ACTAS

Automated Verification Based on Equational Tree Automata

Hitoshi Ohsaki

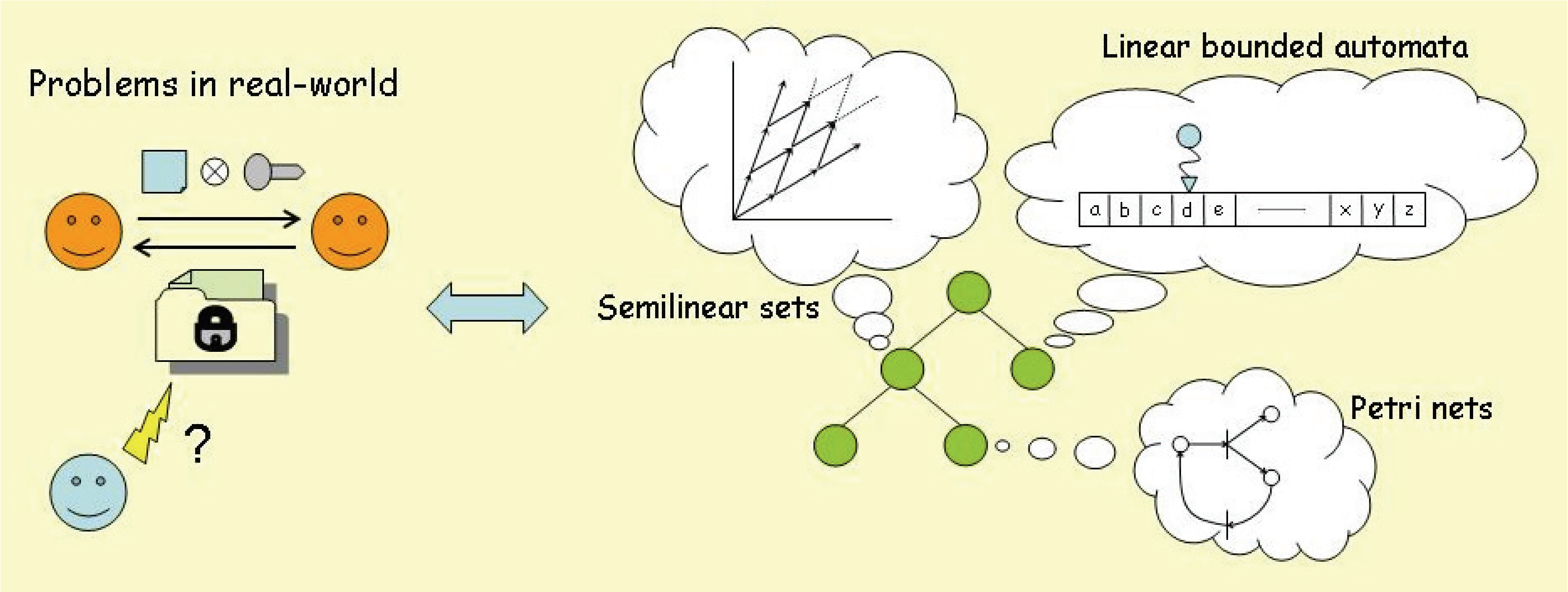
National Institute of
Advanced Industrial Science and Technology (AIST)

JAIST – AIST Workshop, Kanazawa

September 2005

Automated reasoning

- problems with social needs
- decidable fragments of problems (in theory)
- heterogeneous data structures in simple framework



I. Equational tree automata

Automata for strings

Given

a_0	a_1	$\cdots$	a_i	a_{i+1}	$\cdots$	a_n	

and finite automaton $\mathcal{A}$

(q_0)

a_0	a_1	$\cdots$	a_i	a_{i+1}	$\cdots$	a_n	

q_1

a_0	a_1	$\cdots$	a_i	a_{i+1}	$\cdots$	a_n	

1st step: $q_0 \xrightarrow{a_0} q_1$

$\cdots$

q_i

a_0	a_1	$\cdots$	a_i	a_{i+1}	$\cdots$	a_n	

q_{i+1}

a_0	a_1	$\cdots$	a_i	a_{i+1}	$\cdots$	a_n	

i-th step: $q_i \xrightarrow{a_i} q_{i+1}$

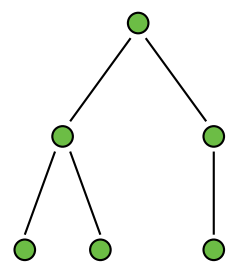
q_{n+1}

Input tape is *accepted* if $\mathcal{A}$ results in

a_0	a_1	$\cdots$	a_i	a_{i+1}	$\cdots$	a_n	

such that q_{n+1} is a final state

Tree automata vs. automata

	tree automata	finite automata
input		
transition rules	$\mathsf{f}(\alpha_1,\ldots,\alpha_n) \longrightarrow \beta$ $\mathsf{f}(\alpha_1,\ldots,\alpha_n) \longrightarrow \mathsf{f}(\beta_1,\ldots,\beta_n)$ $\alpha \longrightarrow \beta$	$\alpha \xrightarrow{\mathsf{a}} \beta$ $\alpha \longrightarrow \beta$
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 $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{fin}, \Delta)$

$\mathcal{F}$	signature, that is set of function symbols with fixed arity
$\mathcal{Q}$	set of state symbols (special constant 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:

$$
\begin{array}{rclr}
f(p_1, \ldots, p_n) & \to & q & \text{(Type 1)} \\
f(p_1, \ldots, p_n) & \to & f(q_1, \ldots, q_n) & \text{(Type 2)} \\
p & \to & q & \text{(Type 3)}
\end{array}
$$

Transition move

- $\to_{\mathcal{A}}$: move relation of tree automaton is defined below

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

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

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

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

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

- $\mathcal{L}(\mathcal{A})$: set of trees t reaching by $\mathcal{A}$ some final state, i.e.

$$t \ \to_{\mathcal{A}} \ t_1 \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} \ t_n \ (\equiv q) \quad \text{for some } \ q \ \text{ in } \ \mathcal{Q}_{fin}$$

- regular tree languages

Basic properties (tree automata)

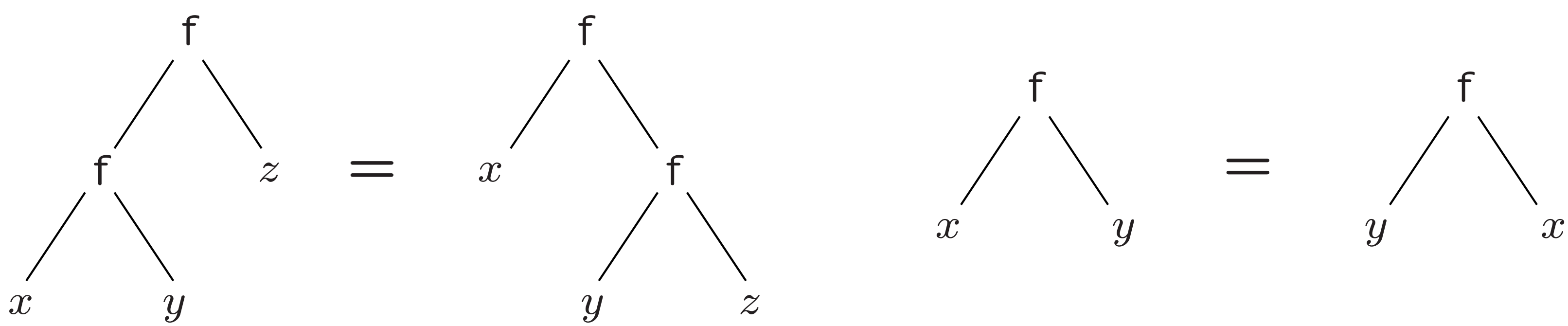
- Union
- Intersection
- Complementation:
  - Deterministic and complete tree automata
- Downsizing technique
  - Epsilon-free tree automata
- Emptiness problem:
  - Pumping lemma

Associativity and commutativity axioms in tree automata

Given a signature $\mathcal{F}$ with $\mathcal{F}_{\mathsf{AC}}$ some of the binary function symbols

AC-tree automaton $\mathcal{A}/\mathsf{AC}$:

a pair of $\mathcal{A}$ and the following AC-axioms for all symbols in $\mathcal{F}_{\mathsf{AC}}$



associativity

commutativity

Likewise

A-tree automaton: tree automaton with associativity axioms of $\mathcal{F}_{\mathsf{A}}$

C-tree automaton: tree automaton with commutativity axioms of $\mathcal{F}_{\mathsf{C}}$

Transition move with associativity and commutativity

- $\rightarrow_{\mathcal{A}/\mathsf{AC}}$: move relation for AC-tree automaton defined as below

$$s \rightarrow_{\mathcal{A}/\mathsf{AC}} t \quad \text{if} \quad s =_{\mathsf{AC}} C[l] \ \text{ and } \ t =_{\mathsf{AC}} C[r]$$
$$\text{for some } \ l \rightarrow r \ \text{ in } \Delta$$

E.g. Consider $\mathcal{A}/\mathsf{AC}$ with $\mathcal{F}_{\mathsf{AC}} = \{ \mathsf{f} \}$ and 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 $\quad \mathsf{f}(\mathsf{b}, \mathsf{a}) \rightarrow_{\mathcal{A}/\mathsf{AC}} \mathsf{f}(q_2, \mathsf{a}) \rightarrow_{\mathcal{A}/\mathsf{AC}} \mathsf{f}(q_2, q_1) \rightarrow_{\mathcal{A}/\mathsf{AC}} q_3$

- $\mathcal{L}(\mathcal{A}/\mathsf{AC})$: set of trees t reaching by $\mathcal{A}/\mathsf{AC}$ some final state, i.e.

$$t \rightarrow_{\mathcal{A}/\mathsf{AC}} t_1 \rightarrow_{\mathcal{A}/\mathsf{AC}} \cdots \rightarrow_{\mathcal{A}/\mathsf{AC}} t_n \ (= q\,) \quad \text{for some } \ q \ \text{ in } \ \mathcal{Q}_{fin}$$

- AC-regular tree languages

Basic properties (AC-tree automata)

- Union
- Intersection
- Complementation:
  - Translation by Parikh theorem
- regular AC-tree automata vs. non-regular AC-tree automata ([6], Ohsaki *et al.* LPAR 2005)
- Emptiness problem:
  - Commutation lemma ([8], Ohsaki CSL 2001)

Example

Consider regular AC-tree automaton with the transition rules

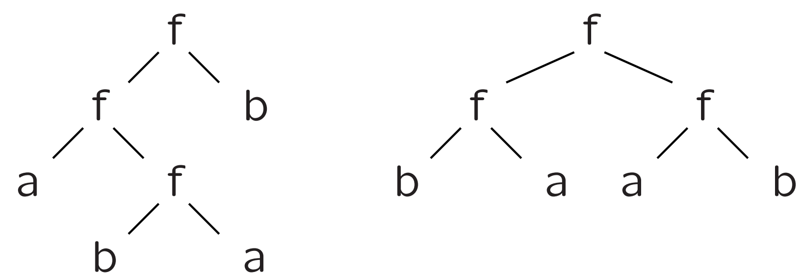
$$\mathsf{a} \to q_{\mathsf{a}} \quad \mathsf{b} \to q_{\mathsf{b}} \quad \mathsf{f}(q_{\mathsf{a}}, q_{\mathsf{b}}) \to q \quad \mathsf{f}(q, q) \to q$$

where $\mathcal{F}_{\mathsf{AC}} = \{\mathsf{f}\}$ and q is final state

Then the accepted tree language L satisfies

$$|t|_{\mathsf{a}} = |t|_{\mathsf{b}}$$

for all t in L



Observations

- The above language L is not accepted by any tree automaton if $\mathcal{F}_{\mathsf{AC}} = \varnothing$
- In general, tree languages L represented by linear (in)equational constraints are accepted by regular AC-tree automata: e.g.

$$|t|_{\mathsf{a}} > |t|_{\mathsf{b}}$$

$$2|t|_{\mathsf{a}} = |t|_{\mathsf{b}}$$

$$|t|_{\mathsf{a}} + 1 = |t|_{\mathsf{b}}$$

Closure under Boolean operations

[Ohsaki CSL'01, Ohsaki & Takai RTA'02
Ohsaki *et al.* RTA'03, Ohsaki *et al.* LPAR'05]

	regular TA	regular AC-TA	AC-TA
closed under $\cup$	$\checkmark$	$\checkmark$	$\checkmark$
closed under $\cap$	$\checkmark$	$\checkmark$	$\checkmark$
closed under $(\ )^c$	$\checkmark$	$\checkmark$	$\times$

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

	regular TA	regular A-TA	A-TA
closed under $\cup$	$\checkmark$	$\checkmark$	$\checkmark$
closed under $\cap$	$\checkmark$	$\times$	$\checkmark$
closed under $(\ )^c$	$\checkmark$	$\times$	$\checkmark$

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

Proof of not closed under complement of AC-TA (outline)

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

P: arithmetic constraint over the natural numbers s.t.

$$
\begin{array}{lcl}
P & := & C \\
 & | & P \vee P \\
 & | & P \wedge P \\
 & | & \neg(\, P\,) \\
C & := & \sharp\mathsf{a}_i \geqslant c \quad (c \in \mathbb{N}) \\
 & | & \sharp\mathsf{a}_i + \sharp\mathsf{a}_j \geqslant \sharp\mathsf{a}_k \\
 & | & \sharp\mathsf{a}_i \times \sharp\mathsf{a}_j \geqslant \sharp\mathsf{a}_k
\end{array}
$$

L_P: tree language over $\mathcal{F}$ s.t. each tree in L_P satisfies P, meaning that

e.g. $L_{\sharp\mathsf{a}=\sharp\mathsf{b}}$ is the tree language whose element t satisfies $|t|_{\mathsf{a}} = |t|_{\mathsf{b}}$

Suppose tree languages recognizable with AC-tree automata are closed under $(\ )^{\mathsf{C}}$

- Given $P \equiv c_1 x_1^{k_1} + \cdots + c_n x_n^{k_n} = c$, L_P is recognizable with AC-tree automata
- $L_P \neq \varnothing$ iff $\exists\,(x_1, \ldots, x_n)$ in $\mathbb{N}^n - \mathbf{0}$: $c_1 x_1^{k_1} + \cdots + c_n x_n^{k_n} = c$

It contradicts to the fact that (weak variant of) Hilbert's 10th problem is undecidable

Decidability results

	regular TA	regular AC-TA	AC-TA
$t \in \mathcal{L}(\mathcal{A}/\mathsf{AC})$?	✓	✓	✓
$\mathcal{L}(\mathcal{A}/\mathsf{AC}) = \emptyset$?	✓	✓	✓
$\mathcal{L}(\mathcal{A}/\mathsf{AC}) \subseteq \mathcal{L}(\mathcal{B}/\mathsf{AC})$?	✓	✓	×
$\mathcal{L}(\mathcal{A}/\mathsf{AC}) = \mathcal{T}(\mathcal{F})$?	✓	✓	?

	regular TA	regular A-TA	A-TA
$t \in \mathcal{L}(\mathcal{A}/\mathsf{A})$?	✓	✓	✓
$\mathcal{L}(\mathcal{A}/\mathsf{A}) = \emptyset$?	✓	✓	×
$\mathcal{L}(\mathcal{A}/\mathsf{A}) \subseteq \mathcal{L}(\mathcal{B}/\mathsf{A})$?	✓	×	×
$\mathcal{L}(\mathcal{A}/\mathsf{A}) = \mathcal{T}(\mathcal{F})$?	✓	×	×

Note 1: One question remains open since CSL 2001

Note 2: The question is registered in the list of RTA Open Questions
(`http://www.lsv.ens-cachan.fr/rtaloop/problems/101.html`)

Yet further tree language hierarchy

$$
\begin{array}{ccccc}
\text{monotone A-TA} & = & \text{monotone A-PTA} & \begin{array}{c}\gtrsim\\ \not\lesssim\end{array} & \text{monotone AC-PTA} \\
 & & \rotatebox{90}{$\supsetneq$} & & \rotatebox{90}{$\supsetneq$} \\
 & & \text{regular A-PTA} & \begin{array}{c}\not\gtrsim\\ \not\lesssim\end{array} & \text{monotone AC-TA} \\
 & & \rotatebox{90}{$\supsetneq$} & \gtrsim,\ \not\lesssim & \rotatebox{90}{$\supsetneq$} \\
 & & \text{regular A-TA} & \begin{array}{c}\not\gtrsim\\ \not\lesssim\end{array} & \text{regular AC-TA} \\
 & & & & \rotatebox{90}{$=$} \\
 & & & & \text{regular AC-PTA}
\end{array}
$$

$\supseteq$: tree language inclusion

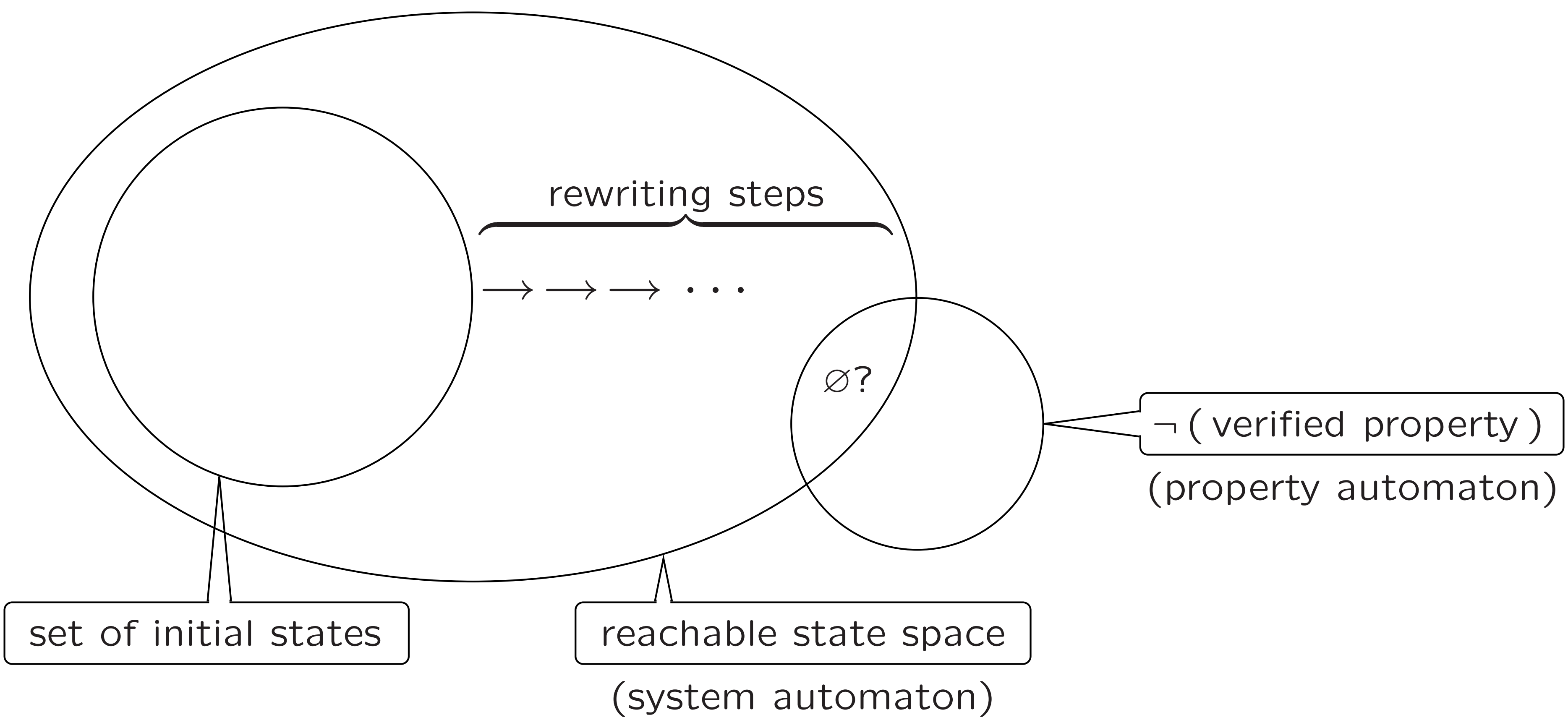
$\gtrsim$: representability relation

II. Towards system verification

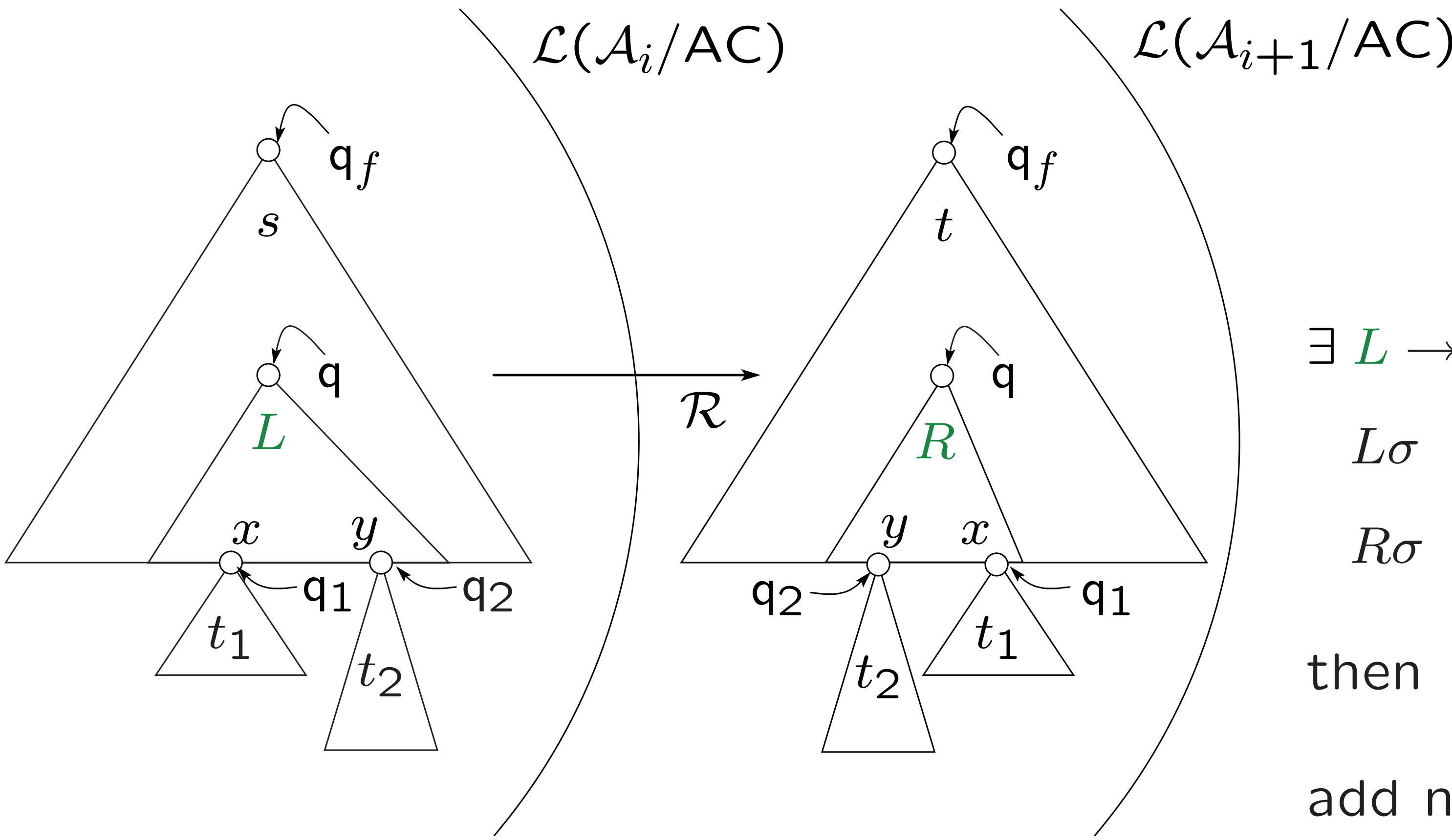
Model checking based on term rewriting and tree automata

<u>Observation1</u> Undecidable fragments exist in the fixpoint computation

<u>Observation2</u> Intersection-emptiness problem is EXPTIME-complete for regular TA



One step of the procedure



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

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

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

then

add new transition rules

to $\mathcal{A}_i/\mathsf{AC}$ to satisfy

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

Specs on ACTAS & user-interface

– Platform OS:

Linux, Solaris

– Software requirement:

Java

ant (for rebuild)

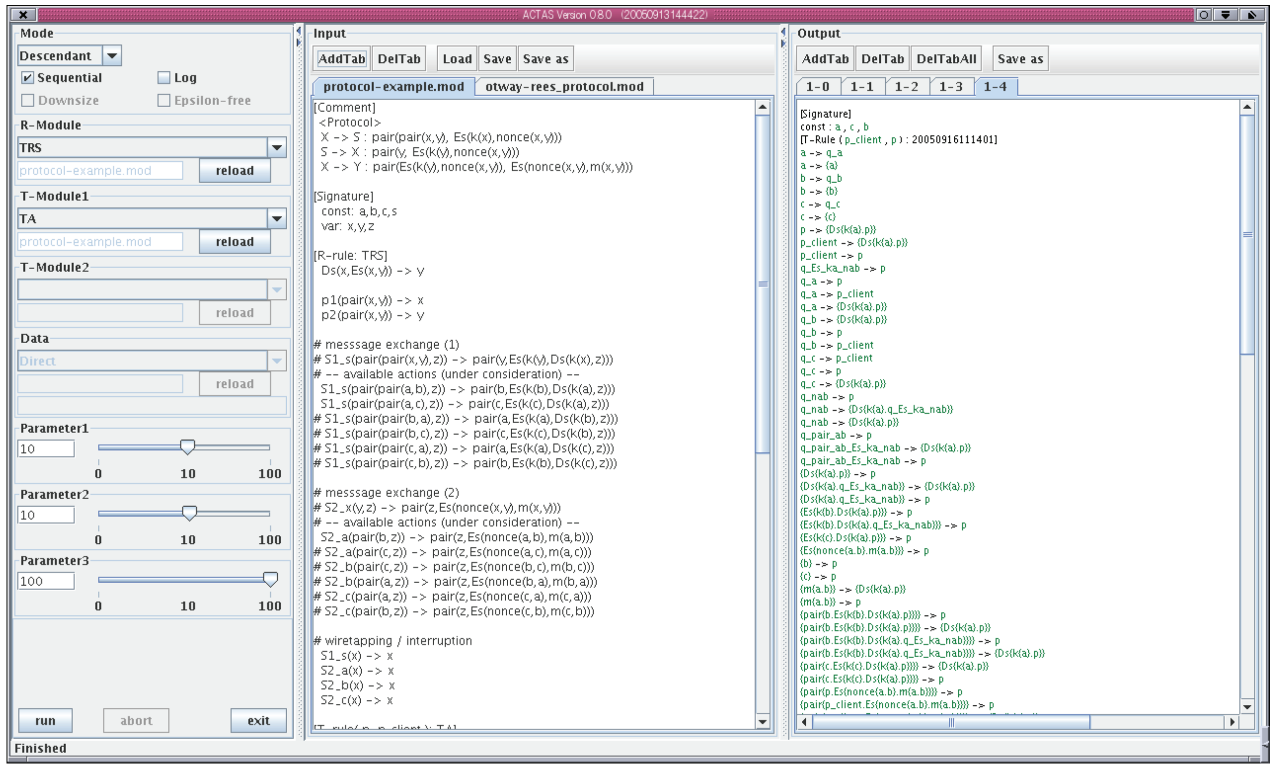
– Memory:

∼ 512M byte (standard model)

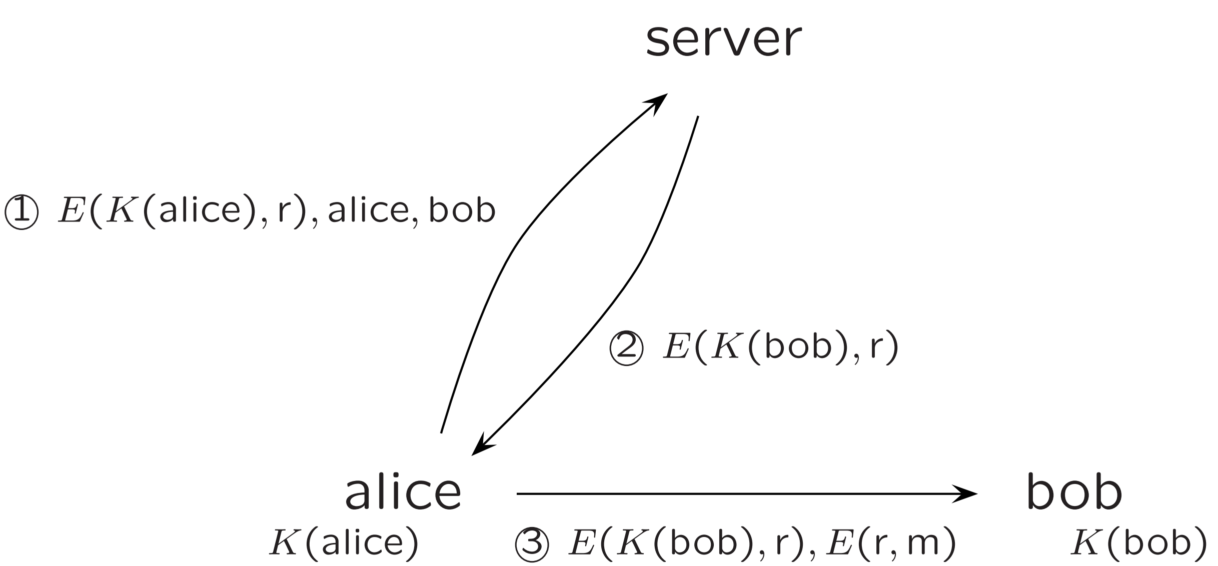
∼ 2G byte (advanced model)

– Version:

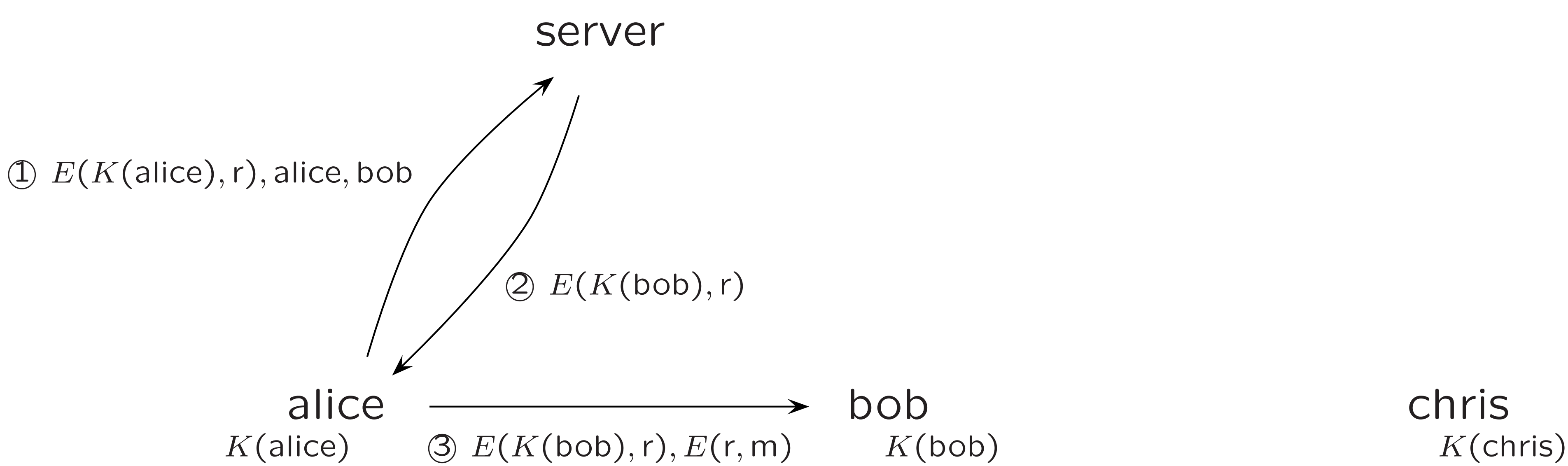
0.8.20050912



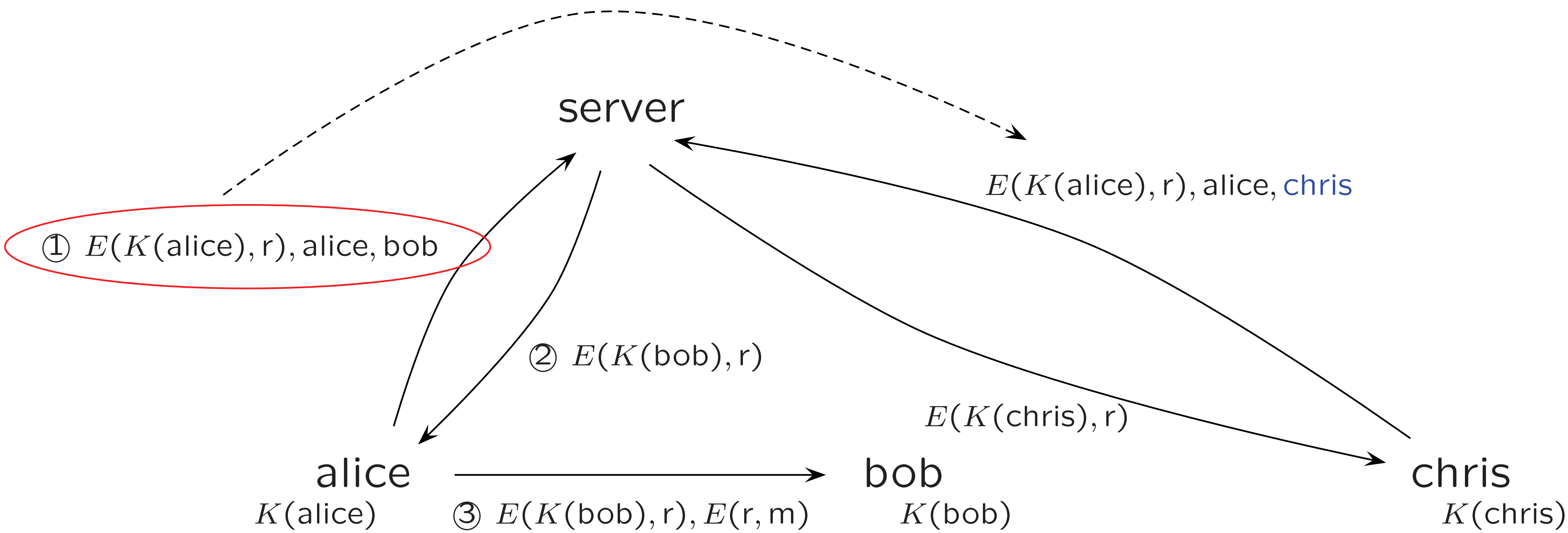
Security flaw in a network protocol (1)



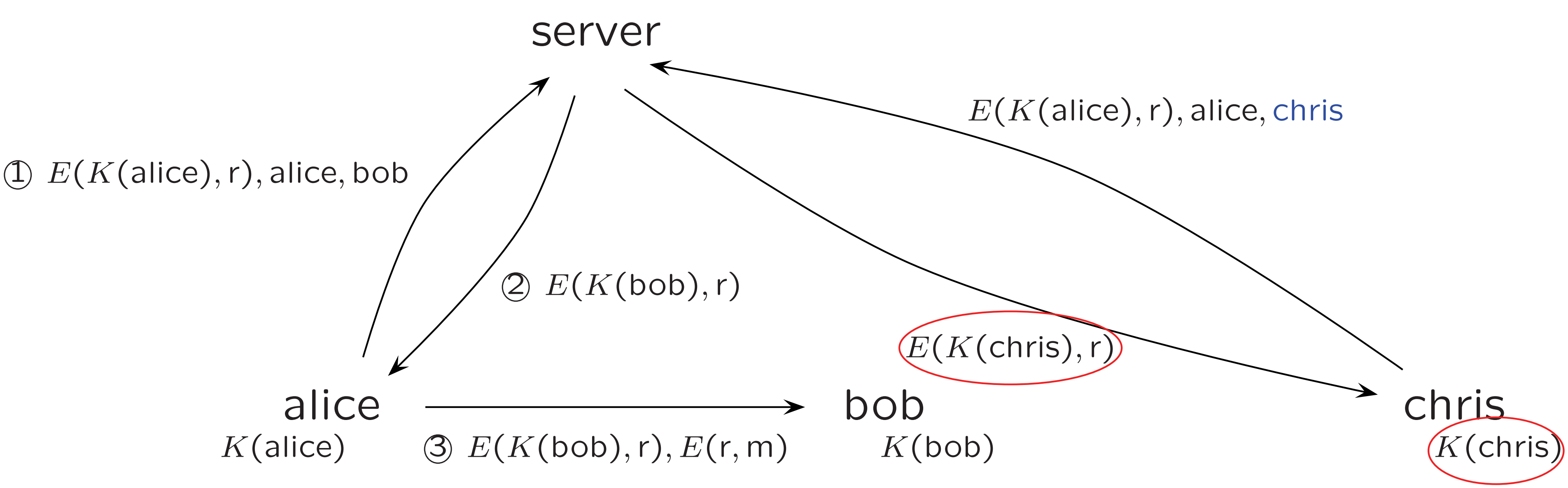
Security flaw in a network protocol (2)



Security flaw in a network protocol (3)



Security flaw in a network protocol (4)

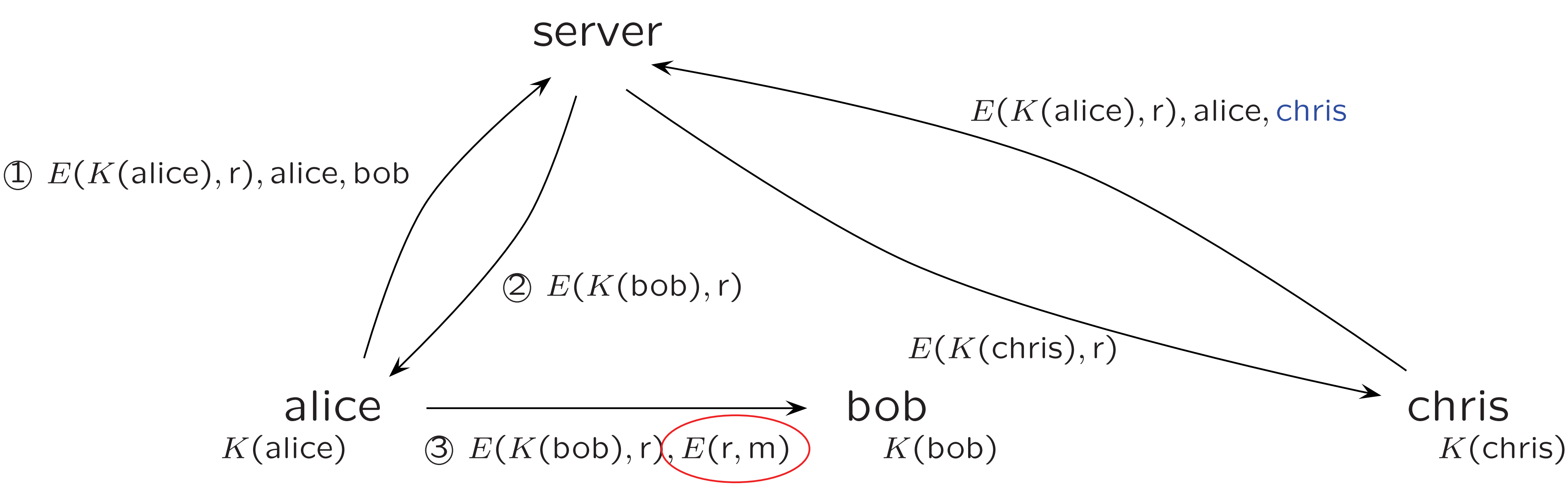


Axiom

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

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

Security flaw in a network protocol (5)



Axiom

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

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

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

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
```

$$x,\ y,\ z \longrightarrow \boxed{\text{server}} \longrightarrow y,\ E(K(y), D(K(x), z)))$$

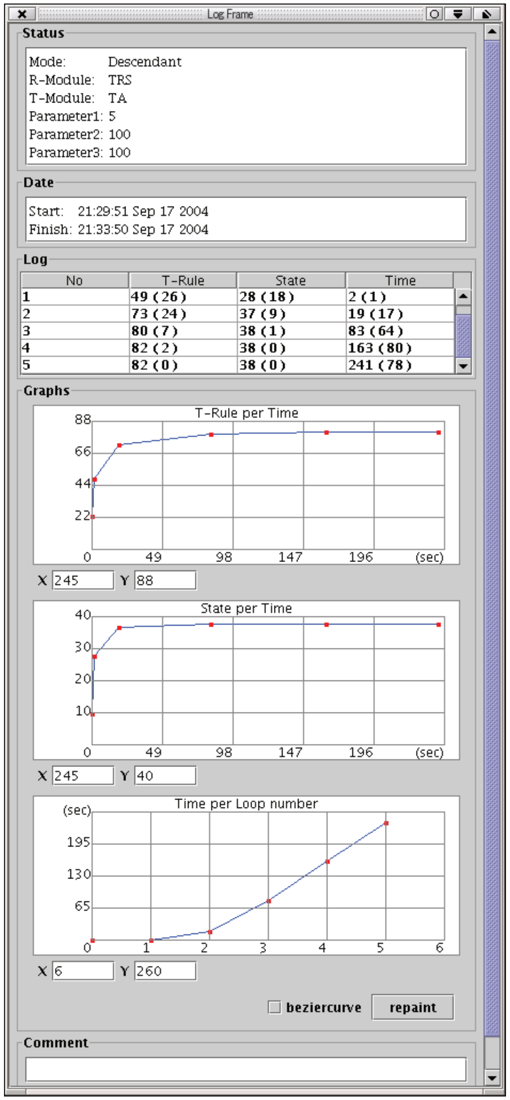
$$y,\ z \longrightarrow \boxed{\text{client}(x)} \longrightarrow z,\ E(\mathsf{r}(x, y), \mathsf{m}(x, y)))$$

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

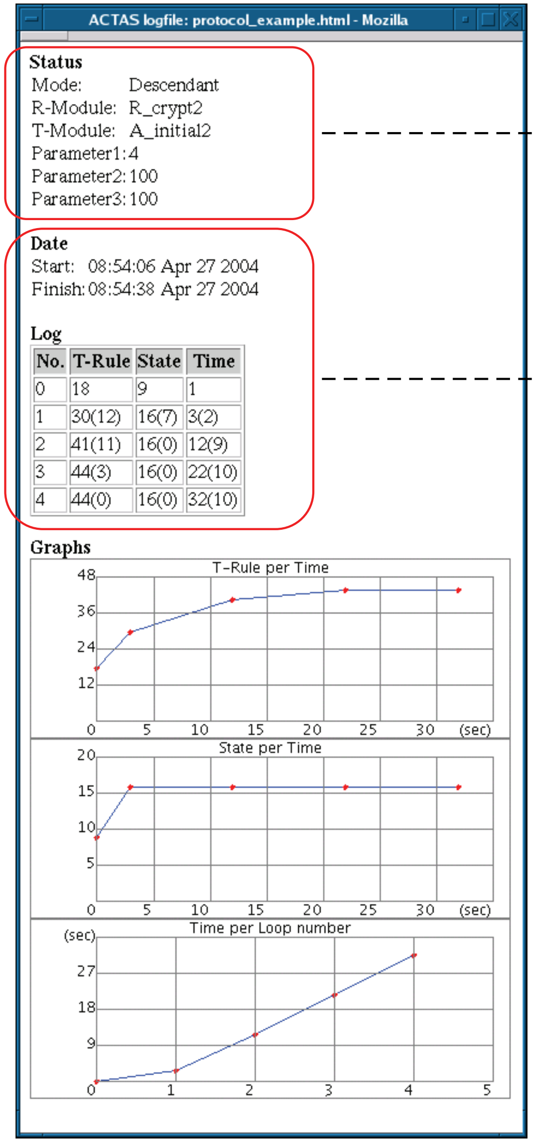
$\Rightarrow \exists\ i\colon\ \mathcal{L}(\mathcal{A}_j/\mathrm{AC}) = \mathcal{L}(\mathcal{A}_{j+1}/\mathrm{AC})$ for all $j \geqslant i$

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

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

alice: $K(a)$ r m

bob: $K(b)$

(1) $r,\ K(a) \circ r$ $\longrightarrow$

(2) $K(b) \circ r$ $\longleftarrow$

(3) $E(K(a) \circ K(b) \circ r,\ m)$ $\longrightarrow$

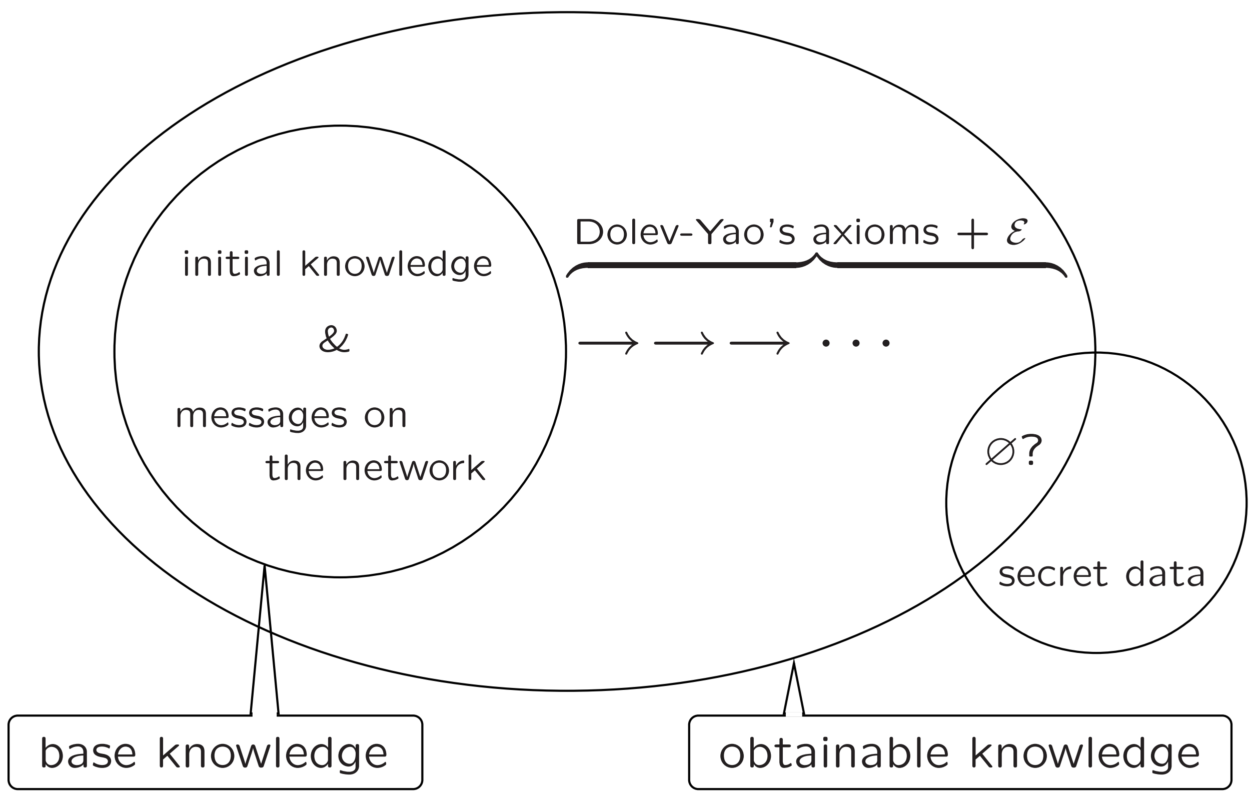
$K(a)$ $K(b)$	:	secret keys
r	:	random number
m	:	secret message
E	:	encryption function
$\circ$	:	AC symbol (infix operator)

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 $\cap$

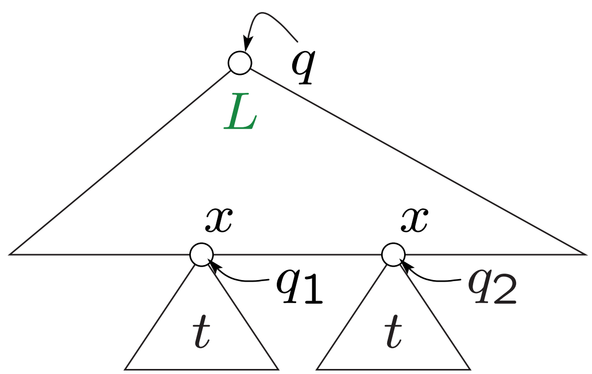
Note 3. The emptiness problems for AC-TA and regular AC-TA are decidable

Non-left-linear case

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

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

  If the above condition holds for $q_1 \neq q_2$, we take the following assignment:

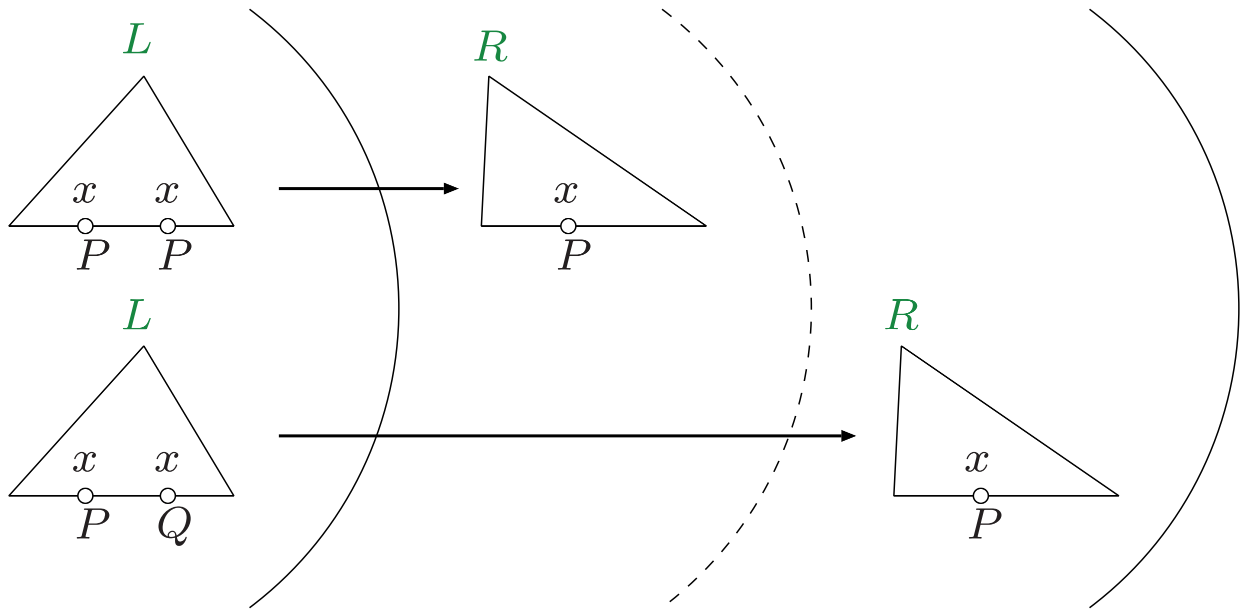


- In fully automated verification, under- or over-approximation is needed to be applied

  The intersection-emptiness for AC-TA is decidable but the complexity is extremely high!

- In fact, we use in ACTAS under- (or over-)approximation algorithms when solving emptiness problems in AC-case

Bounded computation for emptiness test



$\sharp$(T-rule) $= 21$

$\sharp$(state symbols) $= 16$

P2 = 10		
P3	state space for emptiness test	time (sec)
1	0	1
2	256	2
3	4096	2
4	65536	3
5	1048576	9
6	16777216	–

search depth for emptiness test is controlled by Parameters 2,3

References

Research collaboration

[1] Sophie Tison & Jean-Marc Talbot

Université des Sciences et Technologies de Lille, France

– Invited position, June 2002

– Invited position, June 2005

[2] José Meseguer & Joe Hendrix

University of Illinois at Urbana-Champaign, IL, USA

– Invited position, January – March 2004

– Invitation (Hendrix), July – August 2005

[3] Ralf Treinen

École Normale Supérieure de Cachan, France

– Invited position, August – September 2004

– Invitation (Treinen), December 2001 & December 2005

Publications I: software & applications

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

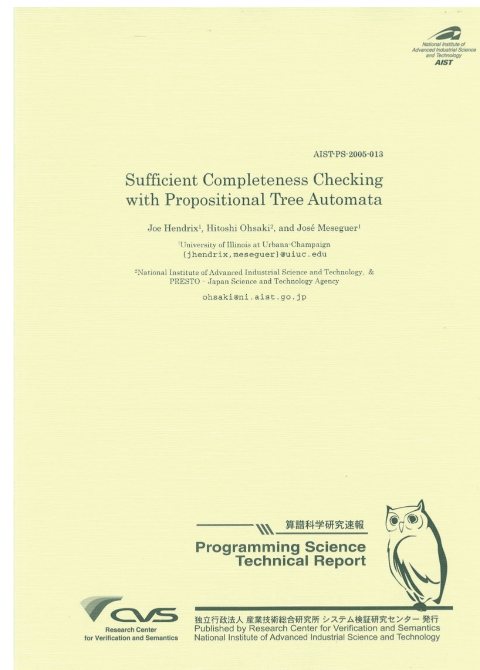
Hitoshi Ohsaki & Toshinori Takai

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

ENTCS 124, pp. 97–111

[5] Sufficient Completeness Checking with Propositional Tree Automata

Joe Hendrix & Hitoshi Ohsaki & José Meseguer
Technical Report AIST-PS-2005-013
AIST/CVS, 2005 (new !)


National Institute of Advanced Industrial Science and Technology
AIST

AIST-PS-2005-013

Sufficient Completeness Checking
with Propositional Tree Automata

Joe Hendrix[1], Hitoshi Ohsaki[2], and José Meseguer[1]

[1]University of Illinois at Urbana-Champaign
{jhendrix,meseguer}@uiuc.edu

[2]National Institute of Advanced Industrial Science and Technology, &
PRESTO - Japan Science and Technology Agency

ohsaki@ni.aist.go.jp

算譜科学研究速報
Programming Science
Technical Report

CVS
Research Center
for Verification and Semantics

独立行政法人 産業技術総合研究所 システム検証研究センター 発行
Published by Research Center for Verification and Semantics
National Institute of Advanced Industrial Science and Technology

Publications II: equational tree automata

[6] AC-Monotone Tree Languages

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

To appear in LNCS

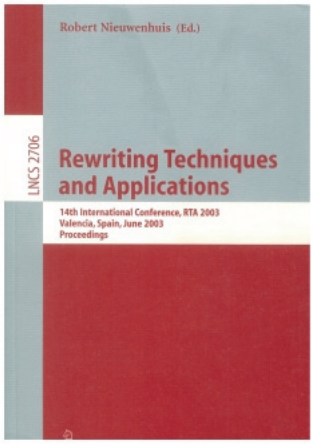
[7] Recognizing Boolean Closed A-Tree Languages with Membership Conditional Rewriting Mechanism

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



Publications II (cont'd)

[7] 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

[8] 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

Tool demonstration

[9] 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

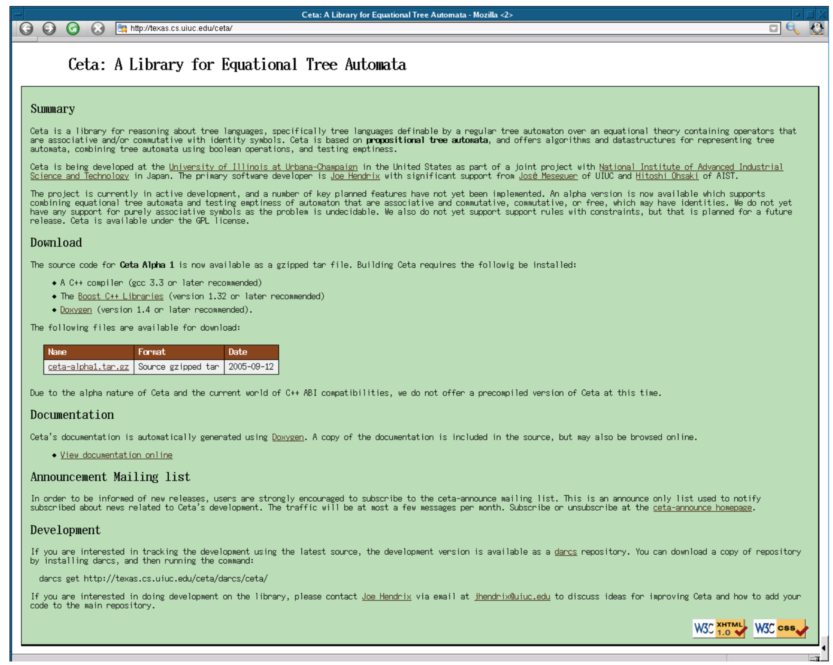
[10] CETA:
Library for Equational Tree Automata

Joe Hendrix

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

[11] ACTAS

Hitoshi Ohsaki
(to be announced)



CETA homepage

Research Center for Verification and Semantics, AIST

Office address: Amagasaki site – AIST Kansai
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