Tree automata techniques for the verification of infinite state-systems

Summer School VTSA 2011

Florent Jacquemard

INRIA Saclay & LSV (UMR CNRS/ENS Cachan)

florent.jacquemard@inria.fr
http://www.lsv.ens-cachan.fr/~jacquema

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

(chapters 1, 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

Automata on Finite Ranked Trees

Terms in first order logic

Plan

Terms

TA: Definitions and Expressiveness

Determinism and Boolean Closures

Decision Problems

Minimization

Closure under Tree Transformations, Program Verification

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 $t(p) \in \Sigma_n$ $(n \geq 1)$, $\left\{pj \in \mathcal{P}os(t) \;\middle|\; j \in \mathbb{N}\right\} = \{p1, ..., pn\}$,
- 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 :

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

Contexts

Definition : Contexte

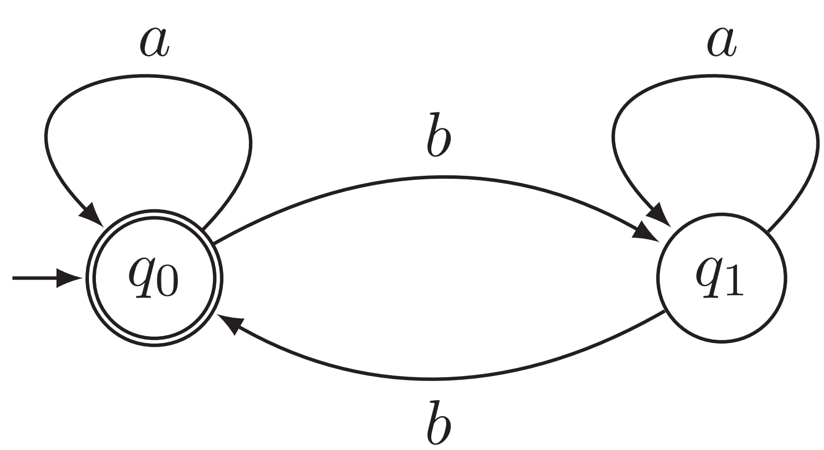
A *context* is a linear term.

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 obtained by the replacement of each x_i by t_i, for $1 \leq i \leq n$.

Plan

Bottom-up Finite Tree Automata

$(a + b\,a^*b)^*$



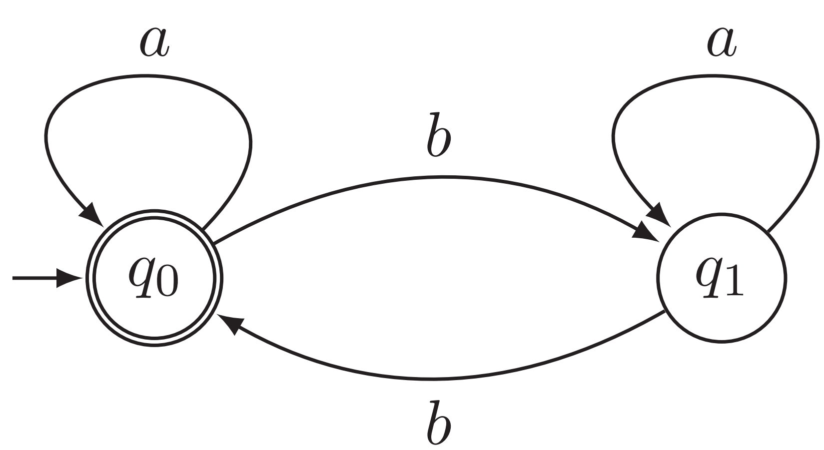
word. run on $aabba$: $q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_0 \xrightarrow{a} q_0$.

tree. run on $a(a(b(b(a(\varepsilon)))))$:
$q_0 \to a(q_0) \to a(a(q_0)) \to a(a(b(q_1))) \to a(a(b(b(q_0)))) \to$
$a(a(b(b(a(q_0))))) \to a(a(b(b(a(\varepsilon)))))$

with $q_0 := \varepsilon$, $q_0 := a(q_0)$, $q_1 := a(q_1)$, $q_1 := b(q_0)$, $q_0 := b(q_1)$.

Bottom-up Finite Tree Automata

$(a + b\,a^* b)^*$



word. run on $aabba$: $q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_0 \xrightarrow{a} q_0$.

tree. run on $a(a(b(b(a(\varepsilon)))))$:
$a(a(b(b(a(\varepsilon))))) \to a(a(b(b(a(q_0))))) \to a(a(b(b(q_0)))) \to$
$a(a(b(q_1))) \to a(a(q_0)) \to a(q_0) \to q_0$

with $\varepsilon \to q_0$, $a(q_0) \to q_0$, $a(q_1) \to q_1$, $b(q_0) \to q_1$, $b(q_1) \to q_0$.

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.
The language of $\mathcal{A}$ in state q is recursively defined by

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

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

We say that $t \in L(\mathcal{A}, q)$ is accepted, or recognized, by $\mathcal{A}$ in state q.

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

Recognized Languages: Operational Definition

Rewrite Relation

The rewrite relation associated to Δ is the smallest binary relation, denoted $\xrightarrow[\Delta]{}$, containing Δ and closed under application of contexts.

The reflexive and transitive closure of $\xrightarrow[\Delta]{}$ is denoted $\xrightarrow[\Delta]{*}$.

For $\mathcal{A} = (\Sigma, Q, Q^{\mathsf{f}}, \Delta)$, it holds that

$$L(\mathcal{A}, q) = \left\{ t \in \mathcal{T}(\Sigma) \;\middle|\; t \xrightarrow[\Delta]{*} q \right\}$$

and hence

$$L(\mathcal{A}) = \left\{ t \in \mathcal{T}(\Sigma) \;\middle|\; t \xrightarrow[\Delta]{*} q \in Q^{\mathsf{f}} \right\}$$

Tree Automata: example 1

Example :

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

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

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

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).

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

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.

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 $h(t) \leq |Q|$.

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

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

Plan

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.

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 $q_\perp$.

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. Transitions:

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

for all $S_1, \ldots, S_n \subseteq Q$.

Determinization (example)

Exercice :

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

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

Top-Down Tree Automata and Determinism

Definition : Determinism

A top-down tree automaton $(\Sigma, Q, Q^{\mathsf{init}}, \Delta)$ is *deterministic* if $|Q^{\mathsf{init}}| = 1$ 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.

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	computation time and size of automata
$\cup$	disjoint $\cup$	
$\cap$	Cartesian product	
$\neg$	determinization, completion, invert final / non-final states	(lower bound)

Remark :

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

Boolean Closure of Regular tree Languages

Proposition : Closure

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

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

Remark :

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

Boolean Closure of Regular tree Languages

Proposition : Closure

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

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

Remark :

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

Boolean Closure of Regular tree Languages

Proposition : Closure

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

op.	technique	computation time 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 Algorithm

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 $q_1, \ldots, q_n \in M$ st 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.

Lemma :

$q \in M_*$ iff $\exists t \in L(\mathcal{A}, q)$.

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.

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.

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.

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.

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) $M = (\Gamma, S, s_0, S_\mathsf{f}, \delta)$.
[Seidl 94], [Veanes 97]

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.

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.

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.

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 $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)} = \emptyset$.

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.

Plan

- Terms
- TA: Definitions and Expressiveness
- Determinism and Boolean Closures
- Decision Problems
- Minimization
- Closure under Tree Transformations, Program Verification

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 Theorem of Myhill-Nerode

$1 \Rightarrow 2$. $\mathcal{A}$ 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_{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 $s \in L$.

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 $L(\mathcal{A}_{\mathsf{min}}) = L(\mathcal{A})$.

Minimization

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 $\mathcal{A}$.