よく効く！ツリーオートマトン

An Introduction to Tree Automata and the Recent Trend

Hitoshi Ohsaki

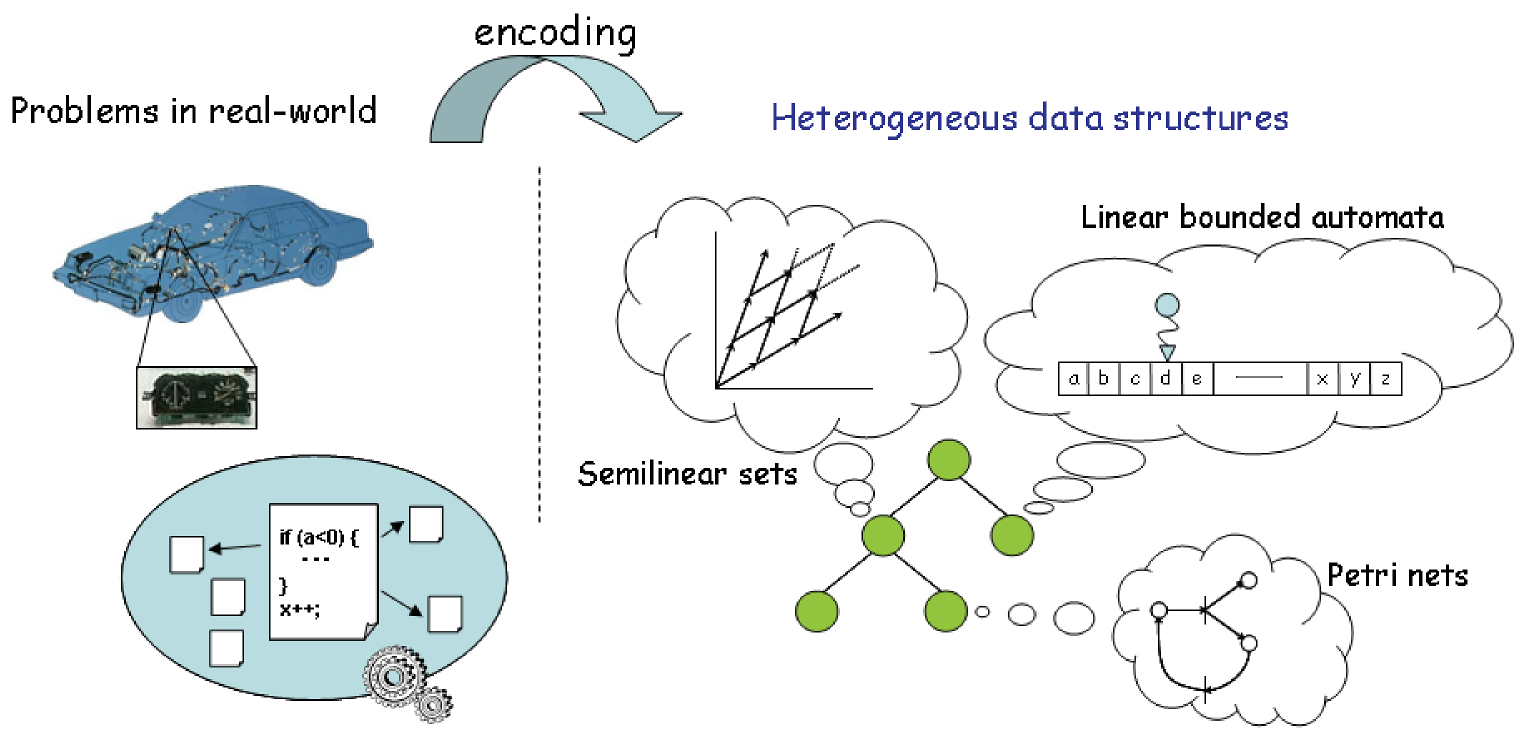
AIST & JST

8th PPL (Ogoto)

March 2006

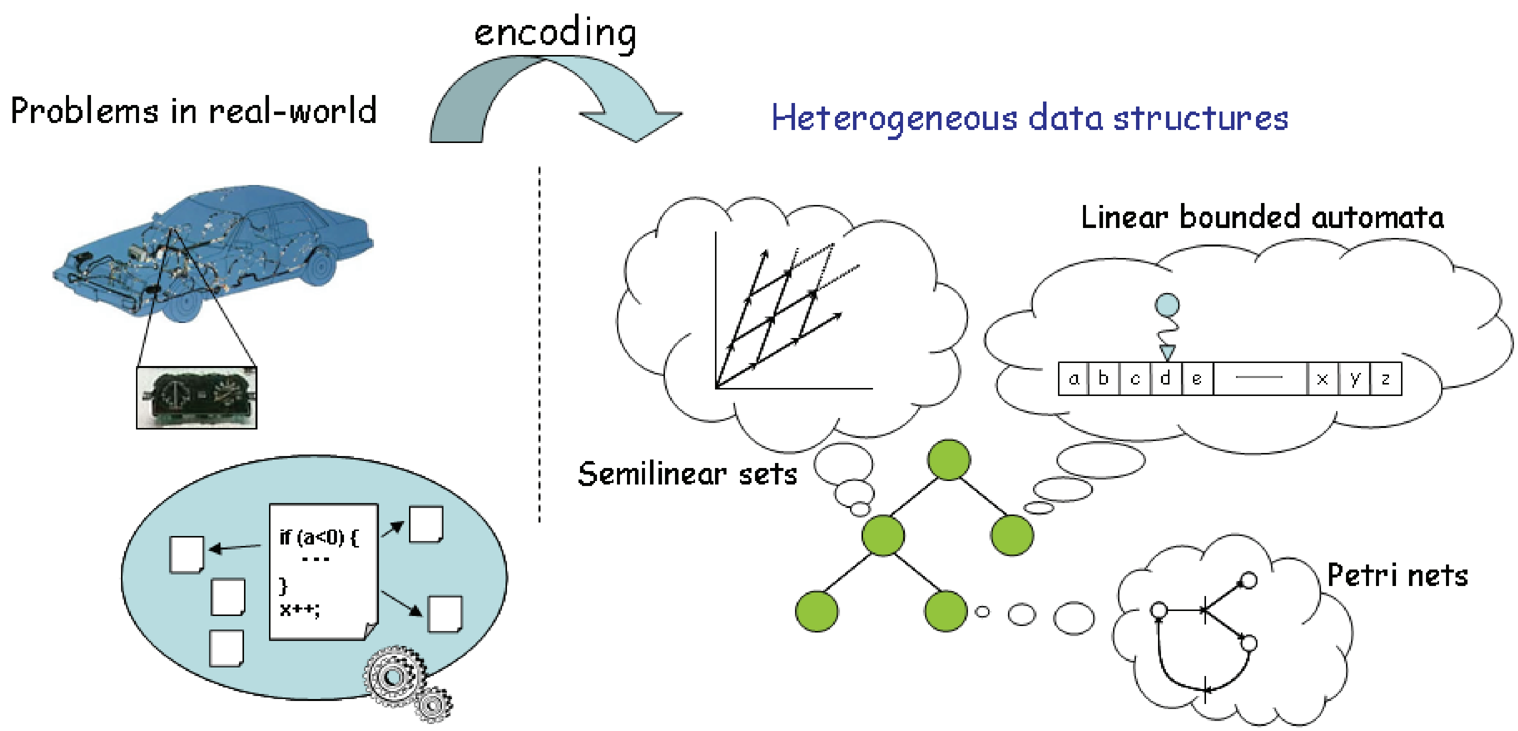
Why tree automata ?   Why not tree automata ?

- structures
- algebraic properties
- decidability
- semantics

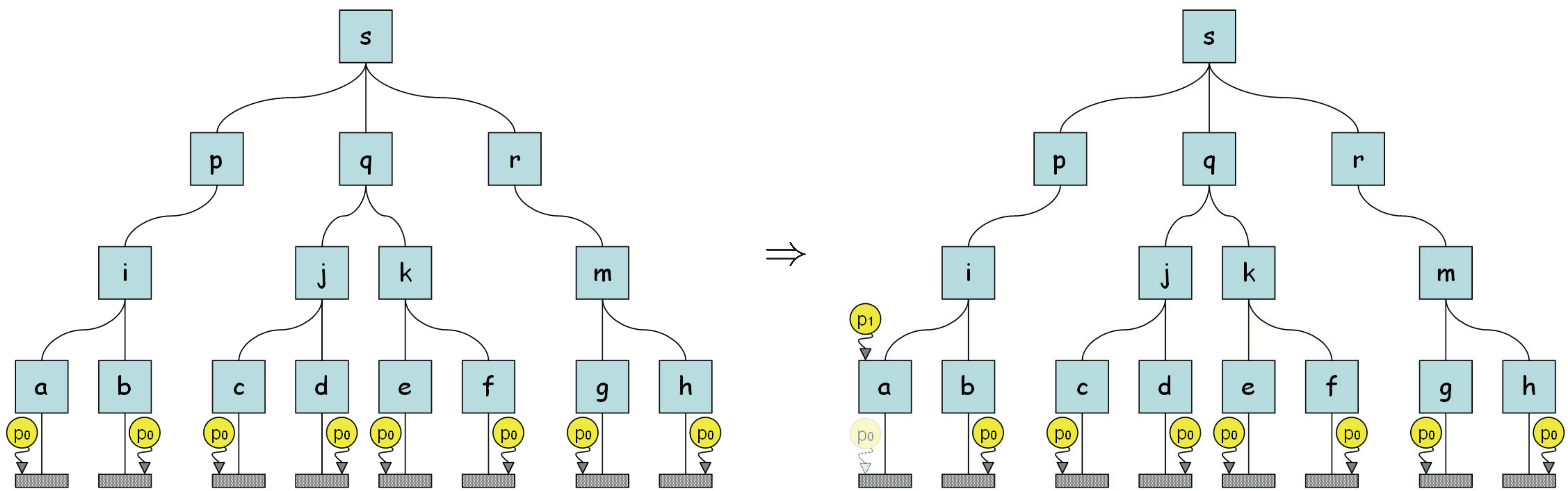


Why tree automata ? Why not tree automata ?

- structures
- algebraic properties
- decidability
- semantics

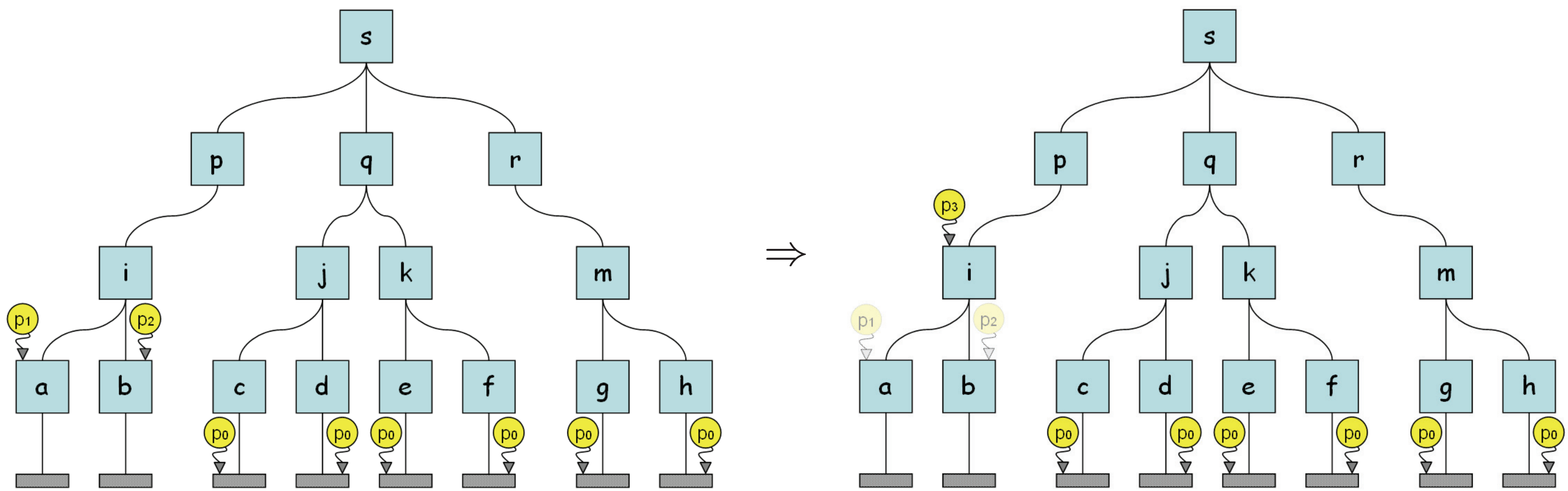


Automata for trees

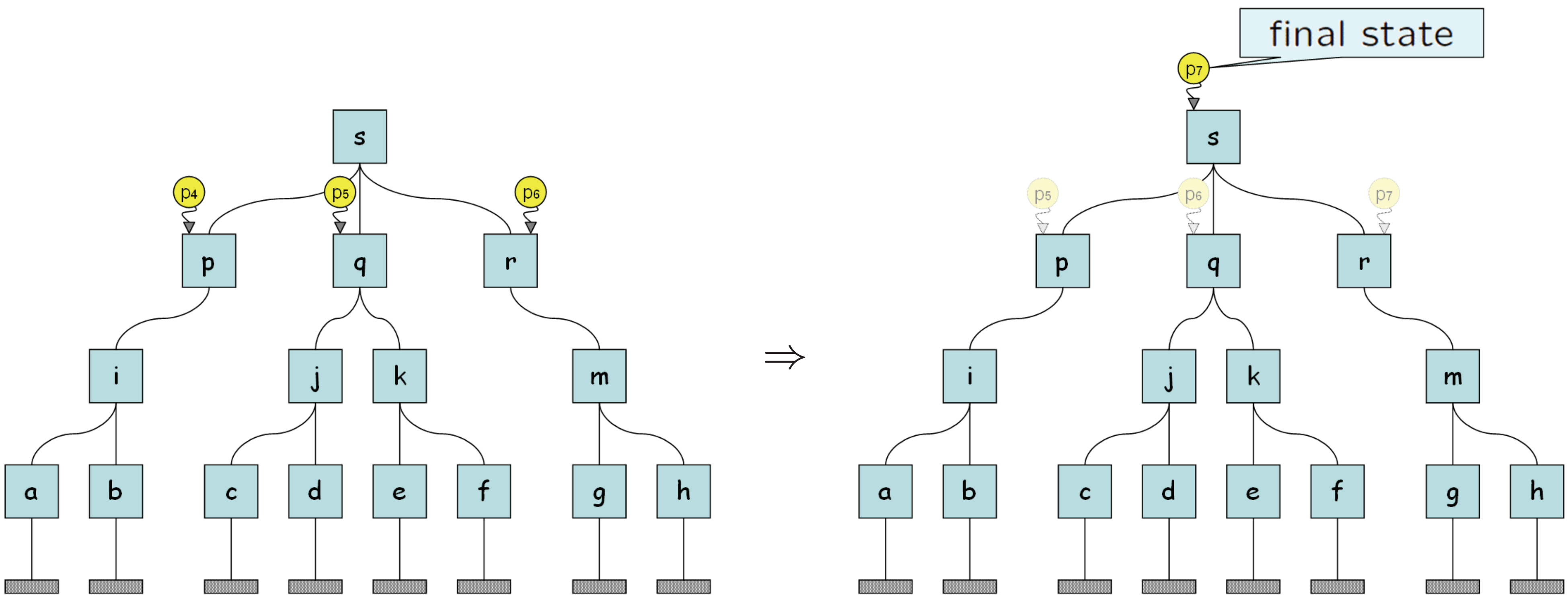


initial configuration

Automata for trees

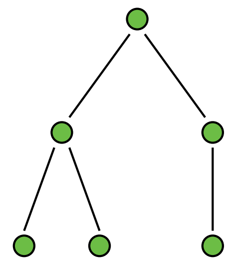


Automata for trees



final configuration

Tree automata vs. automata

	tree automata	automata
input		
transition rules	$\mathsf{f}(\alpha_1, \ldots, \alpha_n) \longrightarrow \beta$ $\mathsf{f}(\alpha_1, \ldots, \alpha_n) \longrightarrow \mathsf{f}(\beta_1, \ldots, \beta_n)$ $\alpha \longrightarrow \beta$	$\alpha \xrightarrow{\mathsf{f}} \beta$ $\alpha \longrightarrow \beta$
closure properties	$\cup \quad \cap \quad (\ )^{\mathsf{c}}$	$\cup \quad \cap \quad (\ )^{\mathsf{c}}$
decidability	$\in \quad \subseteq \quad = \varnothing?$	$\in \quad \subseteq \quad = \varnothing?$

Definition

$\mathcal{A}$: *tree automaton* $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$

$\mathcal{F}$	set of function symbols with fixed arity (signature)
$\mathcal{Q}$	set of state symbols such that $\mathcal{F} \cap \mathcal{Q} = \varnothing$
$\mathcal{Q}_{fin}$	set of final state symbols such that $\mathcal{Q}_{fin} \subseteq \mathcal{Q}$
Δ	set of transition rules with the following forms :

$$f(p_1, \ldots, p_n) \rightarrow q_1 \qquad \text{(TYPE 1)}$$

$$f(p_1, \ldots, p_n) \rightarrow f(q_1, \ldots, q_n) \qquad \text{(TYPE 2)}$$

$$p_1 \rightarrow q_1 \qquad \text{(TYPE 3)}$$

for some $f \in \mathcal{F}$ $\quad p_1, \ldots, p_n, q_1, \ldots, q_n \in \mathcal{Q}$

Transition move

- $\rightarrow_{\mathcal{A}}$ move relation of tree automaton :

  $s \rightarrow_{\mathcal{A}} t$ if $s = C[l]$ and $t = C[r]$

  for some $l \rightarrow r$ in Δ and context C

  E.g. Consider $\mathcal{A}$ with transition rules Δ :

  $\mathsf{a} \rightarrow q_1 \quad \mathsf{b} \rightarrow q_2 \quad \mathsf{f}(q_1, q_2) \rightarrow q_3$

  then

  $\mathsf{f}(\mathsf{a}, \mathsf{b}) \rightarrow_{\mathcal{A}} \mathsf{f}(q_1, \mathsf{b}) \rightarrow_{\mathcal{A}} \mathsf{f}(q_1, q_2) \rightarrow_{\mathcal{A}} q_3$

- $\mathcal{L}(\mathcal{A})$ set of trees reachable by $\mathcal{A}$ to final state

  E.g.

  $\mathsf{f}(\mathsf{a}, \mathsf{b})$ accepted if q_3 is final state

  $\{\ \mathsf{f}(\mathsf{a}, \mathsf{b})\ \}$ language accepted by $\mathcal{A}$

Basic properties

- Epsilon-rule elimination
- Union
- Intersection
- Complementation
  - Deterministic and complete tree automata
  - Downsizing technique (cf. Myhill-Nerode theorem)
- Emptiness problem
  - Pumping lemma

Eliminating transition rules of Types 2 & 3

A tree automaton $\mathcal{A} = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$ is *regular* if Δ consists of (Type 1)-transition rules

Theorem

Given $\mathcal{A}$: tree automaton over $\mathcal{F}$

$\exists$ $\mathcal{B}$: regular tree automaton such that $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$

Proof sketch

Define $\Delta_{\mathcal{B}}$ as follows: $f(p_1, \ldots, p_n) \to p$ in $\Delta_{\mathcal{B}}$ if and only if

$$f(p_1, \ldots, p_n) \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} f(q_1, \ldots, q_n) \to_{\mathcal{A}} q \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} p$$

for some $f \in \mathcal{F}$ and $p_1, \ldots, p_n, p \in \mathcal{Q}$

Note Optimal algorithm for this computation runs in P-time relative to $|\mathcal{A}|$

Deterministic & complete tree automata

Given $\mathcal{A}$: regular tree automaton over $\mathcal{F}$

$\exists$ $\mathcal{B}$: deterministic and complete regular tree automaton

such that $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$

Proof sketch

Define a tree automaton $\mathcal{A}_\mathsf{d}$ as follows:

$$\mathcal{Q}_\mathsf{d} = 2^{\mathcal{Q}}$$

$$\mathcal{Q}_{\mathsf{d}\,fin} = \{ A \in \mathcal{Q}_\mathsf{d} \mid A \cap \mathcal{Q}_{fin} \neq \varnothing \}$$

$$\Delta_\mathsf{d} = \{f(A_1, \ldots, A_n) \to A \mid$$

(1) $A_1, \ldots, A_n \in \mathcal{Q}_\mathsf{d}$

(2) $A = \{q \mid \exists q_1 \in A_1, \ \ldots \ \exists q_n \in A_n, \ \exists f(q_1, \ldots, q_n) \to q \in \Delta\} \}$

By construction $\mathcal{A}_\mathsf{d}$ is regular, deterministic and complete

Moreover, $\mathcal{A}_\mathsf{d}$ satisfies $\mathcal{L}(\mathcal{A}_\mathsf{d}) = \mathcal{L}(\mathcal{A})$

Tree automata and context-free grammar

Given $\mathcal{G}$: context-free grammar in Chomsky normal form over Σ

$\exists$ $\mathcal{A}$: regular tree automaton over $\{\text{f}\} \cup \Sigma$

such that $\mathcal{A}$ simulates run($\mathcal{G}$)

Proof sketch

Define $\mathcal{A}$ to be

(1) $\text{f}(\alpha, \beta) \to \gamma$ in $\mathcal{A}$ iff $\gamma \to \alpha\, \beta$ in $\mathcal{G}$

(2) $\text{a} \to \gamma$ in $\mathcal{A}$ iff $\gamma \to \text{a}$ in $\mathcal{G}$

<u>Note</u> Grammar is not necessarily in Chomsky normal form

$\Rightarrow$ f is replaced by f_2 f_3 $\ldots$ f_n

Observation

leaf($\mathcal{A}$) is context-free language when $\mathcal{A}$ is regular tree automaton

Pumping Lemma for tree automata

Given $\mathcal{A}$: tree automaton

t is accepted by $\mathcal{A}$

&

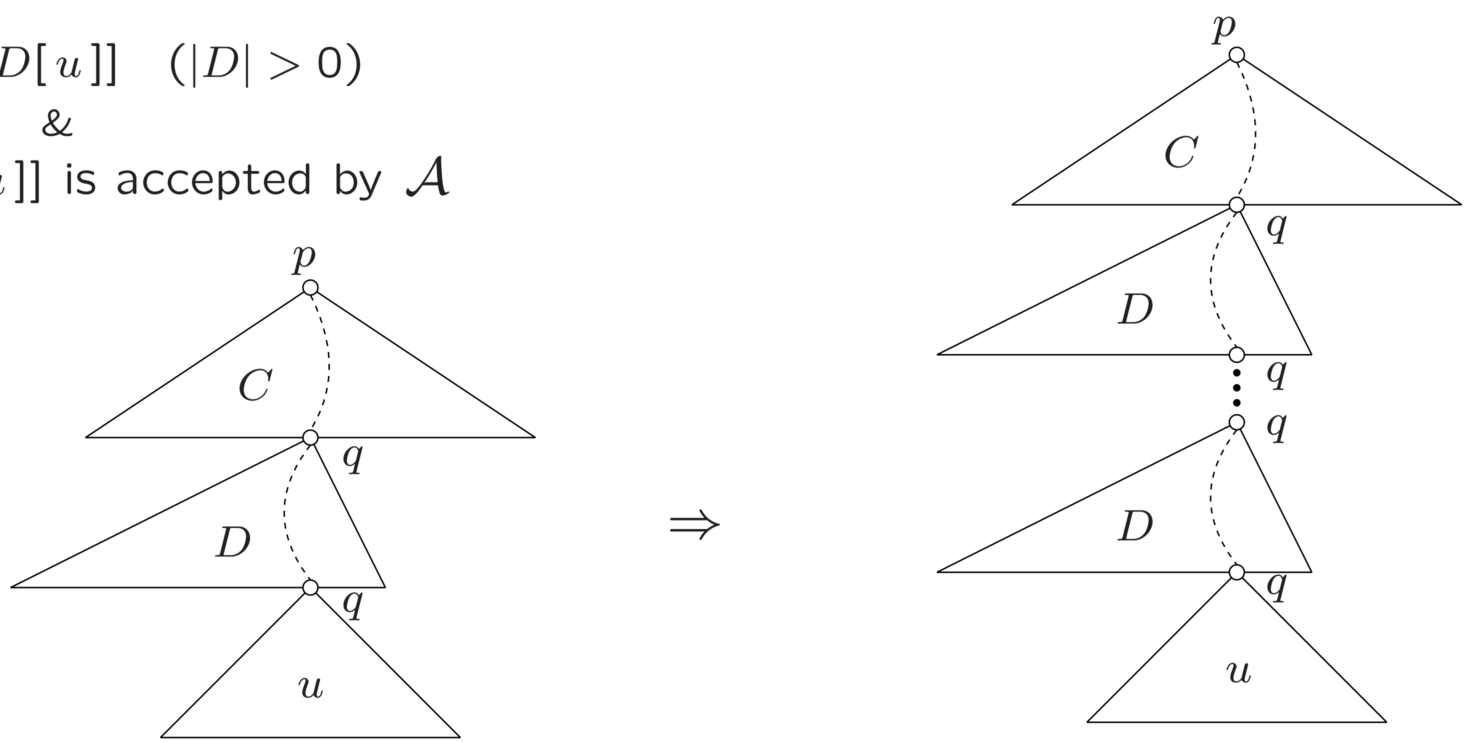
depth$(t) > \min(|\mathcal{Q}|, |\Delta|)$

implies

$t = C[\,D[\,u\,]]\quad (|D| > 0)$

&

$C[\,D^n[\,u\,]]$ is accepted by $\mathcal{A}$



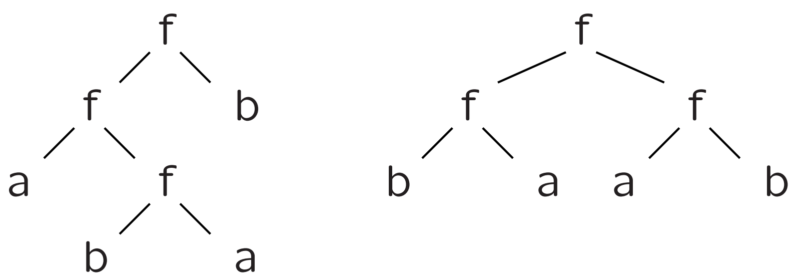
Cf. $uvxyw$-theorem for context-free grammar

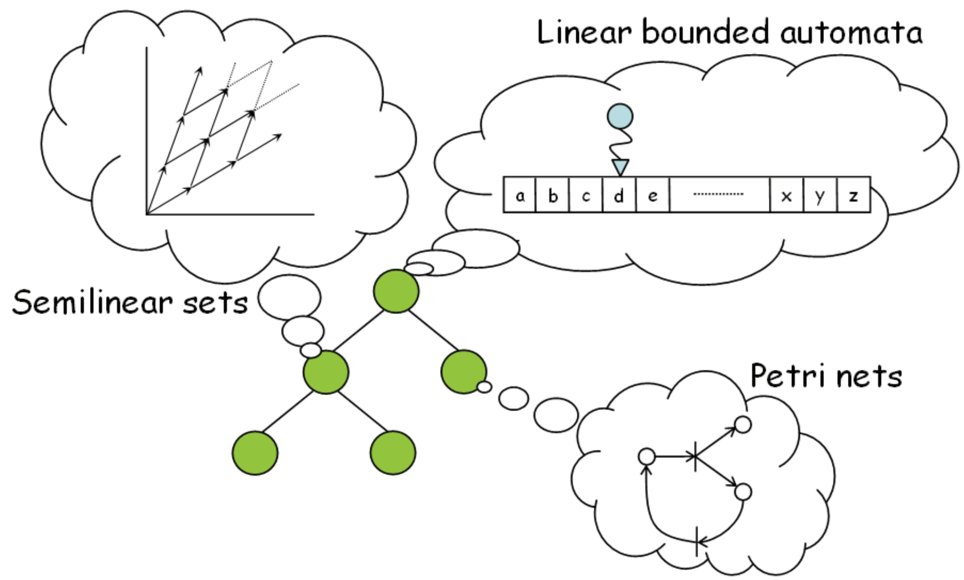
Linear equational constraints

Consider the language over $\mathcal{F} = \{\ \mathsf{f}\ \ \mathsf{a}\ \ \mathsf{b}\ \}$

$$L = \{\ t \mid \|\, t \,\|_{\mathsf{a}} = \|\, t \,\|_{\mathsf{b}}\ \}$$

such as





then

L is not accepted by any tree automaton

Linear equational constraints

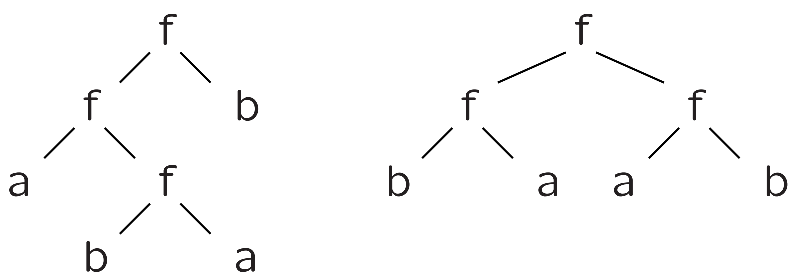
Consider the language over $\mathcal{F} = \{ \text{f} \quad \text{a} \quad \text{b} \}$

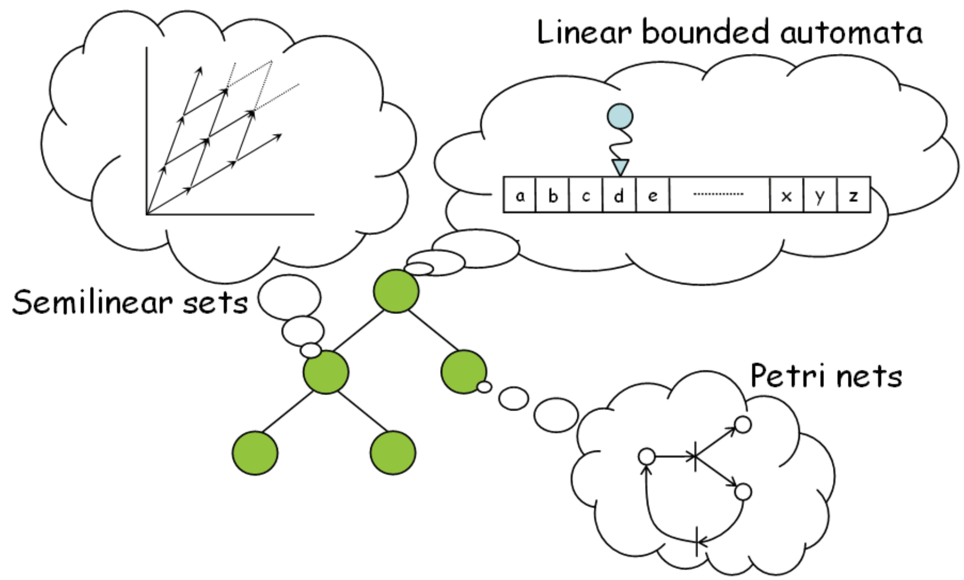
equation $V = \{ x \quad y \}$

$$L = \{ \; t \mid \| t \|_{\text{a}} = \| t \|_{\text{b}} \}$$

$$x = y$$

such as





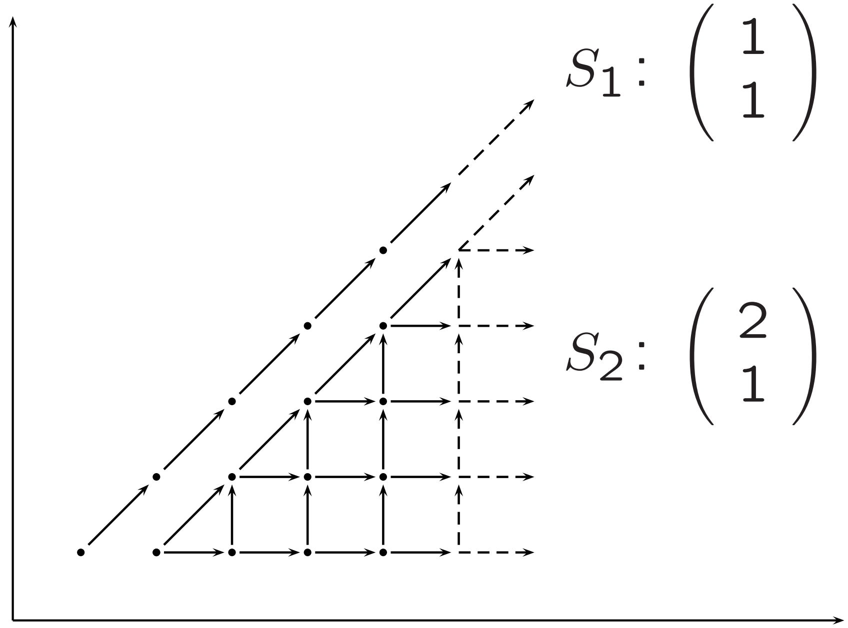
then

L is not accepted by any tree automaton

Commutative grammar and linear sets

S $(\subseteq \mathbb{N}^n)$ is linear set if $\exists$ vectors c p_1 p_2 $\ldots$ p_k in $\mathbb{N}^n$ such that

$$S = \left\{ v \;\middle|\; \begin{array}{l} \exists x_1\ x_2\ \cdots\ x_m \in \mathbb{N} \\ v = c + x_1 \cdot p_1 + x_2 \cdot p_2 + \cdots + x_k \cdot p_k \end{array} \right\}$$



$$S_1 : \begin{pmatrix} 1 \\ 1 \end{pmatrix} + x_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$S_2 : \begin{pmatrix} 2 \\ 1 \end{pmatrix} + x_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + x_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

A finite union of linear sets is called a semi-linear set (e.g. $S_1 \cup S_2$)

Parikh's mapping

Given $\Sigma = \{ \mathsf{a}_1 \ \ \mathsf{a}_2 \ \ \cdots \ \ \mathsf{a}_n \}$

Parikh image $\Psi_\Sigma : \Sigma^* \rightarrow \mathbb{N}^n$ such that

$$\Psi_\Sigma(w) = \begin{pmatrix} \sharp\mathsf{a}_1(w) \\ \sharp\mathsf{a}_2(w) \\ \cdots \\ \sharp\mathsf{a}_n(w) \end{pmatrix}$$

$\sharp\mathsf{a}_i(w)$ denotes the number of occurrences of a_i in w

Theorem [Parikh, Ginsburg 1966]

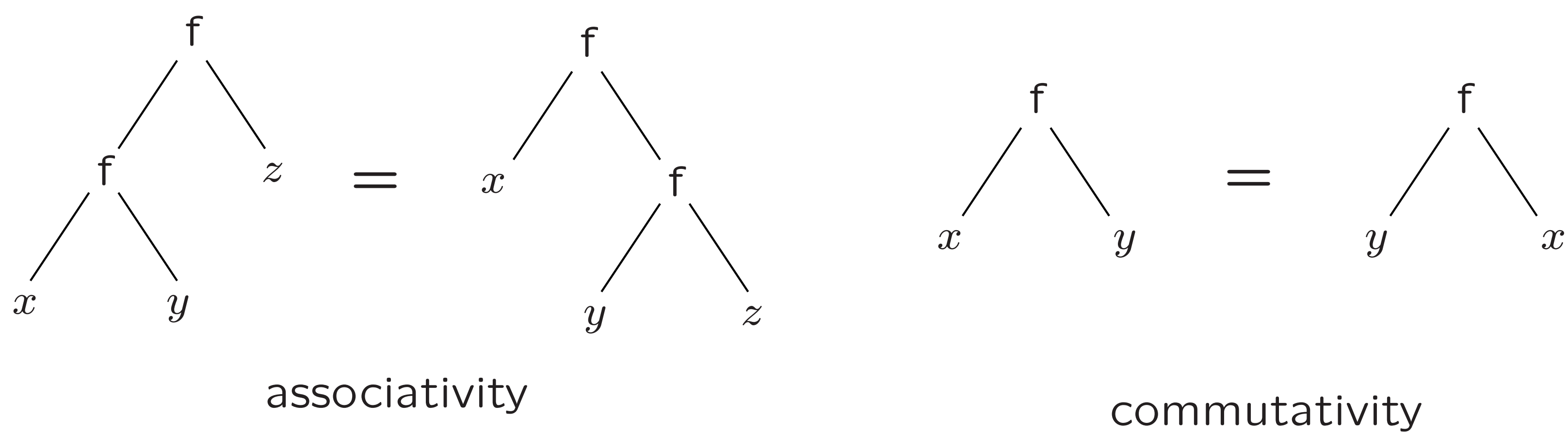
$\forall\, L$: commutative language, i.e $L = C(L)$

Parikh image $\Psi(L)$ is semi-linear iff

$\exists\, M$: context-free language such that $L = C(M)$

AC-axioms in tree structure

Suppose A (associativity) and C (commutativity) for f in the previous example :



then

L is *AC-closure* of the following tree language L'

$\mathsf{f}(\mathsf{a},\mathsf{b}) \in L'$

$\mathsf{f}(t_1, t_2) \in L'$ if $t_1, t_2 \in L'$

Note

L' is tree language accepted by tree automaton

Equational tree automata

$\mathcal{A}\,/\,\mathcal{E}$: *equational tree automaton*

$\mathcal{A}$	tree automaton $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$
$\mathcal{E}$	set of equations over $\mathcal{F}$ with $\mathcal{V}$

In particular

	(notation)	(name)
$\mathcal{E} = \mathsf{AC}$ (set of AC-axioms)	$\mathcal{A}/\mathsf{AC}$	monotone AC-tree automaton
	$\mathcal{L}(\mathcal{A}/\mathsf{AC})$	AC-monotone tree language
$\mathcal{E} = \mathsf{A}$ (set of A-axioms)	$\mathcal{A}/\mathsf{A}$	monotone A-tree automaton
	$\mathcal{L}(\mathcal{A}/\mathsf{A})$	A-monotone tree language

If Δ consists only of (TYPE 1) transition rules

$\mathcal{A}/\mathsf{AC}$	regular AC-tree automaton
$\mathcal{L}(\mathcal{A}/\mathsf{AC})$	AC-regular tree language

Transition move (in equational case)

- $\rightarrow_{\mathcal{A}/\mathcal{E}}$ move relation of equational tree automaton :

$$s \ \rightarrow_{\mathcal{A}/\mathcal{E}} \ t \quad \text{if} \quad s =_{\mathcal{E}} C[l] \ \text{ and } \ t =_{\mathcal{E}} C[r]$$

$$\text{for some } \ l \rightarrow r \ \text{ in } \Delta \ \text{ and context } C$$

E.g. Consider $\mathcal{A}$ with transition rules Δ and $\mathcal{F}_{\mathsf{AC}} = \{ \mathsf{f} \}$:

$$\mathsf{a} \rightarrow q_1 \qquad \mathsf{b} \rightarrow q_2 \qquad \mathsf{f}(q_1, q_2) \rightarrow q_3$$

then

$$\mathsf{f}(\mathsf{b}, \mathsf{a}) \ \rightarrow_{\mathcal{A}/\mathsf{AC}} \ \mathsf{f}(q_2, \mathsf{a}) \ \rightarrow_{\mathcal{A}/\mathsf{AC}} \ \underline{\mathsf{f}(q_2, q_1)} \ \rightarrow_{\mathcal{A}/\mathsf{AC}} \ q_3$$

- $\mathcal{L}(\mathcal{A}/\mathcal{E})$ set of trees reachable by $\mathcal{A}/\mathcal{E}$ to final state

E.g.

$\mathsf{f}(\mathsf{b}, \mathsf{a})$	accepted if q_3 is final state
$\{ \ \mathsf{f}(\mathsf{a}, \mathsf{b}) \ \ \mathsf{f}(\mathsf{b}, \mathsf{a}) \ \}$	language accepted by $\mathcal{A}/\mathsf{AC}$

Closure under Boolean operations

[Ohsaki CSL'01, Ohsaki & Takai RTA'02
Ohsaki & Seki & Takai RTA'03
Ohsaki & Talbot & Tison & Roos LPAR'05]

	regular	AC-regular	AC-monotone
closed under $\cup$	✓	✓	✓
closed under $\cap$	✓	✓	✓
closed under $(\,)^c$	✓	✓	×

regular TA $<$ regular AC-TA $<$ monotone AC-TA

commutative CFG   commutative CSG

	regular	A-regular	A-monotone
closed under $\cup$	✓	✓	✓
closed under $\cap$	✓	×	✓
closed under $(\,)^c$	✓	×	✓

regular TA $<$ regular A-TA $<$ monotone A-TA

CFG   CSG

Decidability results

	regular	AC-regular	AC-monotone
$t \in \mathcal{L}(\mathcal{A}/\mathrm{AC})$?	✓ (LOGCFL)	✓ (NP-complete)	✓ (PSPACE-compl.)
$\mathcal{L}(\mathcal{A}/\mathrm{AC}) = \varnothing$?	✓	✓	✓
$\mathcal{L}(\mathcal{A}/\mathrm{AC}) \subseteq \mathcal{L}(\mathcal{B}/\mathrm{AC})$?	✓	✓	×

	regular	A-regular	A-monotone
$t \in \mathcal{L}(\mathcal{A}/\mathrm{A})$?	✓ (LOGCFL)	✓ (P-time)	✓ (PSPACE-compl.)
$\mathcal{L}(\mathcal{A}/\mathrm{A}) = \varnothing$?	✓	✓	×
$\mathcal{L}(\mathcal{A}/\mathrm{A}) \subseteq \mathcal{L}(\mathcal{B}/\mathrm{A})$?	✓	×	×

Note Universality problem for monotone AC-tree automata remains open

See `http://www.lsv.ens-cachan.fr/rtaloop/problems/101.html`

Proof idea of non-closedness under complement

Given a signature $\mathcal{F} = \{\,\mathsf{f}\,\} \cup \{\,\mathsf{a}_1, \ldots, \mathsf{a}_n\,\}$

P: conjunction of C arithmetic constraints over positive integers $\mathbb{N}_+$:

$$
\begin{array}{rcll}
C & := & x_i = c & (c \ : \ \text{fixed natural number}) \\
 & \mid & x_i + x_j = x_k & \\
 & \mid & x_i \times x_j = x_k &
\end{array}
$$

such that $i, j, k \leqslant n$ and $k \neq i, j$

L_P: tree language over $\mathcal{F}$ whose Parikh's image satisfies P, meaning that

for each $t \in L_P$, $\sharp(t) = (\,\|t\|_{\mathsf{a}_1}, \ldots, \|t\|_{\mathsf{a}_n}\,)$ is a solution of P

Suppose $L_{x_i \times x_j \leqslant x_k}$ is accepted by monotone AC-TA then

- L_P is accepted by monotone AC-TA
- $L_P \neq \varnothing$ iff $\exists\,(x_1, \ldots, x_n)$ in $\mathbb{N}_+^n$: $P(x_1, \ldots, x_n) = \text{true}$

"$L_P \neq \varnothing\,?$" is decidable

but then it contradicts to the undecidability of Hilbert's 10th problem $\square$

Proof idea of non-closedness under complement

Given a signature $\mathcal{F} = \{\,\mathsf{f}\,\} \cup \{\,\mathsf{a}_1, \dots, \mathsf{a}_n\,\}$

P: conjunction of C arithmetic constraints over positive integers $\mathbb{N}_+$:

$$C \;:=\; x_i = c \qquad (c\ :\ \text{fixed natural number})$$
$$\mid\; x_i + x_j = x_k$$
$$\mid\; x_i \times x_j = x_k$$

such that $i, j, k \leqslant n$ and $k \neq i, j$

L_P: tree language over $\mathcal{F}$ whose Parikh's image satisfies P, meaning that

for each $t \in L_P$, $\sharp(t) = (\,\|t\|_{\mathsf{a}_1}, \dots, \|t\|_{\mathsf{a}_n}\,)$ is a solution of P

Suppose $L_{x_i \times x_j \leqslant x_k}$ is accepted by monotone AC-TA then

- L_P is accepted by monotone AC-TA
- $L_P \neq \varnothing$ iff $\exists\,(x_1, \dots, x_n)$ in $\mathbb{N}_+^n$: $P(x_1, \dots, x_n) = \text{true}$

"$L_P \neq \varnothing$?" is decidable

but then it contradicts to the undecidability of Hilbert's 10th problem $\square$

Lemma 1

There exists $\mathcal{A}/\mathsf{AC}$ over $\mathcal{F} = \{\mathsf{f}\} \cup \{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$ with $\mathcal{F}_{\mathsf{AC}} = \{\mathsf{f}\}$ such that Parikh's image of $\mathcal{L}(\mathcal{A}/\mathsf{AC})$ satisfies $x_i \times x_j \geqslant x_k \ \ (i, j, k \leqslant n$ and $k \neq i, j)$

Proof Example of $\mathcal{A}/\mathsf{AC}$ is found in our paper [Ohsaki *et al.* LPAR'05] □

Lemma 2

There exists $\mathcal{B}/\mathsf{AC}$ that represents $x_i \times x_j > x_k \ \ (i, j, k \leqslant n$ and $k \neq i, j)$

Proof Example of $\mathcal{B}/\mathsf{AC}$ over the same $\mathcal{F}$ is exhibited □

Suppose $\exists\ \mathcal{C}/\mathsf{AC}$ over $\mathcal{F}$ that represents $x_i \times x_j \leqslant x_k$

then $\exists\ \mathcal{D}/\mathsf{AC}$ over $\mathcal{F}$ that represents $x_i \times x_j = x_k$ ($\because$ Lemma 1)

It admits $\mathcal{M}$ determining, for arbitrary constraint P

- "yes" if P has a solution
- "no" otherwise

Note $\mathcal{L}(\mathcal{C}/\mathsf{AC})$ is the complement of $\mathcal{L}(\mathcal{B}/\mathsf{AC})$ (cf. Lemma 2)

Theorem 1

AC-monotone tree languages are not closed under complementation □

Corollary 1

regular AC-TA $<$ monotone AC-TA

Proof

- regular AC-TA $\leqslant$ monotone AC-TA (by definition)
- the class of regular AC-TA is closed under Boolean operations

(another proof)

Suppose $\mathcal{F} = \{\mathsf{f}\} \cup \{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$ with $\mathcal{F}_{\mathsf{AC}} = \{\mathsf{f}\}$

then

L : AC-regular tree language iff Parikh's image $\sharp(L)$: semilinear

Tree language representing $x_i \times x_j \geqslant x_k$ is not AC-regular □

Theorem 2

The inclusion problem for monotone AC-TA is undecidable

Proof

Suppose $P \equiv (p_1 = q_1) \wedge \cdots \wedge (p_k = q_k)$ over $\{ x_1, \ldots, x_n \}$

Let

$$P_{\geqslant} \equiv (p_1 \geqslant q_1) \wedge \cdots \wedge (p_i \geqslant q_i) \wedge \cdots \wedge (p_k \geqslant q_k)$$
$$Q_i \equiv (p_1 \geqslant q_1) \wedge \cdots \wedge (p_i > q_i) \wedge \cdots \wedge (p_k \geqslant q_k)$$

then

$$\exists (x_1, \ldots, x_n) : P(x_1, \ldots, x_n) = \text{true} \quad \text{iff} \quad \exists (x_1, \ldots, x_n) : P_{\geqslant}(x_1, \ldots, x_n) = \text{true} \wedge \bigwedge_{1 \leqslant i \leqslant k} Q_i(x_1, \ldots, x_n) = \text{false}$$

$$\text{iff} \quad L_{P_{\geqslant}} \not\subseteq \bigcup_{1 \leqslant i \leqslant k} L_{Q_i}$$

$L_{P_{\geqslant}}$ L_{Q_i} $(1 \leqslant i \leqslant k)$: AC-monotone ($\because$ Lemma 1 & Lemma 2) $\square$

Theorem 3

The membership problem $t \in \mathcal{L}(\mathcal{A}/\mathsf{AC})$ for monotone AC-TA is PSPACE-complete

Proof

- PSPACE : This problem is solvable with polynomially space-bounded TM

  In fact, e.g. the question "$t \in \mathcal{L}(\mathcal{A}/\mathsf{AC})$?" is $<^P$-reducible to the membership problem $t \in \mathcal{L}(\mathcal{B}_{\mathcal{A}}/\mathsf{A})$ for monotone A-TA

  Note 1 the membership problem for monotone A-TA is PSPACE-complete

  Note 2 PSPACE is closed under $<^P$

- PSPACE-hardness : Use QBF (quantified Boolean formula) problem

  Note 3 To determine whether Φ is valid is PSPACE-complete

$$\Phi \;:=\; x \;\mid\; \neg\Phi \;\mid\; \Phi \wedge \Phi \;\mid\; \exists x : \Phi$$

Proof (cont'd)

Given QBF Φ

we can construct t_Φ and $\mathcal{A}_\Phi/\mathsf{AC}$ in linear time

such that

$$\Phi \text{ is valid} \quad \text{iff} \quad t_\Phi \in \mathcal{L}(\mathcal{A}_\Phi/\mathsf{AC})$$

(another proof suggested by LPAR'05 referee)

Use reachability problem for 1-conservative Petri nets :

- this problem is PSPACE-complete
- given Petri net N and the initial and final configurations m m' they are linear-time reducible to t_m and $\mathcal{A}_{N,m'}/\mathsf{AC}$ such that

$$m \ \to_N^* \ m' \quad \text{iff} \quad t_m \in \mathcal{L}(\mathcal{A}_{N,m'}/\mathsf{AC})$$

□

Related work

Verma & Goubault-Larrecq [RTA'03]

Alternating two-way AC-tree automata

Seidl & Schwentick & Muscholl [PODS'03]

Presburger tree automata

Lugiez [FOSSACS'03]

Multitree automata with counting and equality constraints

Comon-Lundh & Cortier [RTA'03]

Narrowing technique manipulating xor (A, C, U, X) theory

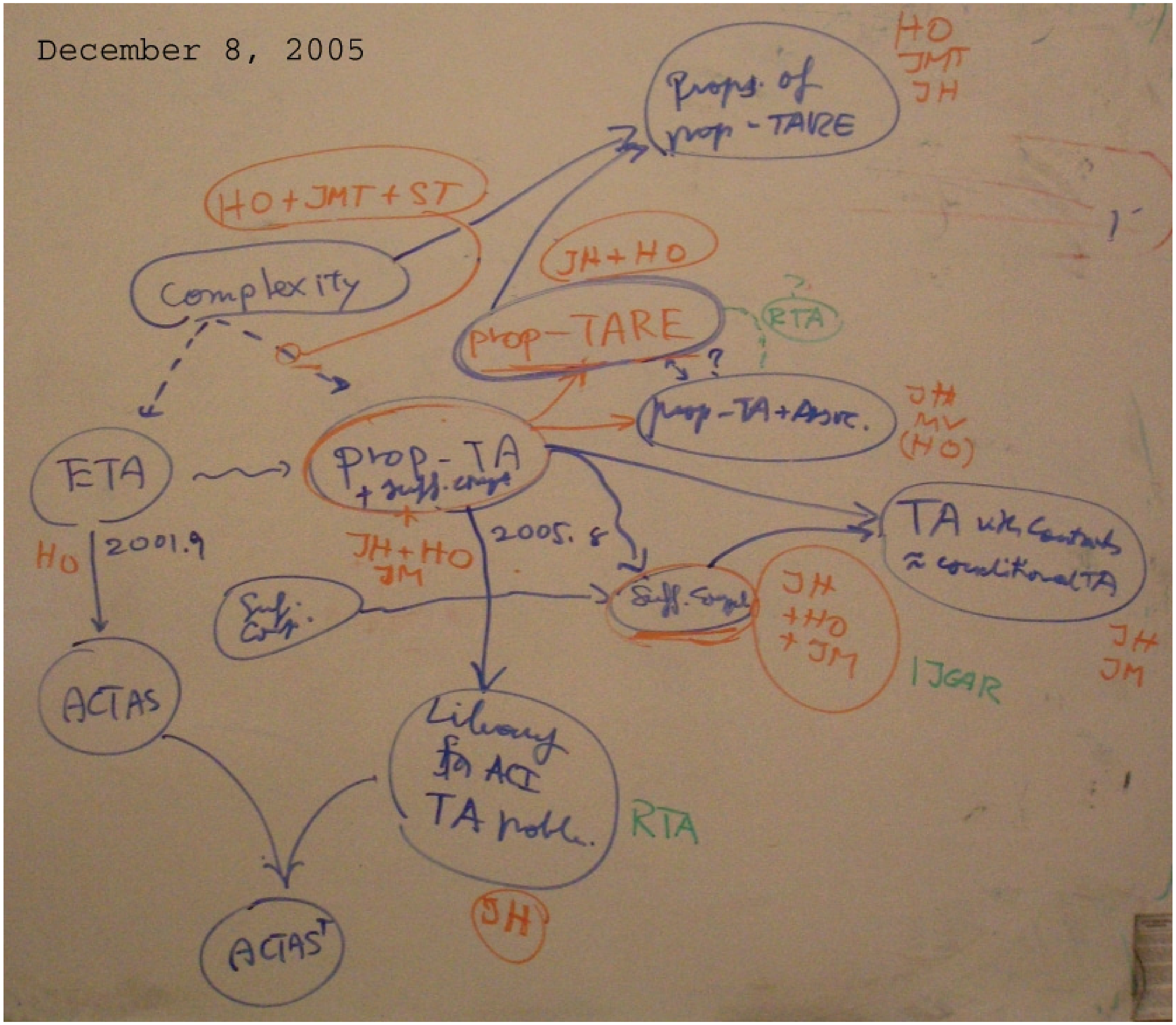
ACUX-tree languages are not closed under complementation [Verma LPAR'03]

Genet & Viet Triem Tong [LPAR'01]

Timbuk : tree automata library

AC-theory is handled by approximation

Roadmap on ACTAS project (2001–)



at University of Illinois at Urbana-Champaign

References

Publications I: equational tree automata (1)

[1] Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories

Hitoshi Ohsaki

15th International Conference of
the European Association for Computer Science Logic (CSL 2001)
Paris (France), September 2001

LNCS 2142, pp. 539–553

[2] Decidability and Closure Properties of Equational Tree Languages

Hitoshi Ohsaki & Toshinori Takai

13th International Conference on
Rewriting Techniques and Applications (RTA 2002)
Copenhagen (Denmark), July 2002

LNCS 2378, pp. 114–128

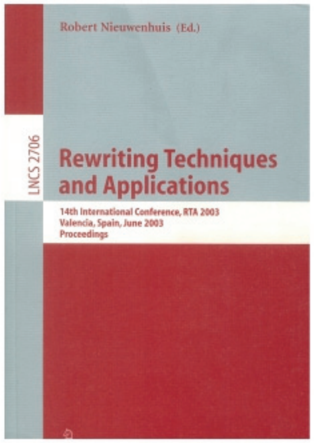
Publications I: equational tree automata (2)

[3] Recognizing Boolean Closed A-Tree Languages with Membership Conditional Rewriting Mechanism

Hitoshi Ohsaki & Hiroyuki Seki & Toshinori Takai

14th International Conference on
Rewriting Techniques and Applications (RTA 2003)
Valencia (Spain), June 2003

LNCS 2706, pp. 483–498



[4] Monotone AC-Tree Automata

Hitoshi Ohsaki & Jean-Marc Talbot & Sophie Tison & Yves Roos
12th International Conference on
Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2005)
Montego Bay (Jamaica), December 2005
LNAI 3855, pp. 337–351

Publications II: software & applications

[5] ACTAS: A System Design for Associative and Commutative Tree Automata Theory

Hitoshi Ohsaki & Toshinori Takai

5th International Workshop on Rule-Based Programming (RULE 2004)
Aachen (Germany), June 2004

ENTCS 124, pp. 97–111

[6] Sufficient Completeness Checking with Propositional Tree Automata

Joe Hendrix & Hitoshi Ohsaki & José Meseguer
technical report August 2005

[7] Propositional Tree Automata

Joe Hendrix & Hitoshi Ohsaki & Mahesh Viswanathan
technical report February 2006

Tool demonstration

[8] ACTAS: Associative and Commutative Tree Automata Simulator

(presented by Toshinori Takai)

4th International Conference on Application of Concurrency to System Design (ACSD 2004), Hamilton (Canada), June 2004

Software products

[9] CETA:
Library for Equational Tree Automata

Joe Hendrix

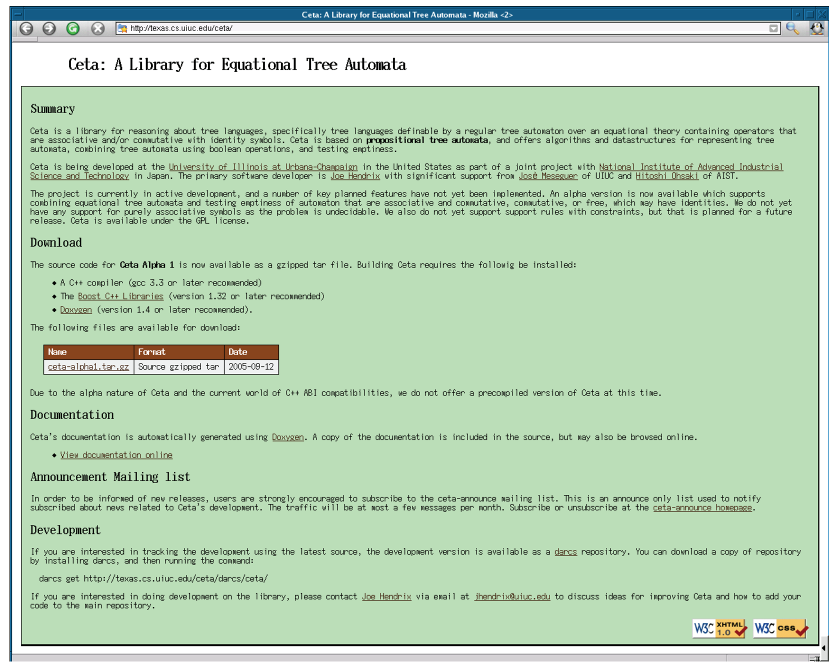
`http://texas.cs.uiuc.edu/ceta/`

[10] ACTAS

Hitoshi Ohsaki

To be announced at

`http://staff.aist.go.jp/hitoshi.ohsaki/actas/`



Ceta: A Library for Equational Tree Automata - Mozilla <2>

http://texas.cs.uiuc.edu/ceta/

Ceta: A Library for Equational Tree Automata

Summary

Ceta is a library for reasoning about tree languages, specifically tree languages definable by a regular tree automaton over an equational theory containing operators that are associative and/or commutative with identity symbols. Ceta is based on **propositional tree automata**, and offers algorithms and datastructures for representing tree automata, combining tree automata using boolean operations, and testing emptiness.

Ceta is being developed at the University of Illinois at Urbana-Champaign in the United States as part of a joint project with National Institute of Advanced Industrial Science and Technology in Japan. The primary software developer is Joe Hendrix with significant support from José Meseguer of UIUC and Hitoshi Ohsaki of AIST.

The project is currently in active development, and a number of key planned features have not yet been implemented. An alpha version is now available which supports combining equational tree automata and testing emptiness of automaton that are associative and commutative, commutative, or free, which may have identities. We do not yet have any support for purely associative symbols as the problem is undecidable. We also do not yet support support rules with constraints, but that is planned for a future release. Ceta is available under the GPL license.

Download

The source code for **Ceta Alpha 1** is now available as a gzipped tar file. Building Ceta requires the followig be installed:

- A C++ compiler (gcc 3.3 or later recommended)
- The Boost C++ Libraries (version 1.32 or later recommended)
- Doxygen (version 1.4 or later recommended).

The following files are available for download:

Name	Format	Date
ceta-alpha1.tar.gz	Source gzipped tar	2005-09-12

Due to the alpha nature of Ceta and the current world of C++ ABI compatibilities, we do not offer a precompiled version of Ceta at this time.

Documentation

Ceta's documentation is automatically generated using Doxygen. A copy of the documentation is included in the source, but may also be browsed online.

- View documentation online

Announcement Mailing list

In order to be informed of new releases, users are strongly encouraged to subscribe to the ceta-announce mailing list. This is an announce only list used to notify subscribed about news related to Ceta's development. The traffic will be at most a few messages per month. Subscribe or unsubscribe at the ceta-announce homepage.

Development

If you are interested in tracking the development using the latest source, the development version is available as a darcs repository. You can download a copy of repository by installing darcs, and then running the command:

```
darcs get http://texas.cs.uiuc.edu/ceta/darcs/ceta/
```

If you are interested in doing development on the library, please contact Joe Hendrix via email at jhendrix@uiuc.edu to discuss ideas for improving Ceta and how to add your code to the main repository.

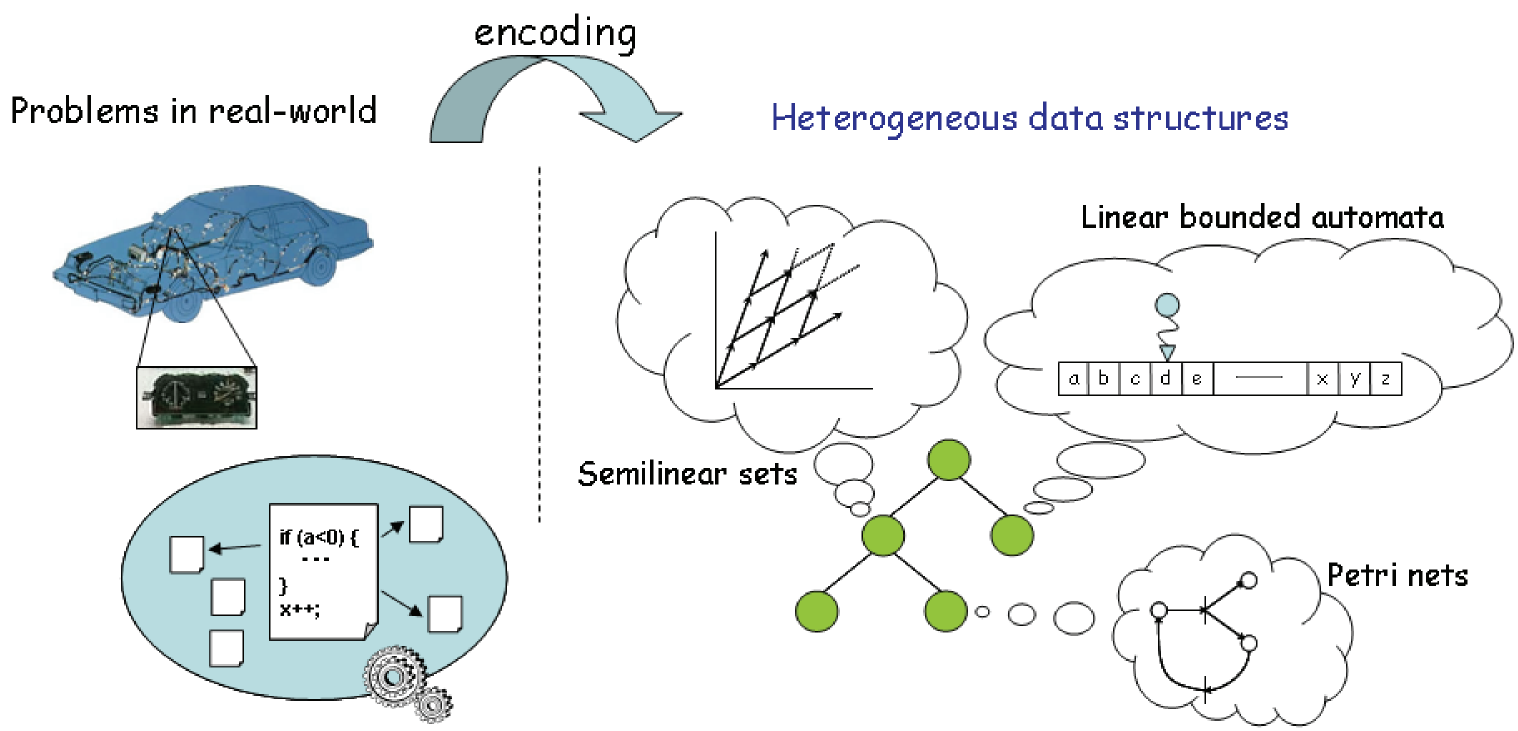
CETA homepage

Part II : System verification and tree automata

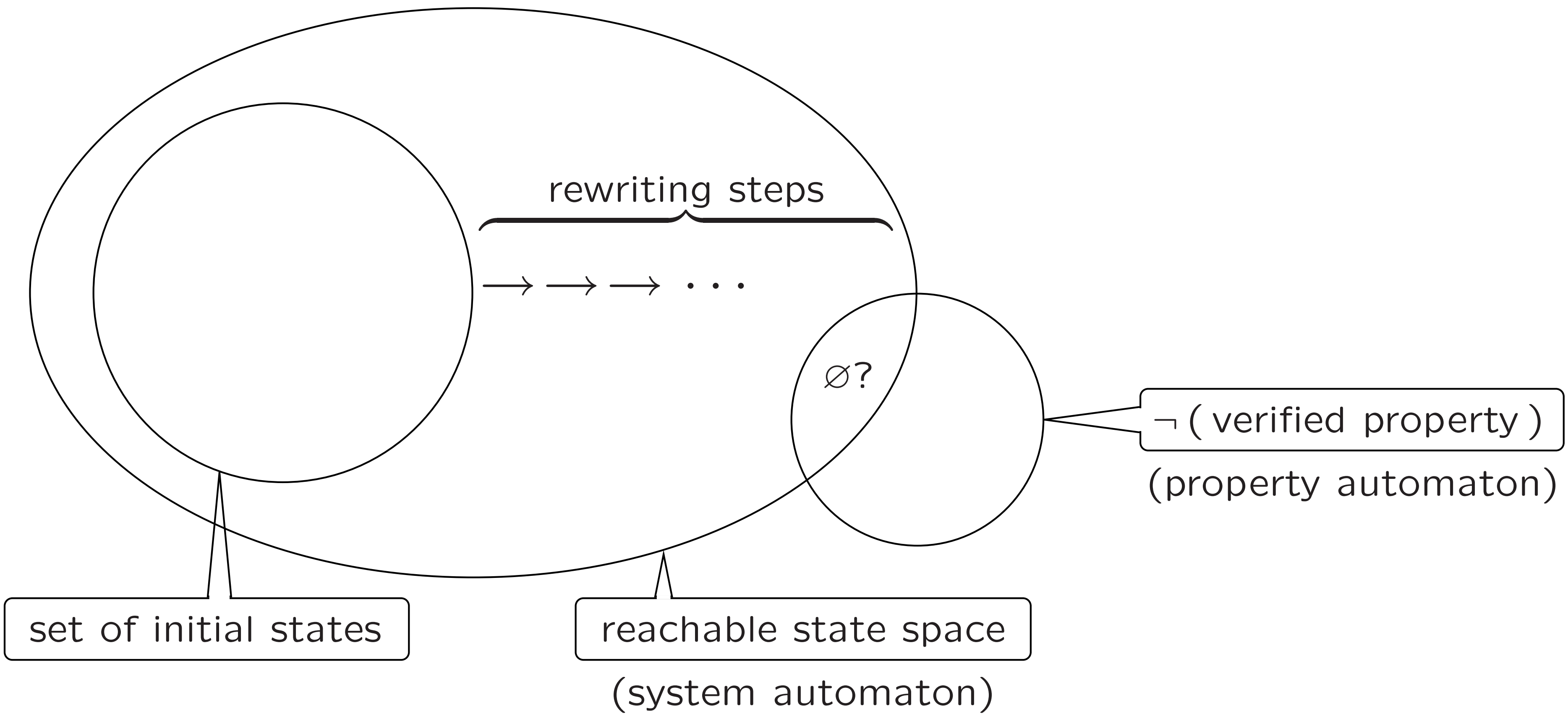
Solving model checking problem in tree automata

Automated reasoning :

- closure properties of Boolean operations
- decidable sub-classes



Reachability analysis based on rewriting and tree automata

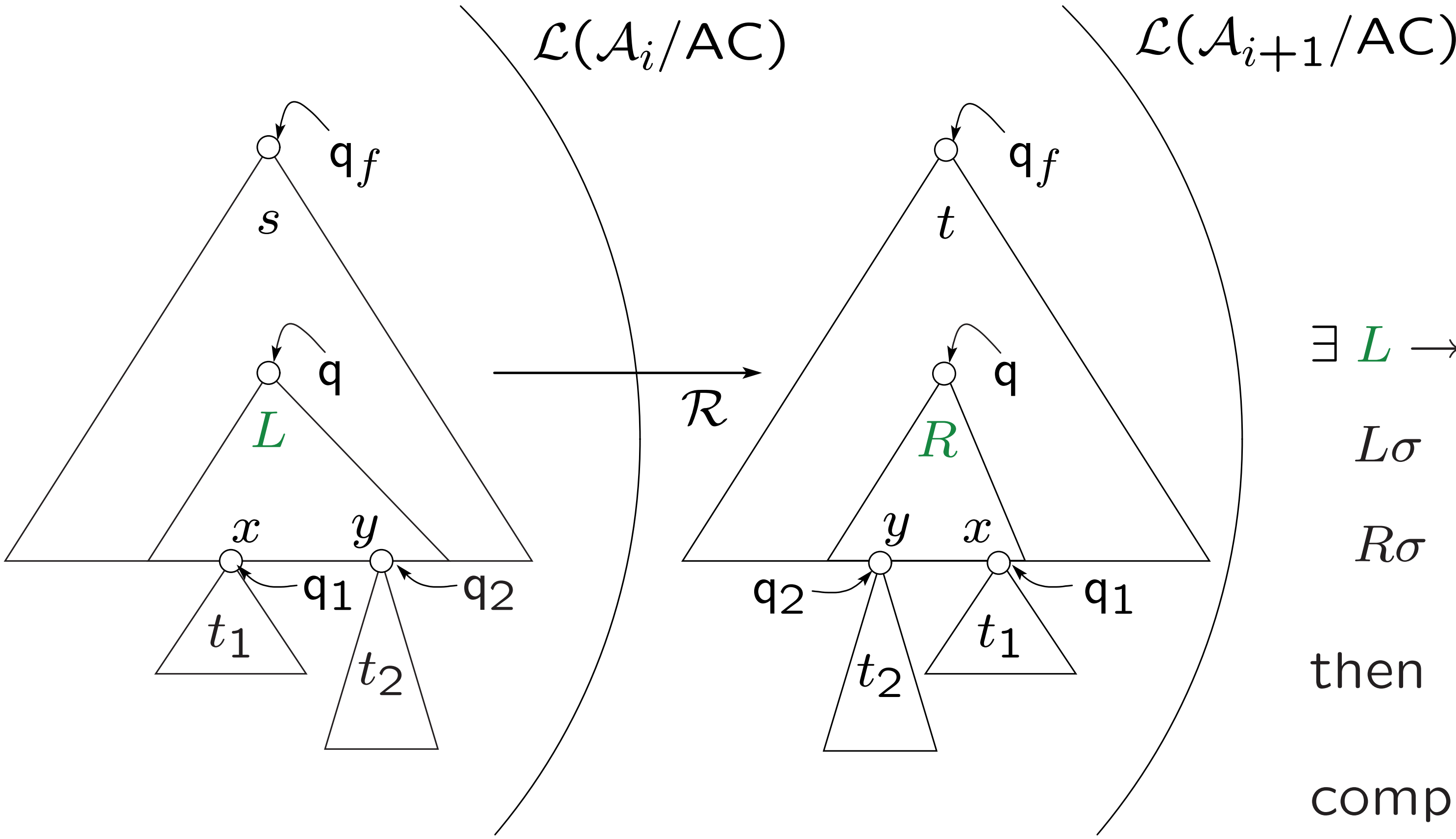


model : term rewriting system + tree automaton

property : tree automaton

verification : Boolean operations & decision problems

One step of the procedure



$\exists\ L \to R$ in $\mathcal{R}$ such that

$L\sigma \ \to^*_{\mathcal{A}_i/\mathsf{AC}}$ q

$R\sigma \ \not\to^*_{\mathcal{A}_i/\mathsf{AC}}$ q

then

complete $\mathcal{A}_i/\mathsf{AC}$ so that

$R\sigma \ \to^*_{\mathcal{A}_{i+1}/\mathsf{AC}}$ q

ACTAS : A tool for equational tree automata computation

- Platform OS:

  Linux

  Solaris

  Windows

- Software requirement:

  Java

  ant (for rebuild)

  libstdc++ (for CETA library)

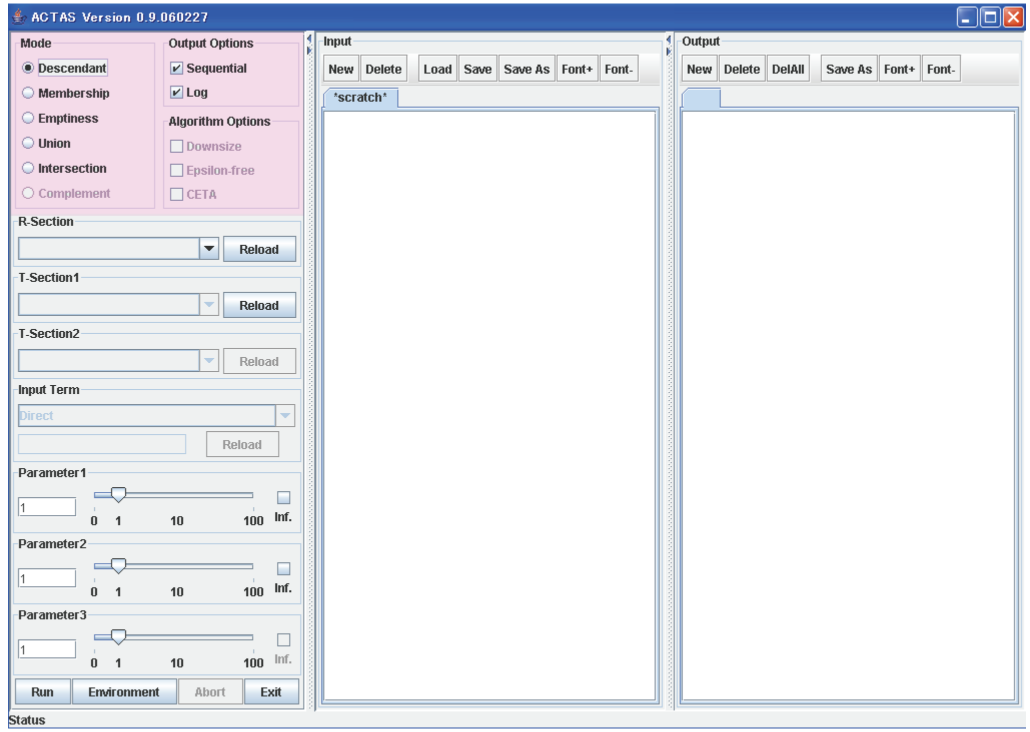
- Memory:

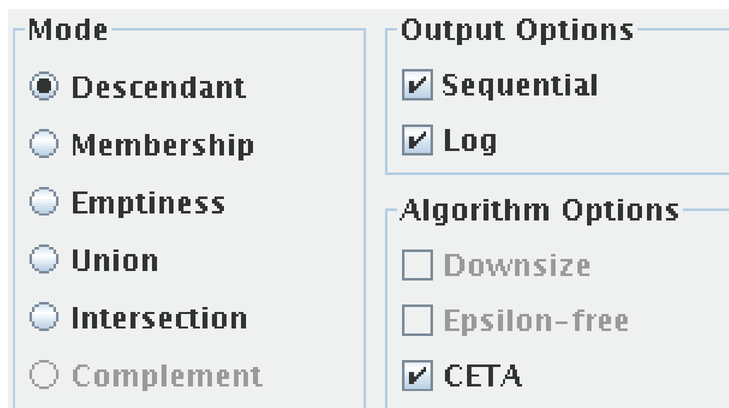
  up to 2G byte (32 bit CPU)

  over 20G byte (64 bit CPU)

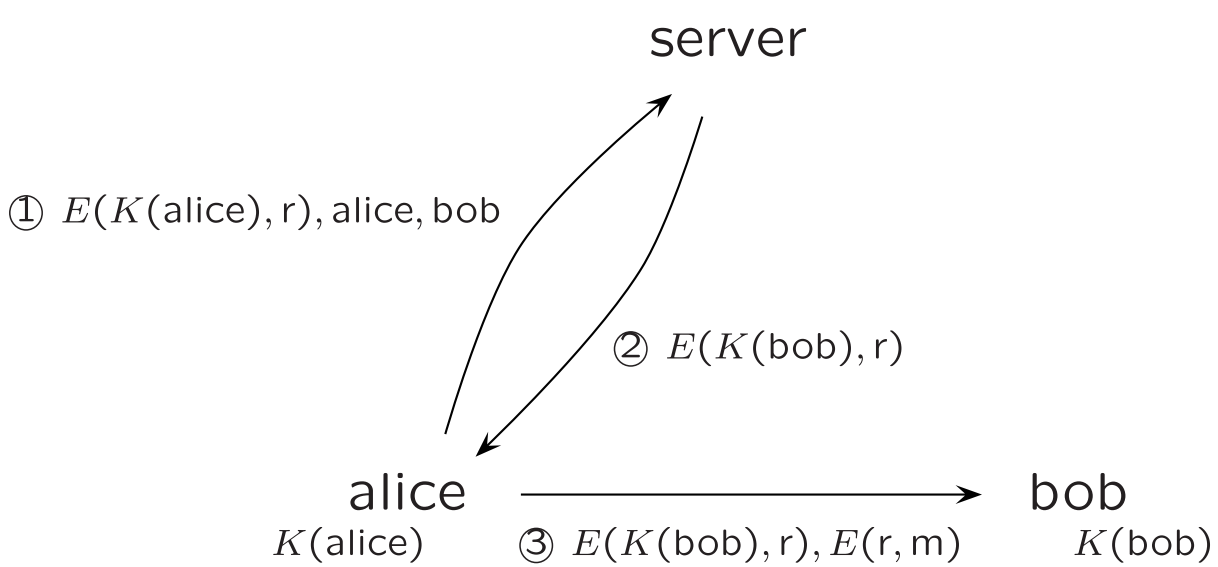
- Version:

  0.9.060227

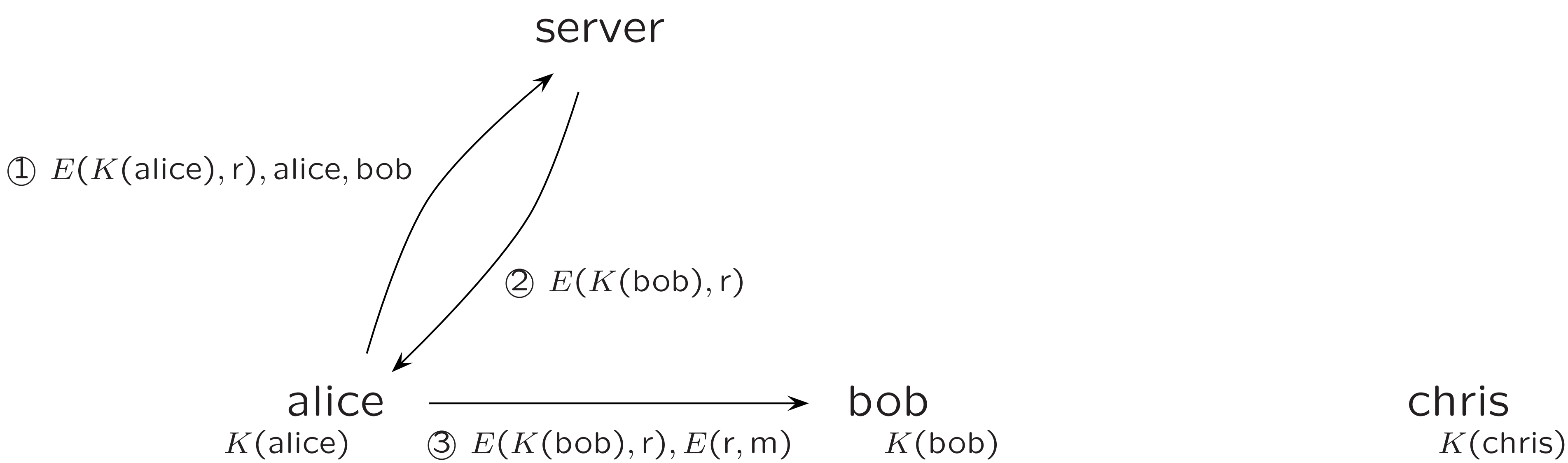




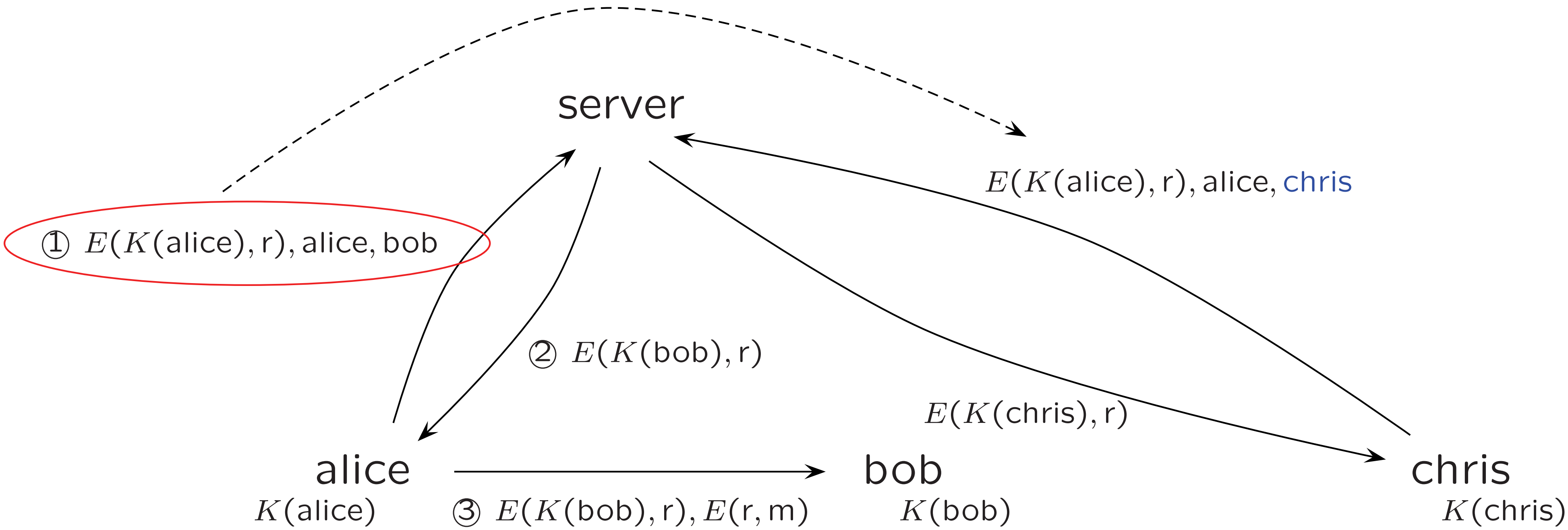
Security flaw in a network protocol (1)



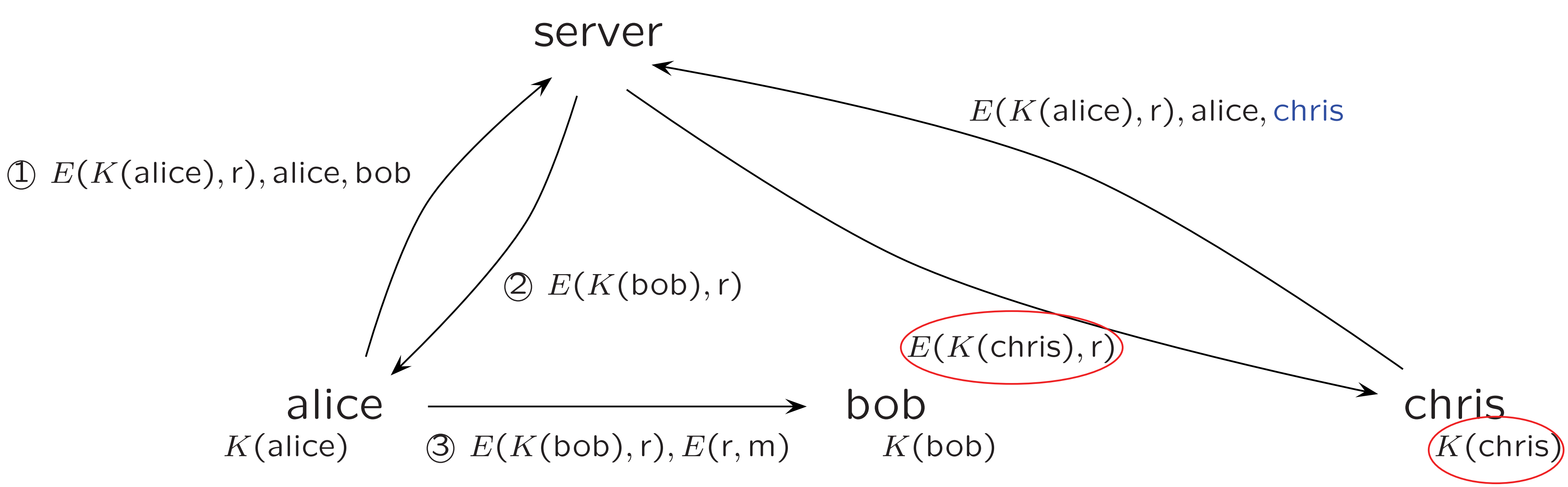
Security flaw in a network protocol (2)



Security flaw in a network protocol (3)



Security flaw in a network protocol (4)

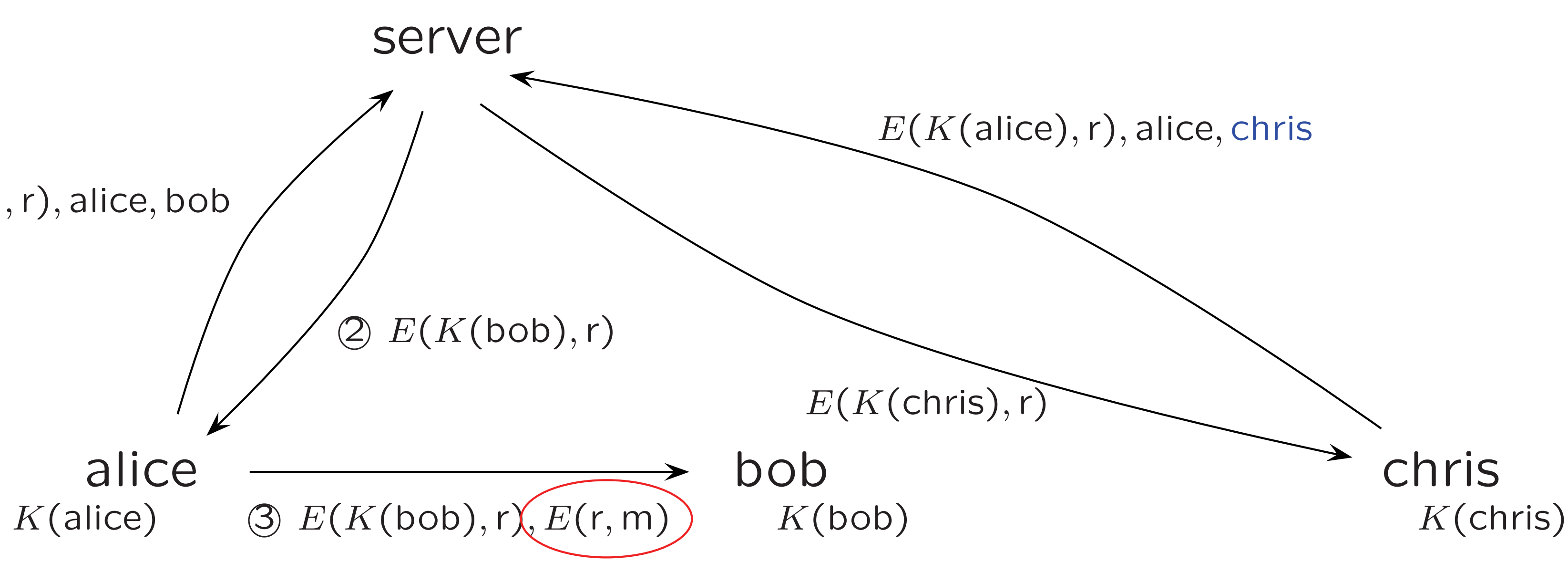


Axiom

$$D(\,x,\ E(\,x,\ y\,)\,)\ \longrightarrow\ y$$

$$D(\ K(\text{chris})\ ,\ E(K(\text{chris}),\text{r})\ )\longrightarrow\ \text{r}$$

Security flaw in a network protocol (5)



Axiom

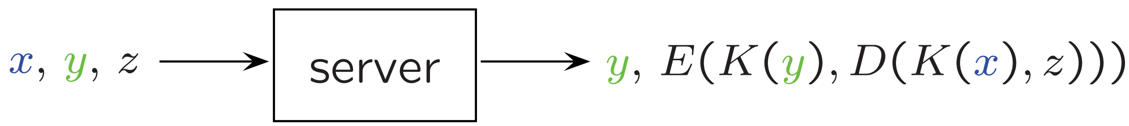
$$D(\, x,\ E(\, x,\ y\,)\,) \longrightarrow y$$

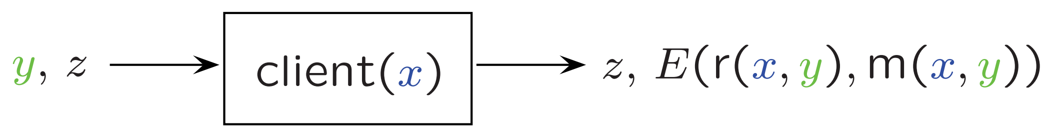
$$D(\ K(\text{chris})\ ,\ E(K(\text{chris}), \text{r})\ ) \longrightarrow \text{r}$$

$$D(\ \text{r}\ ,\ E(\text{r}, \text{m})\ ) \longrightarrow \text{m}$$ (secret message)

ACTAS specification (Lines $1-25$)

```
 1: [Signature]
 2:  const: a,b,c,s
 3:  var: x,y,z
 4:
 5: [R-rule: TRS]
 6:   Ds(x,Es(x,y)) -> y
 7:
 8:   p1(pair(x,y)) -> x
 9:   p2(pair(x,y)) -> y
10:
11: # S1_s(pair(pair(x,y),z)) -> pair(y,Es(k(y),Ds(k(x),z)))
12:   S1_s(pair(pair(a,b),z)) -> pair(b,Es(k(b),Ds(k(a),z)))
13:   S1_s(pair(pair(a,c),z)) -> pair(c,Es(k(c),Ds(k(a),z)))
14:
15: # S2_x(y,z) -> pair(z,Es(nonce(x,y),m(x,y)))
16:   S2_a(pair(b,z)) -> pair(z,Es(nonce(a,b),m(a,b)))
17:
18:   S1_s(x) -> x
19:   S2_a(x) -> x
20:
21: [T-rule( p, p_client ): TA]
22:  Es(p,p) -> p
23:  Ds(p,p) -> p
24:  p1(p) -> p
25:  p2(p) -> p
```



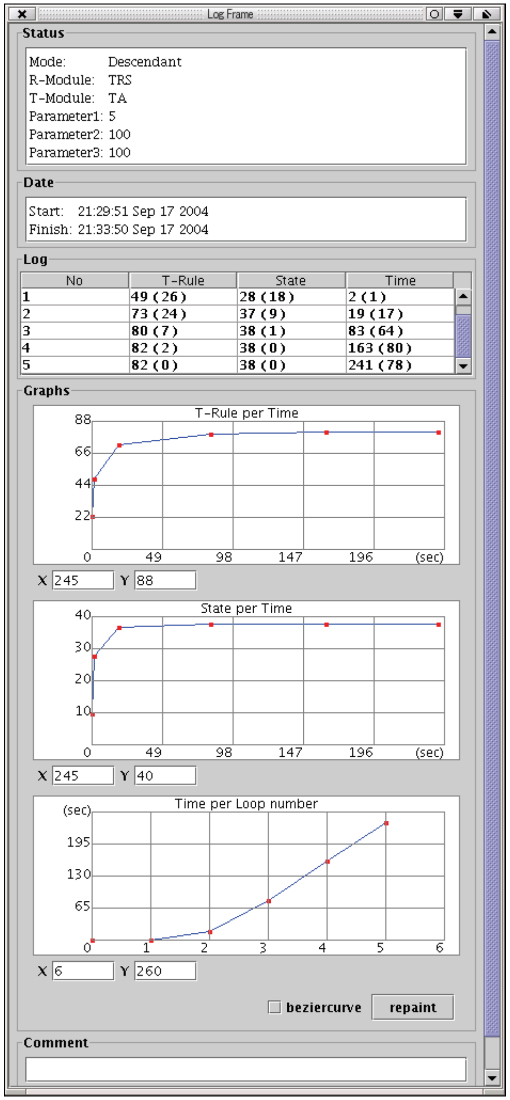


ACTAS specification (Lines 26 –)

```
26:  pair(p,p) -> p
27:  pair(p_client,p_client) -> p
28:  q_a -> p_client
29:  q_b -> p_client
30:  q_c -> p_client
31:
32:  S1_s(p) -> p
33:  S2_a(p) -> p
34:
35: # C's initial knowledge
36:  k(q_c) -> p
37:
38: # initial messsage transfer:
39: # S1_s(pair(pair(a,b),Es(k(a),nonce(a,b)))) -> p
40:
41  # --- subterm decomposition ---
42:  S1_s(q_p_ab_Es_ka_nab) -> p
43:  pair(q_p_ab,q_Es_ka_nab) -> q_p_ab_Es_ka_nab
44:  pair(q_a,q_b) -> q_p_ab
45:  Es(q_ka,q_nab) -> q_Es_ka_nab
46:  k(q_a) -> q_ka
47:  nonce(q_a,q_b) -> q_nab
48:  a -> q_a
49:  b -> q_b
50:  c -> q_c
```

1. If chris knows x and $E(x,y)$, then chris also knows y
2. If chris knows x and y, chris can construct $E(x,y)$ and $D(x,y)$
3. chris knows its own secret key K(chris) and all principals names: alice, bob, chris
4. chris knows message going through the network (wiretapping)
5. chris decomposes sequences of data (modification)
6. chris pretends to be other principals (impersonation)

Descendant computation for reachability analysis



Loop number	♯(T-rules)	♯(states)	time (sec)
0	23	13	3
1	56	34	4
2	102	46	6
3	109	46	18
4	109	46	23

Note 1. $\forall\ i\colon\ \mathcal{L}(\mathcal{A}_i/\mathrm{AC})\ \subseteq\ \mathcal{L}(\mathcal{A}_{i+1}/\mathrm{AC})$

Note 2. $\exists\ i\colon\ \mathcal{L}(\mathcal{A}_i/\mathrm{AC})\ =\ \mathcal{L}(\mathcal{A}_{i+1}/\mathrm{AC})$

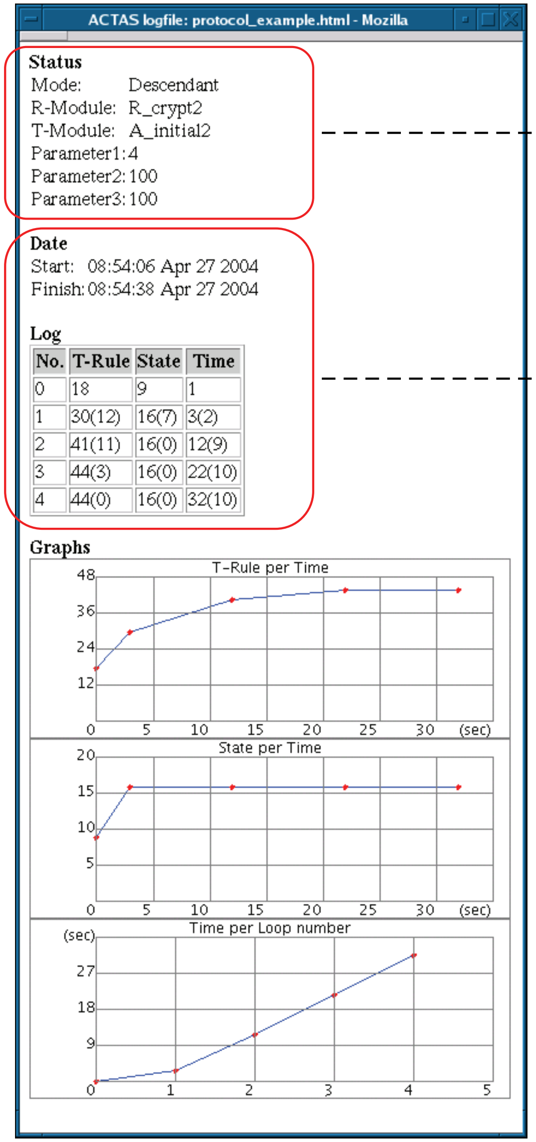
$\Rightarrow \exists\ i\colon\ \mathcal{L}(\mathcal{A}_j/\mathrm{AC})\ =\ \mathcal{L}(\mathcal{A}_{j+1}/\mathrm{AC})$ for all $j \geqslant i$

$(\Rightarrow \exists\ i\colon\ \mathcal{L}(\mathcal{A}_i/\mathrm{AC})\ =\ \mathcal{L}(\mathcal{A}_{\infty}/\mathrm{AC})\,)$

Note 3. $\exists\ i\colon\ \mathtt{m}(\mathtt{a},\mathtt{b})\ \in\ \mathcal{L}(\mathcal{A}_i/\mathrm{AC})$

$\Rightarrow$ secret message m is retrieved by chris

Tool support for state space analysis



Computation mode

Module names (i.e. selected R-rule and T-rule names)

Parameters 1–3 $(0 \leqslant i \leqslant 100)$

Execution time

Number of transition rules

Number of state symbols for each loop computation

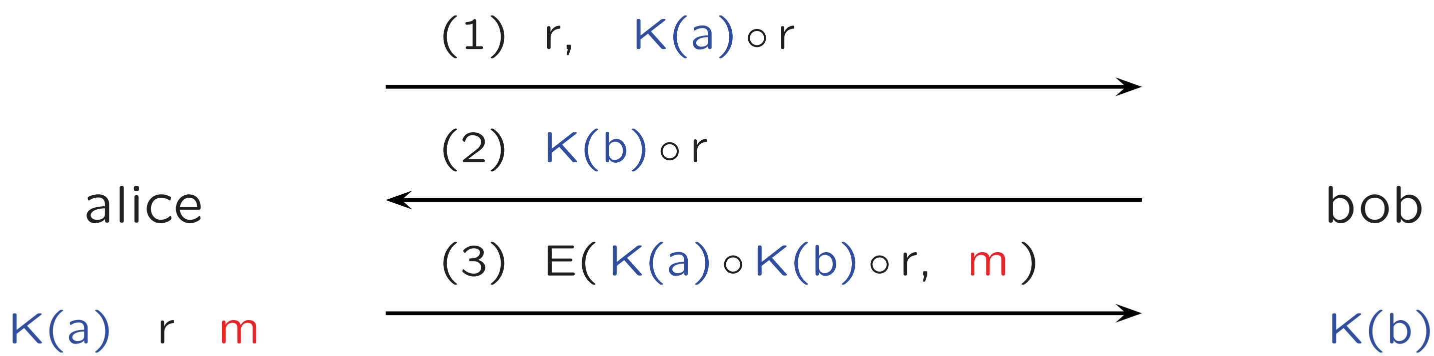
Graph1: number of transition rules $\times$ time(sec)

Graph2: number of state symbols $\times$ time(sec)

Graph3: time(sec) $\times$ loop number

(in HTML file format)

AC-axioms in encryption scheme



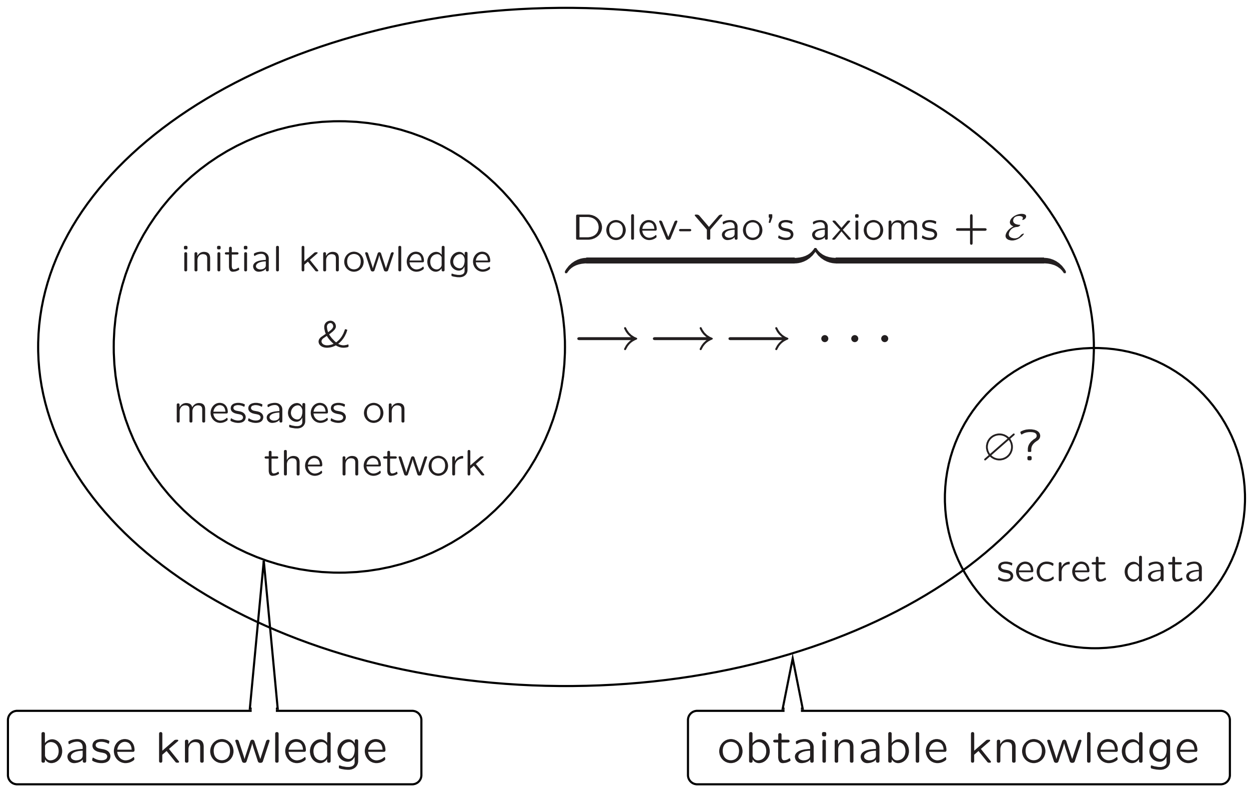
K(a) K(b)	:	secret keys
r	:	random number
m	:	secret message
E	:	encryption function
$\circ$	:	AC symbol (infix operator)

Claim: secret message m is not retrieved by wiretapping only

(Cf. “Easy Intruder Deductions” by Comon-Lundh & Treinen 2003)

AC-function symbols in ACTAS specification

```
[Signature]
 AC: f
 const: a,b,c,m,r
 var: x,y

[R-rule: TRS2]
 Ds(x,Es(x,y)) -> y

[T-rule( p ): TA2]
 Ds(p,p) -> p
 Es(p,p) -> p
 f(p,p) -> p

 f(q_ka,q_r) -> p
 r -> q_r

 r -> p

 f(q_kb,q_r) -> p
 k(q_b) -> q_kb
 b -> q_b

 e(q_f_kba_r,q_m) -> p
 f(q_kab,q_r) -> q_f_kba_r
 f(q_kb,q_ka) -> q_k_ba
 k(q_a) -> q_ka
 a -> q_a
 m -> q_m
# C's initial knowledge
 a -> p
 b -> p
 c -> p
 k(q_c) -> p
 c -> q_c
```



Intruder deduction problem (general version)

Given two sets L, M (of messages) and equational rewrite system $\mathcal{R}/\mathcal{E}$:

Is the intersection of $[\to^*_{\mathcal{R}/\mathcal{E}}](L)$ and M the empty or not ?

Note 1. In the previous setting

L : initial knowledge + messages on the network

M : secret data

$\mathcal{R}/\mathcal{E}$: Dolev-Yao's axioms and AC({f})

Note 2. Tree languages recognized by AC-TA, called *AC-recognizable tree languages* are closed under $\cap$ and

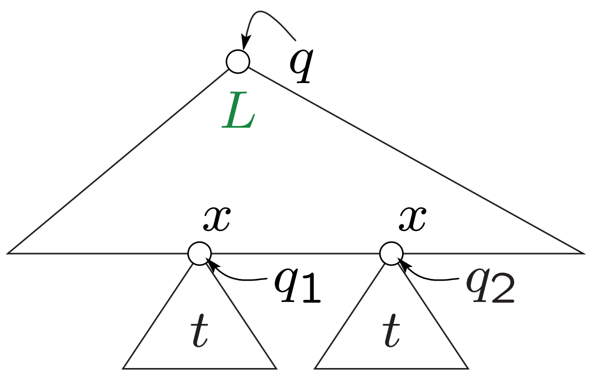
AC-regular tree languages are also closed under $\cap$

Note 3. The emptiness problems for AC-TA and regular AC-TA are decidable

Non-left-linear case

$\forall\ L \rightarrow R$ in $\mathcal{R}$ such that $L = C[\,x,\, x\,]$ e.g. $D(\,x,\ E(x, y)\,) \rightarrow y$

Check $\mathcal{L}(\mathcal{A}_i/\mathsf{AC}, q_1) \cap \mathcal{L}(\mathcal{A}_i/\mathsf{AC}, q_2) \neq \varnothing$



Note

- Using CETA library, the intersection-emptiness problem for regular AC-TA can be handled
- The intersection-emptiness for monotone AC-TA is decidable but the known algorithm solving the problem is extremely expensive!
- In ACTAS, under- (over-)approximation algorithm is applied when solving emptiness problems in AC-case

Research collaborators

Sophie Tison & Jean-Marc Talbot & Yves Roos

Université des Sciences et Technologies de Lille, France

– Invited positions, June 2002 &
June 2005

– Invitation (Talbot) to AIST, April 2006 (planned)

José Meseguer & Joe Hendrix

University of Illinois at Urbana-Champaign, IL, USA

– Invited position, January – March 2004

– Invitation (Hendrix) to AIST, July – August 2005

Ralf Treinen

École Normale Supérieure de Cachan, France

– Invited position, August – September 2004

– Invitation (Treinen) to AIST, December 2001 &
December 2004 &
mid-February – mid-March 2006

Tree automata techniques and applications

Rusinowitch *et al.*

INRIA – AVISPA project

`http://www.avispa-project.org/`

Hosoya & Vouillon & Pierce [ICFP'00]

Murata [PODS'01]

Dal Zilio & Lugiez [RTA'03]

Types in XML , XML manipulation

Yagi & Takata & Seki [ATVA'05]

Querying in Database

Klarlund & Møller & Schwartzbach

BRICS – MONA project

`http://www.brics.dk/mona/`

Ralf Treinen (LSV, ENS de Cachan)

PROUVÉ project

jointly with:

Loria Laboratoire Verimag

Cril Technology France Telecom

(Research Center for Verification and Semantics, AIST)

Office: Amagasaki site – AIST Kansai

Office address:
Nakoji 3–11–46, Amagasaki, Hyogo 661–0974, Japan

URL: **http://staff.aist.go.jp/hitoshi.ohsaki/**

Phone: +81–6–6494–7823
FAX: +81–6–6494–7844