_n(y)\big) \\
\varepsilon & q(y) & \to & q'(y)
\end{array}$$

- même expressivité que les grammaires d'arbres hors-contexte.

Automates d'arbres à une mémoire

Automates ascendants avec une mémoire auxiliaire contenant un arbre. $\mathcal{A} = (\Sigma, \Gamma, Q, Q^{\mathsf{f}}, \Delta)$.

- Σ: signature d'entrée; en lecture: termes de $\mathcal{T}(\Sigma)$,
- Γ: signature de pile; mémoire auxiliaire = terme de $\mathcal{T}(\Gamma)$.
- Δ: transitions de la forme

$$
\begin{array}{lllll}
push & a & & \to & q(c) \\
push & f\big(q_1(y_1), & q_2(y_2)\big) & \to & q\big(h(y_1, y_2)\big) \\
pop_{11} & f\big(q_1(h(y_{11}, y_{12})), & q_2(y_2)\big) & \to & q(y_{11}) \\
 & f\big(q_1(\bot), & q_2(y_2)\big) & \to & q(\bot) \\
pop_{12} & f\big(q_1(h(y_{11}, y_{12})), & q_2(y_2)\big) & \to & q(y_{12}) \\
 & f\big(q_1(\bot), & q_2(y_2)\big) & \to & q(\bot) \\
 & \vdots & & & \\
int_0 & a & & \to & q(\bot) \\
int_1 & f\big(q_1(y_1), & q_2(y_2)\big) & \to & q(y_1) \\
int_2 & f\big(q_1(y_1), & q_2(y_2)\big) & \to & q(y_2)
\end{array}
$$

Automates d'arbres à une mémoire (propriétés)

- expressivité $\supsetneq$ automates d'arbres ascendants
- généralise les automates (de mots) à pile
- vide décidable
- pas clos par intersection ni complément
- la restriction visibly a toutes les clôtures Booléennes.

$$\Sigma = \Sigma_{push} \uplus \Sigma_{pop_{11}} \uplus \Sigma_{pop_{12}} \uplus \Sigma_{pop_{21}} \uplus \Sigma_{pop_{22}} \uplus \Sigma_{int_0} \uplus \Sigma_{int_1} \uplus \Sigma_{int_2}$$

Décision du vide

Theorem :

Le vide est décidable en temps polynomial pour les automates à 1 mémoire.

- $L(\mathcal{A}, q) := \left\{ t \;\middle|\; \exists m \in \mathcal{T}(\Gamma), t \xrightarrow[\Delta]{*} q(m) \right\}$
- $M(\mathcal{A}, q) := \left\{ m \;\middle|\; \exists t \in \mathcal{T}(\Sigma), t \xrightarrow[\Delta]{*} q(m) \right\}$.

Lemma :

Pour tous $\mathcal{A}$, q, $L(\mathcal{A}, q) = \emptyset$ ssi $M(\mathcal{A}, q) = \emptyset$.

Lemma :

Pour tous $\mathcal{A}$, q, $M(\mathcal{A}, q)$ est un langage d'automate bidirectionnel.

Décision du vide (2)

$$
\begin{array}{lrcll}
push & f\big(q_1(y_1), q_2(y_2)\big) & \to & q\big(h(y_1,y_2)\big) & \\
 & Q_1(y_1), Q_2(y_2) & \Rightarrow & Q\big(h(y_1,y_2)\big) & (\mathsf{reg}) \\
pop_{11} & f\big(q_1(h(y_{11},y_{12})), q_2(y_2)\big) & \to & q(y_{11}) & \\
 & Q_1\big(h(y_{11},y_{12})\big), Q_2(y_2) & \Rightarrow & Q(y_{11}) & \\
split: & Q_1\big(h(y_{11},y_{12})\big), q_{Q_2} & \Rightarrow & Q(y_{11}) & (bidi) \\
 & Q_2(y_2) & \Rightarrow & q_{Q_2} & \\
int_1 & f\big(q_1(y_1), q_2(y_2)\big) & \to & q(y_1) & \\
 & Q_1(y_1), Q_2(y_2) & \Rightarrow & Q(y_1) & \\
split: & Q_1(y_1), q_{Q_2} & \Rightarrow & Q(y_1) & (\varepsilon) \\
 & Q_2(y_2) & \Rightarrow & q_{Q_2} &
\end{array}
$$

Automates d'arbres à une mémoire et contraintes

Version étendue avec contraintes $=$ et $\neq$ entre mémoires dans les transitions int.

$$
\begin{array}{lll}
int_1^{=} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1=y_2} \quad q(y_1) \\
int_1^{\neq} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1\neq y_2} \quad q(y_1) \\
int_2^{=} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1=y_2} \quad q(y_2) \\
int_2^{\neq} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1\neq y_2} \quad q(y_2)
\end{array}
$$

Automates d'arbres à une mémoire et contraintes: exemples

- arbres binaires équilibrés
- powerlists
- arbres (binaires) rouge-noir
  1. chaque noeud est noir ou rouge,
  2. la racine est noire,
  3. toutes les feuilles sont noires,
  4. les deux fils d'un noeud rouge sont noirs,
  5. tous les chemins contiennent le même nombre de noeuds noirs.

Automates d'arbres à une mémoire et contraintes: décision

Theorem :

Le vide est décidable en temps exponentiel pour les automates à 1 mémoire et contraintes.

pr.: Résolution ordonnée avec sélection et ε-splitting.