


Title	ACTAS : Automated Verification Based on Equational Tree Automata
Author(s)	Ohsaki, Hitoshi
Citation	
Issue Date	2005-09-21
Type	Presentation
Text version	publisher
URL	http://hdl.handle.net/10119/8327
Rights	
Description	1st VERITE : JAIST/TRUST-AIST/CVS joint workshop on VERification TEchnologyでの発表資料, 開催 : 2005年9月21日 ~ 22日, 開催場所 : 金沢市文化ホール 3F

ACTAS

Automated Verification Based on Equational Tree Automata

Hitoshi Ohsaki

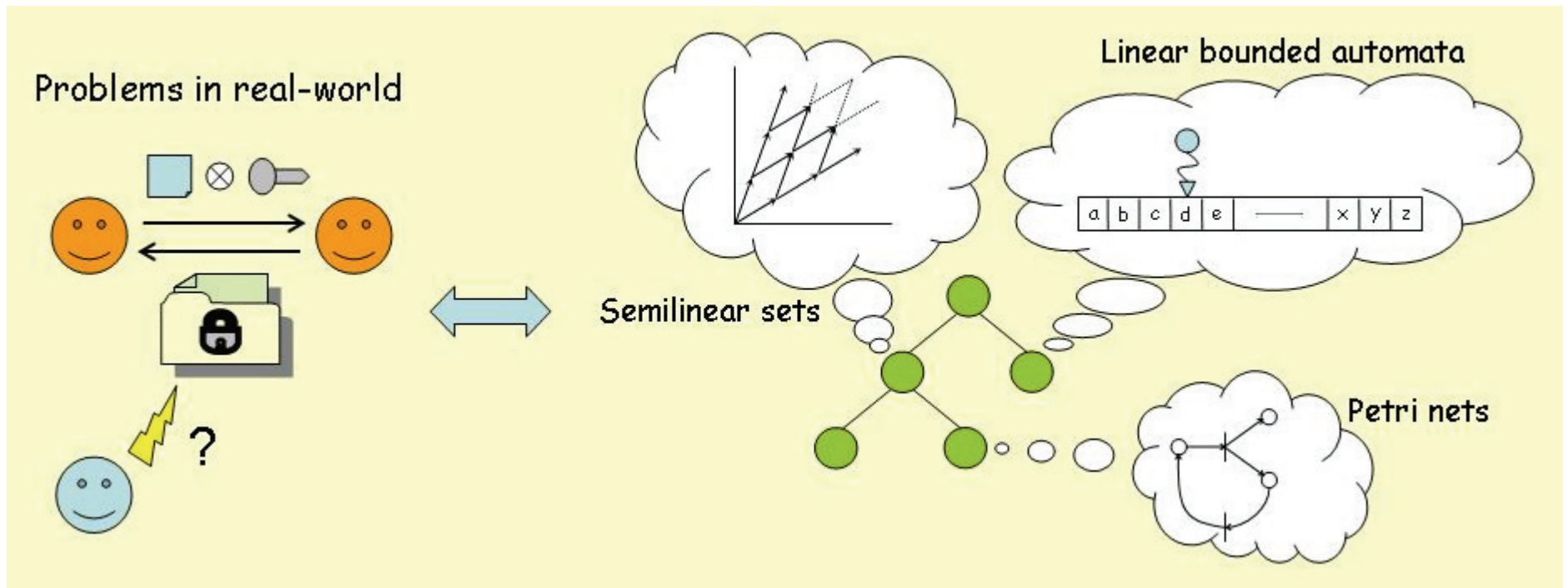
 National Institute of
Advanced Industrial Science and Technology (AIST)

JAIST – AIST Workshop, Kanazawa

September 2005

Automated reasoning

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



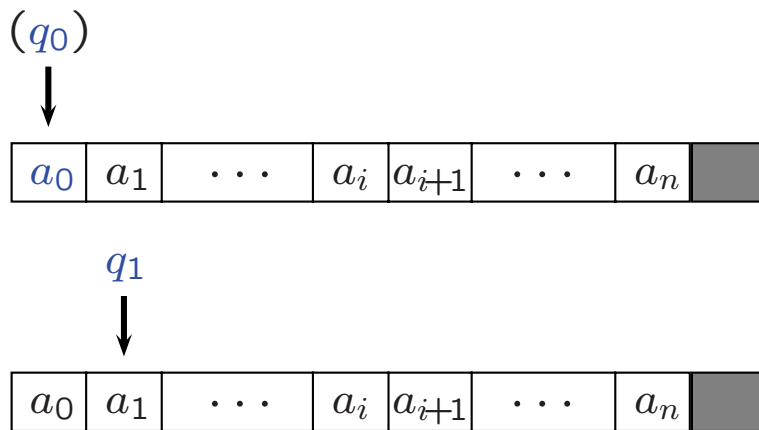
I. Equational tree automata

Automata for strings

Given

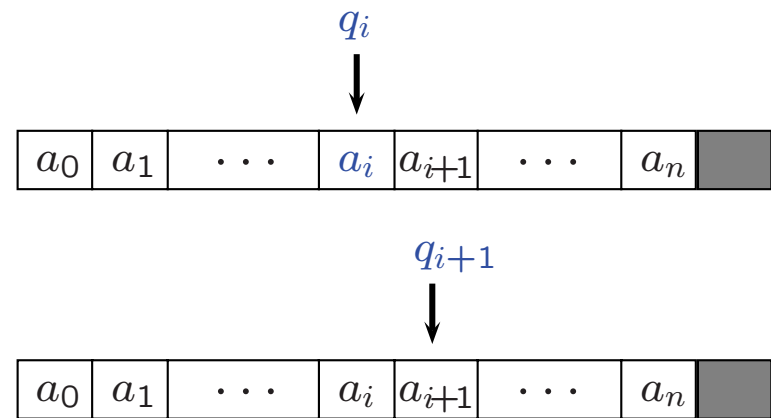
a_0	a_1	\dots	a_i	a_{i+1}	\dots	a_n	
-------	-------	---------	-------	-----------	---------	-------	--

and finite automaton \mathcal{A}



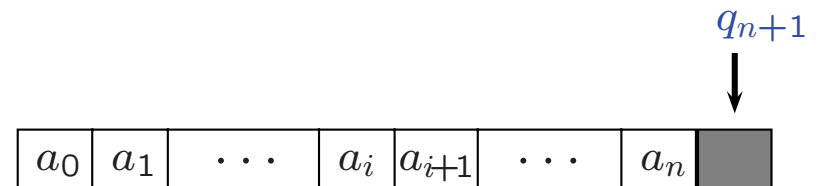
1st step: $q_0 \xrightarrow{a_0} q_1$

...



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

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



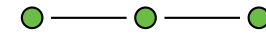
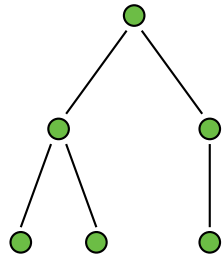
such that q_{n+1} is a final state

Tree automata vs. automata

tree automata

finite automata

input



transition rules

$$f(\alpha_1, \dots, \alpha_n) \rightarrow \beta$$

$$f(\alpha_1, \dots, \alpha_n) \rightarrow f(\beta_1, \dots, \beta_n)$$

$$\alpha \rightarrow \beta$$

$$\alpha \xrightarrow{a} \beta$$

$$\alpha \rightarrow \beta$$

closure properties

$$\cup \quad \cap \quad ()^c$$

$$\cup \quad \cap \quad ()^c$$

decidability

$$\in \quad \subseteq \quad = \emptyset?$$

$$\in \quad \subseteq \quad = \emptyset?$$

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
 $\mathcal{F} \cap \mathcal{Q} = \emptyset$

\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:

$$f(p_1, \dots, p_n) \rightarrow q \quad (\text{Type 1})$$

$$f(p_1, \dots, p_n) \rightarrow f(q_1, \dots, q_n) \quad (\text{Type 2})$$

$$p \rightarrow q \quad (\text{Type 3})$$

Transition move

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

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

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

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

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

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

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

$$t \rightarrow_{\mathcal{A}} t_1 \rightarrow_{\mathcal{A}} \cdots \rightarrow_{\mathcal{A}} t_n (\equiv q) \text{ for some } q \text{ in } Q_{fin}$$

- regular tree languages

Basic properties (tree automata)

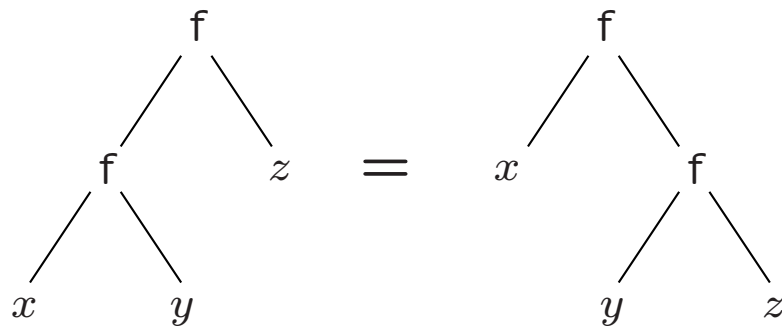
- Union
- Intersection
- Complementation:
 - Deterministic and complete tree automata
- Downsizing technique
 - Epsilon-free tree automata
- Emptiness problem:
 - Pumping lemma

Associativity and commutativity axioms in tree automata

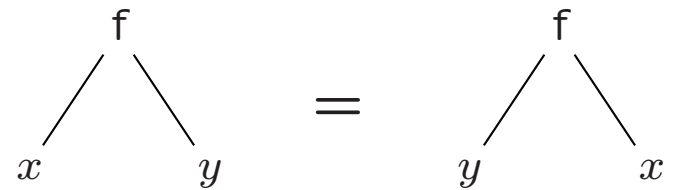
Given a signature \mathcal{F} with \mathcal{F}_{AC} some of the binary function symbols

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

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



associativity



commutativity

Likewise

A-tree automaton: tree automaton with associativity axioms of \mathcal{F}_A

C-tree automaton: tree automaton with commutativity axioms of \mathcal{F}_C

Transition move with associativity and commutativity

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

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

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

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

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

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

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

- 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

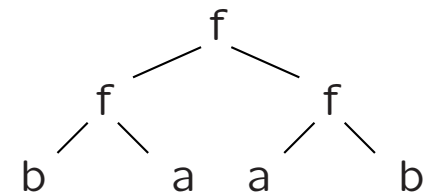
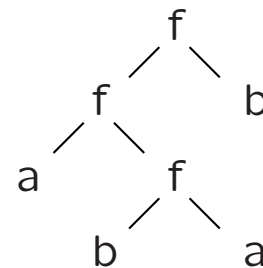
$$a \rightarrow q_a \quad b \rightarrow q_b \quad f(q_a, q_b) \rightarrow q \quad f(q, q) \rightarrow q$$

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

Then the accepted tree language L satisfies

$$|t|_a = |t|_b$$

for all t in L



Observations

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

$$|t|_a > |t|_b$$

$$2|t|_a = |t|_b$$

$$|t|_a + 1 = |t|_b$$

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	✓	✓	✓
closed under \cap	✓	✓	✓
closed under $()^c$	✓	✓	✗

regular TA < regular AC-TA < AC-TA

	regular TA	regular A-TA	A-TA
closed under \cup	✓	✓	✓
closed under \cap	✓	✗	✓
closed under $()^c$	✓	✗	✓

regular TA < regular A-TA < A-TA

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

Given a signature $\mathcal{F} = \{f\} \cup \{a_1, \dots, a_n\}$

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

$$P := C$$

$$| P \vee P$$

$$| P \wedge P$$

$$| \neg(P)$$

$$C := \#a_i \geq c \quad (c \in \mathbb{N})$$

$$| \#a_i + \#a_j \geq \#a_k$$

$$| \#a_i \times \#a_j \geq \#a_k$$

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

e.g. $L_{\#a=\#b}$ is the tree language whose element t satisfies $|t|_a = |t|_b$

Suppose tree languages recognizable with AC-tree automata are closed under $(\)^c$

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

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

Decidability results

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

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

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

monotone A-TA = monotone A-PTA $\overset{\sim}{\supseteq}$ monotone AC-PTA

\cup

\cup

regular A-PTA

$\overset{\sim}{\supseteq}$
 $\overset{\sim}{\supseteq}$

monotone AC-TA

\cup

$\overset{\sim}{\supseteq}$

\cup

regular A-TA

$\overset{\sim}{\supseteq}$
 $\overset{\sim}{\supseteq}$

regular AC-TA

\parallel

regular AC-PTA

\supseteq : tree language inclusion

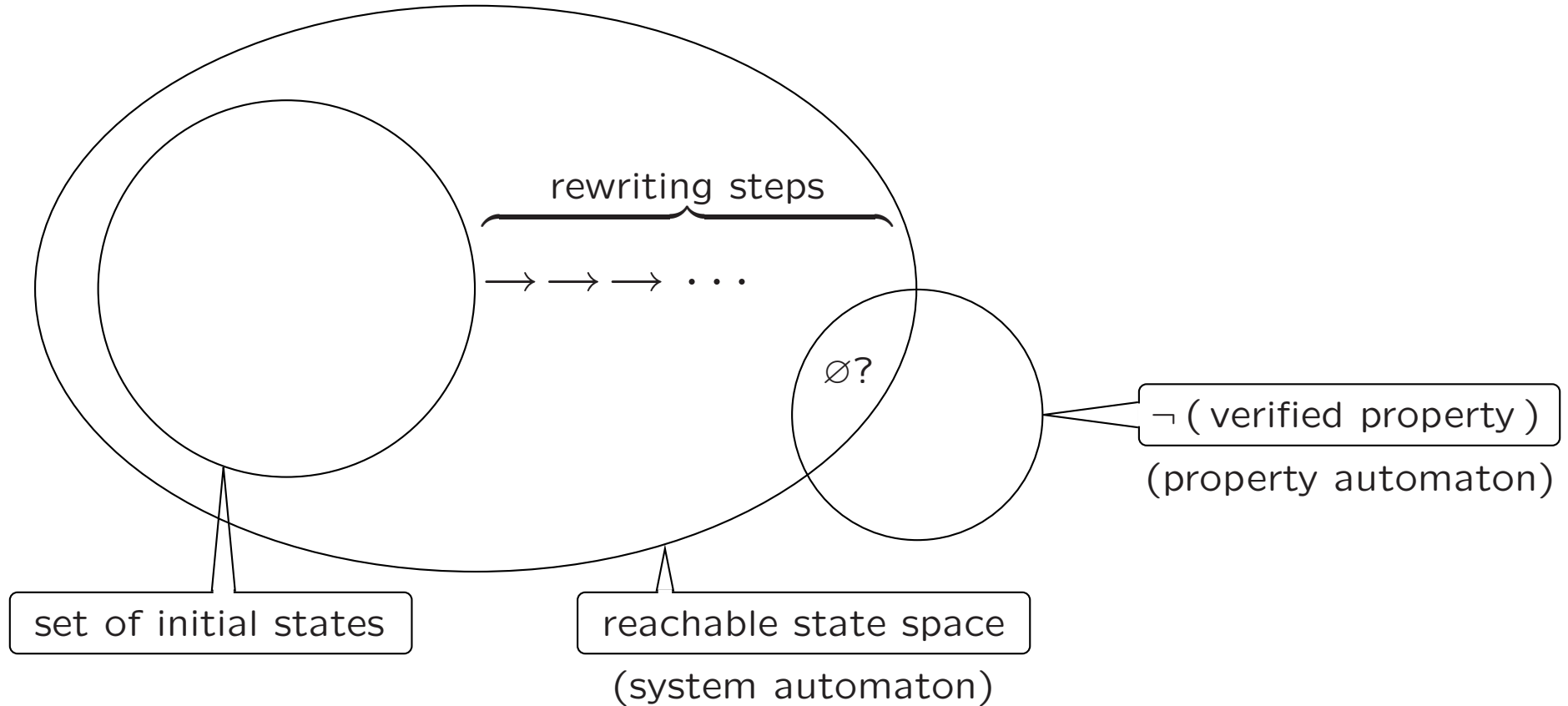
$\overset{\sim}{\supseteq}$: representability relation

II. Towards system verification

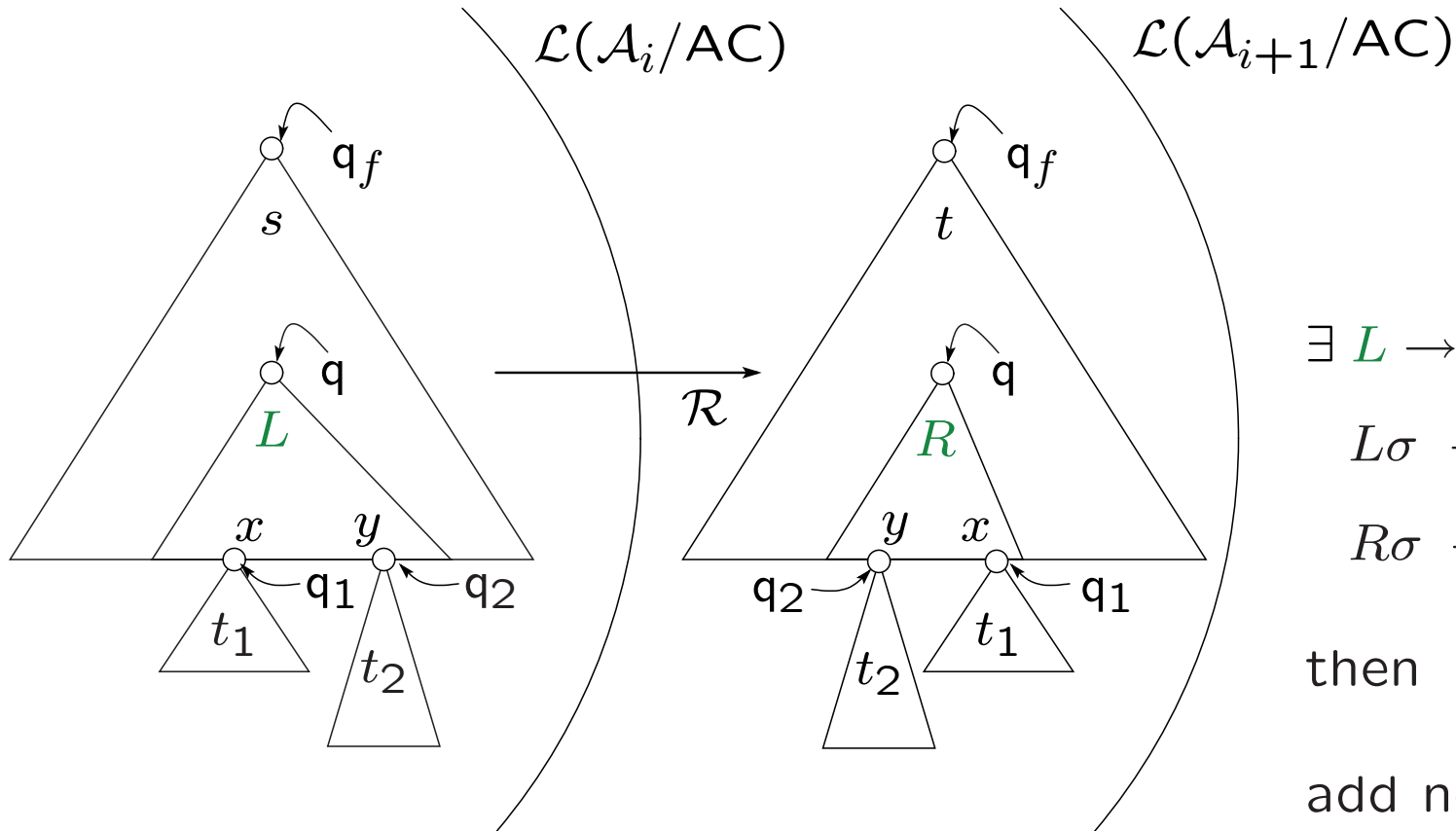
Model checking based on term rewriting and tree automata

Observation1 Undecidable fragments exist in the fixpoint computation

Observation2 Intersection-emptiness problem is EXPTIME-complete
for regular TA



One step of the procedure



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

$L\sigma \rightarrow_{\mathcal{A}_i/AC}^* q$ but

$R\sigma \not\rightarrow_{\mathcal{A}_i/AC}^* q$

then

add new transition rules
to \mathcal{A}_i/AC to satisfy

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

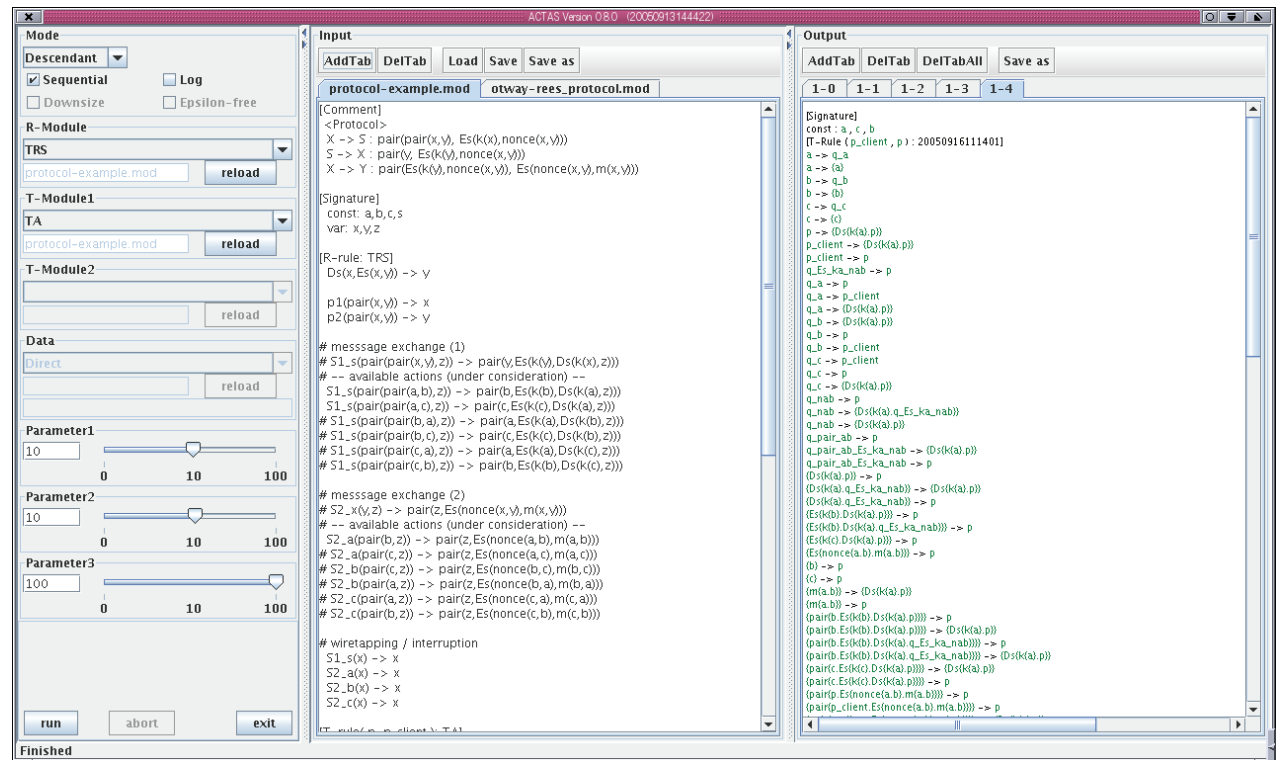
Specs on ACTAS & user-interface

– Platform OS:
Linux, Solaris

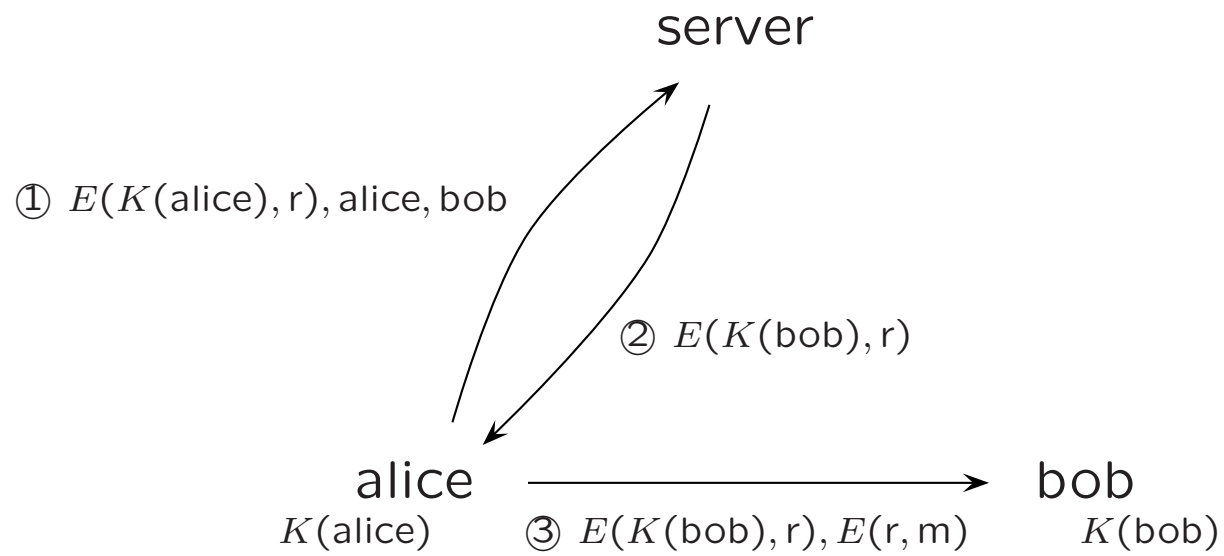
– Software requirement:
Java
ant (for rebuild)

– Memory:
~ 512M byte
(standard model)
~ 2G byte
(advanced model)

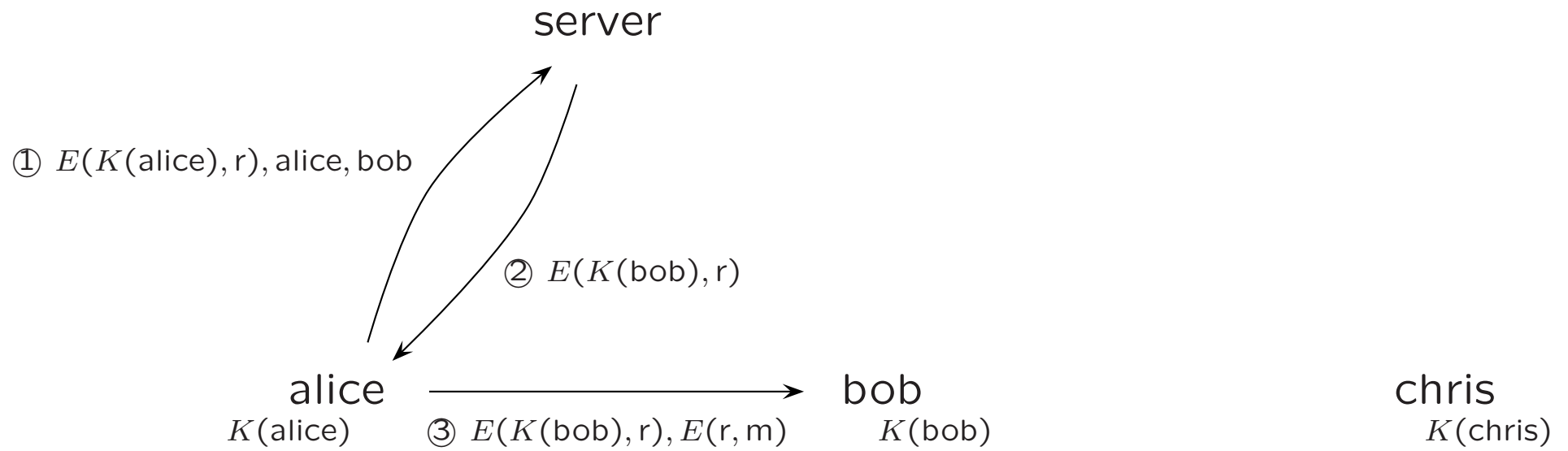
– Version:
0.8.20050912



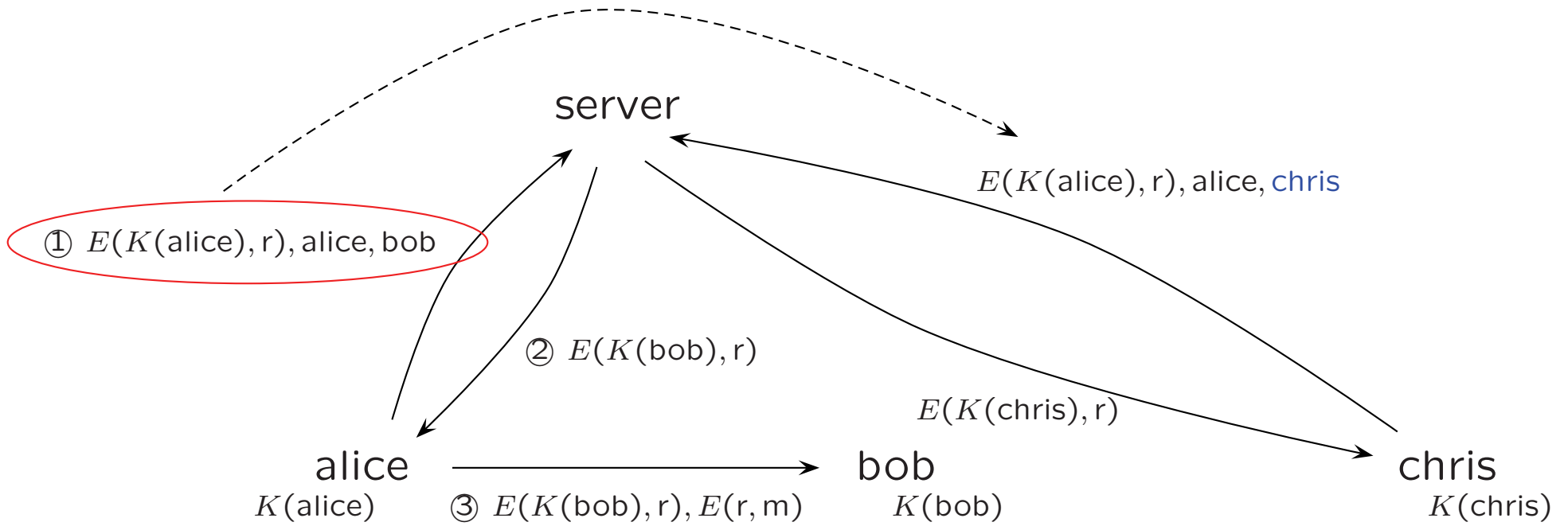
Security flaw in a network protocol (1)



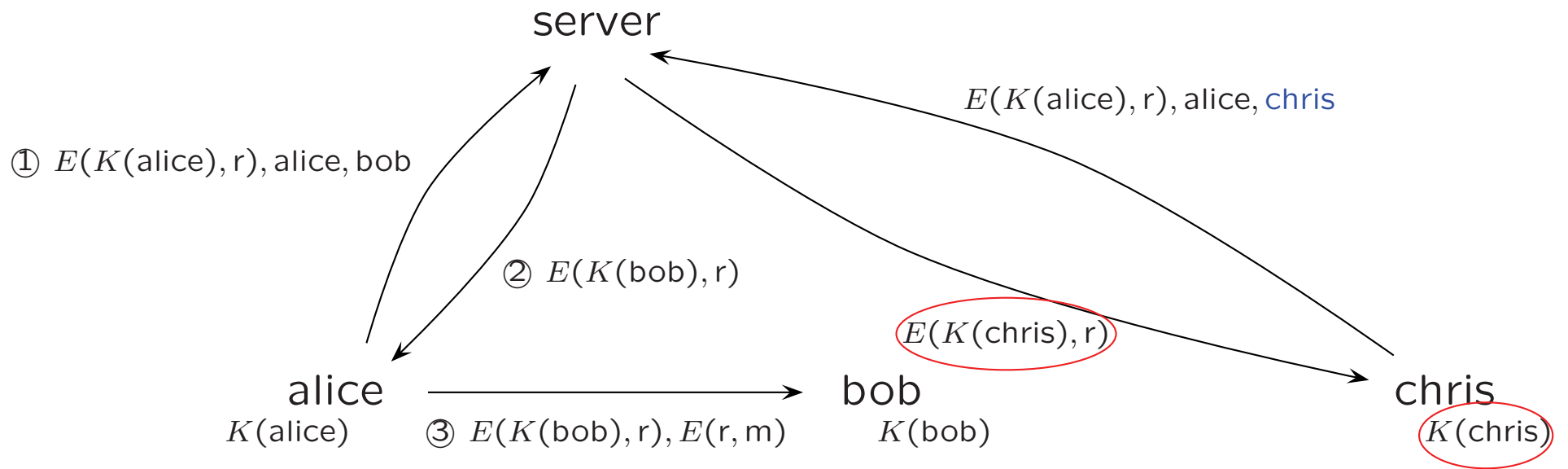
Security flaw in a network protocol (2)



Security flaw in a network protocol (3)



Security flaw in a network protocol (4)

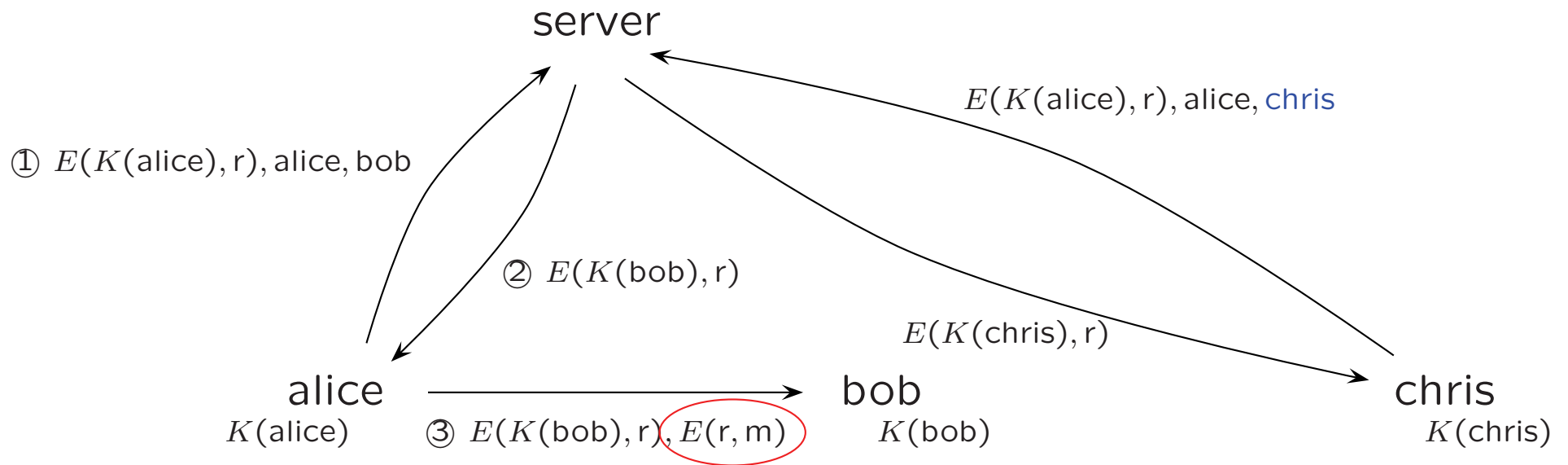


Axiom

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

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

Security flaw in a network protocol (5)



Axiom

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

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

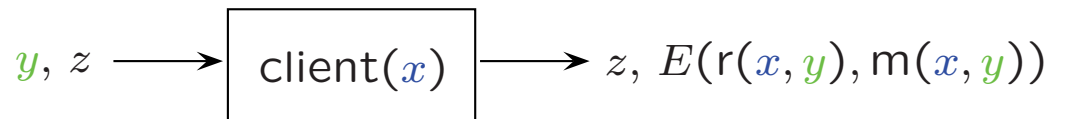
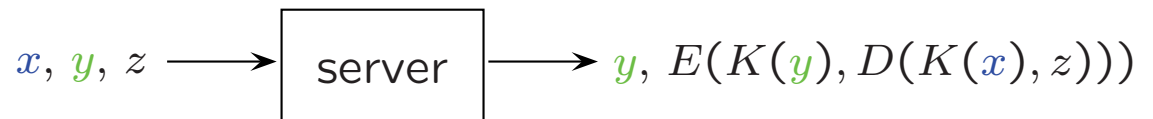
$$D(r, E(r, m)) \rightarrow m \text{ (secret message)}$$

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

```

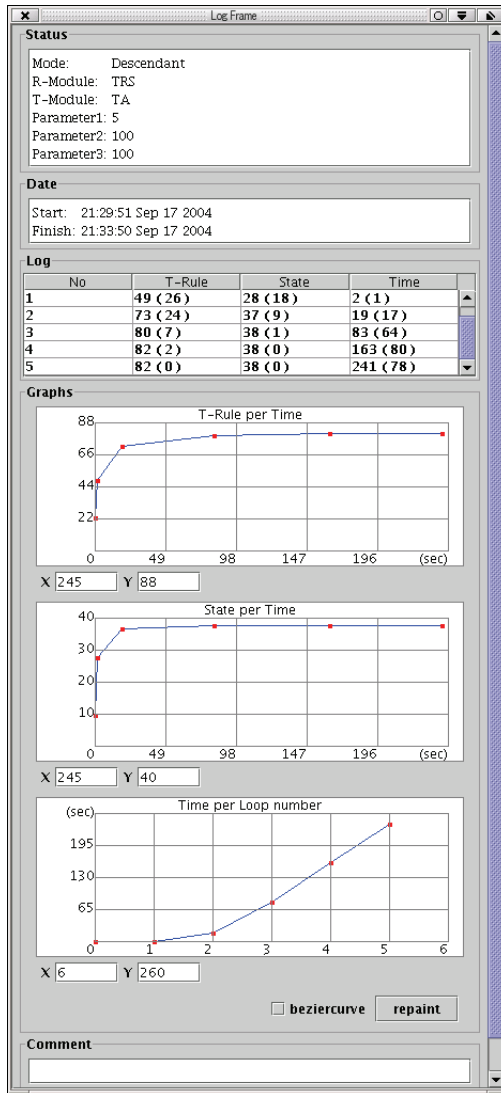


ACTAS specification (Lines 26 –)

```
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 message 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 $K(\text{chris})$ 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. $\forall i: \mathcal{L}(\mathcal{A}_i/\text{AC}) \subseteq \mathcal{L}(\mathcal{A}_{i+1}/\text{AC})$

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

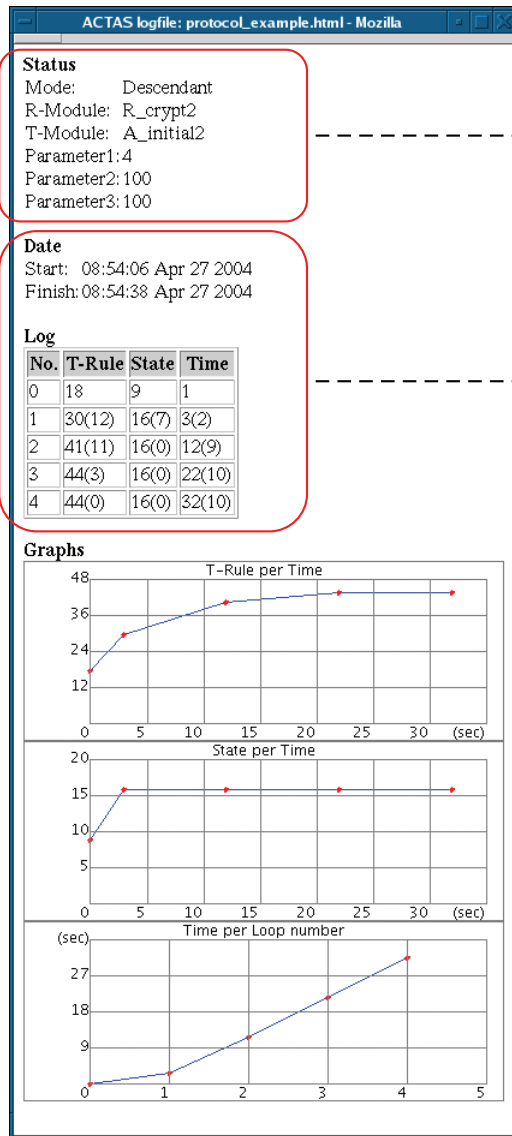
$\Rightarrow \exists i: \mathcal{L}(\mathcal{A}_j/\text{AC}) = \mathcal{L}(\mathcal{A}_{j+1}/\text{AC})$ for all $j \geq i$

($\Rightarrow \exists i: \mathcal{L}(\mathcal{A}_i/\text{AC}) = \mathcal{L}(\mathcal{A}_\infty/\text{AC})$)

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

\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 ($0 \leq i \leq 100$)

Execution time

Number of transition rules

Number of state symbols for each loop computation

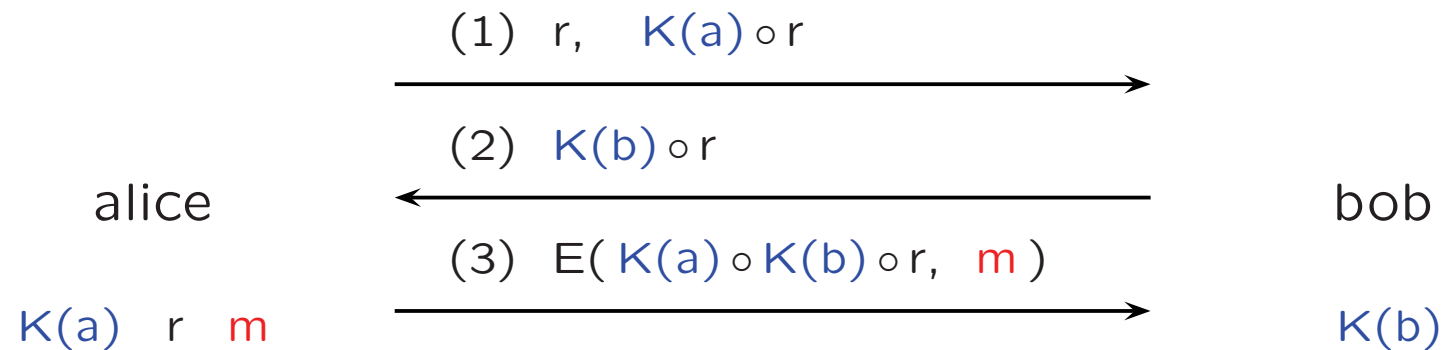
Graph1: number of transition rules \times time(sec)

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

Graph3: time(sec) \times loop number

(in HTML file format)

AC-axioms in encryption scheme



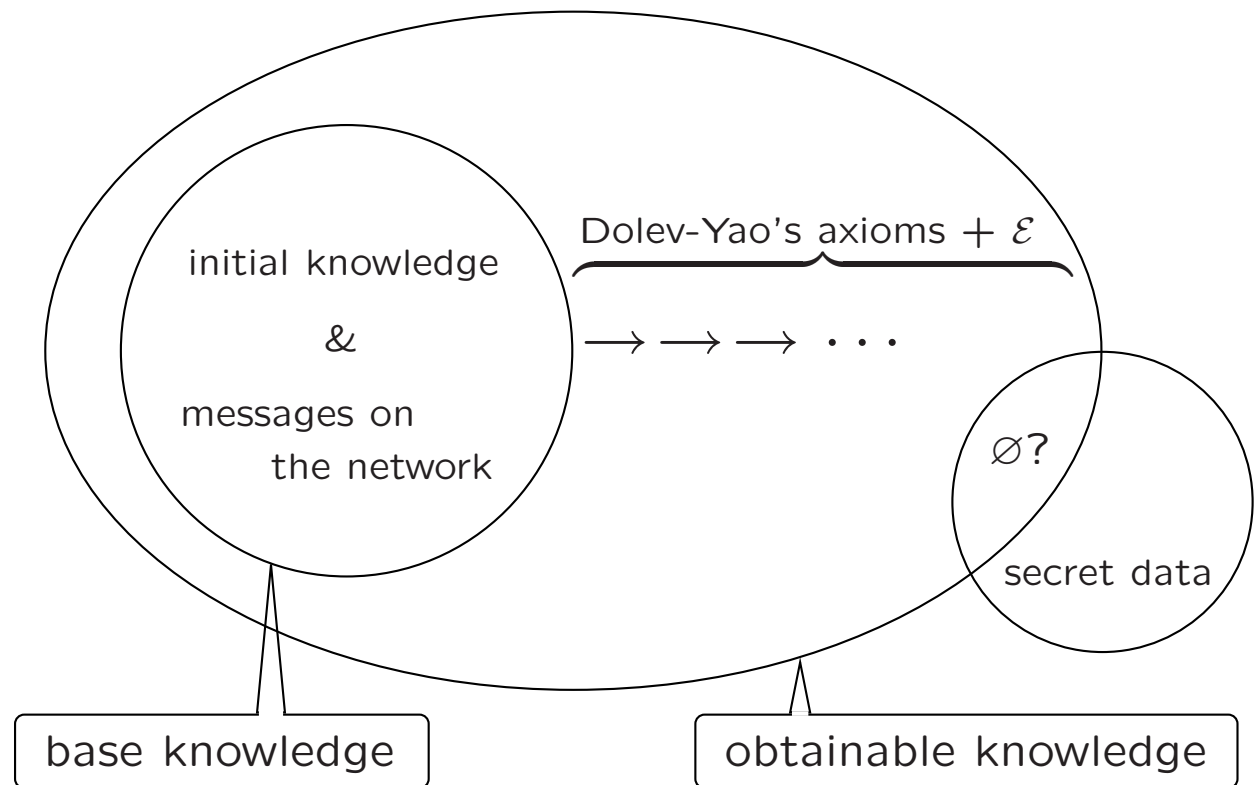
$K(a)$ $K(b)$: secret keys
 r : random number
 m : secret message
 E : encryption function
 \circ : AC symbol (infix operator)

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

1: [Signature]	29: # C's initial knowledge
2: AC: f	30: a -> p
3: const: a,b,c,m,r	31; b -> p
4: var: x,y	32: c -> p
5:	33: k(q_c) -> p
6: [R-rule: TRS2]	34: c -> q_c
7: Ds(x,Es(x,y)) -> y	
8:	
9: [T-rule(p): TA2]	
10: Ds(p,p) -> p	
11: Es(p,p) -> p	
12: f(p,p) -> p	
13:	
14: f(q_ka,q_r) -> p	
15: r -> q_r	
16:	
17: r -> p	
18:	
19: f(q_kb,q_r) -> p	
20: k(q_b) -> q_kb	
21: b -> q_b	
22:	
23: e(q_f_kba_r,q_m) -> p	
24: f(q_kab,q_r) -> q_f_kba_r	
25: f(q_kb,q_ka) -> q_k_ba	
26: k(q_a) -> q_ka	
27: a -> q_a	
28: m -> q_m	



Intruder deduction problem (general version)

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

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

Note 1. In the previous setting

L : initial knowledge \vdash 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

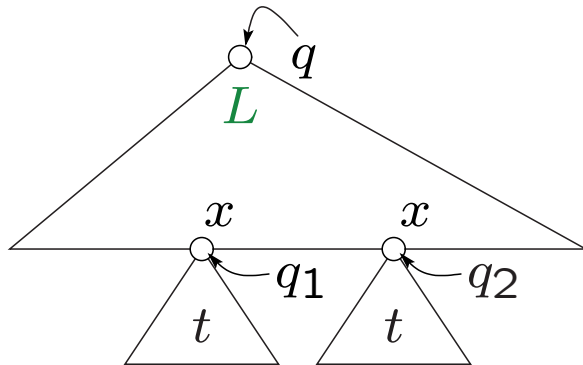
Note 3. The emptiness problems for AC-TA and regular AC-TA are decidable

Non-left-linear case

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

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

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

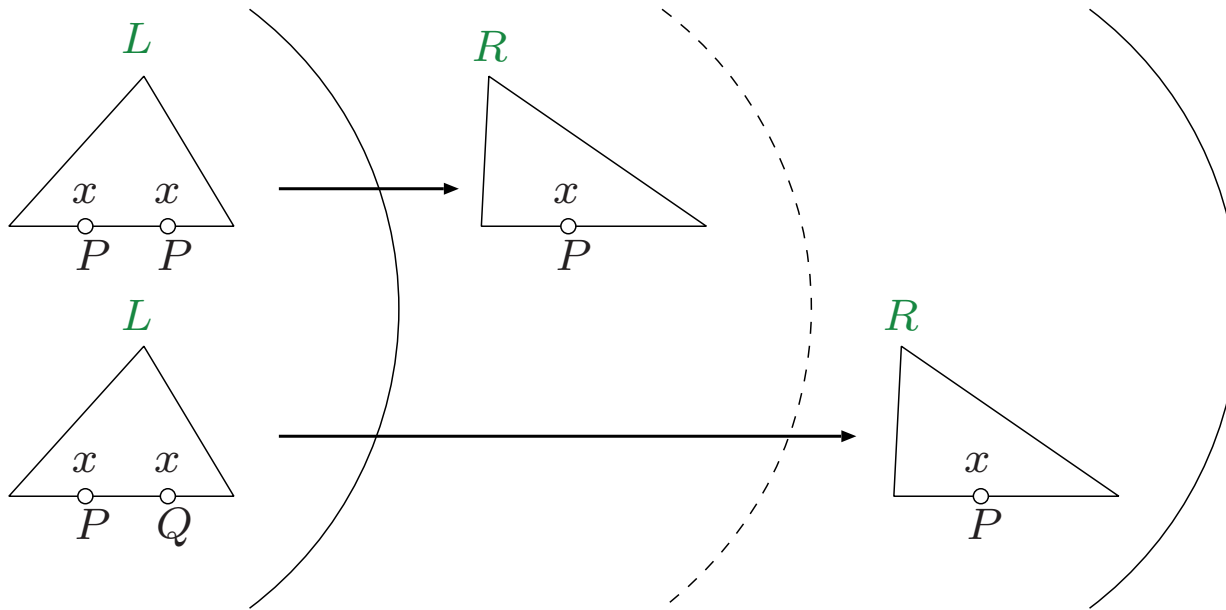


- In fully automated verification, under- or over-approximation is needed to be applied

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



$\#(\text{T-rule}) = 21$
 $\#(\text{state symbols}) = 16$

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

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	—

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

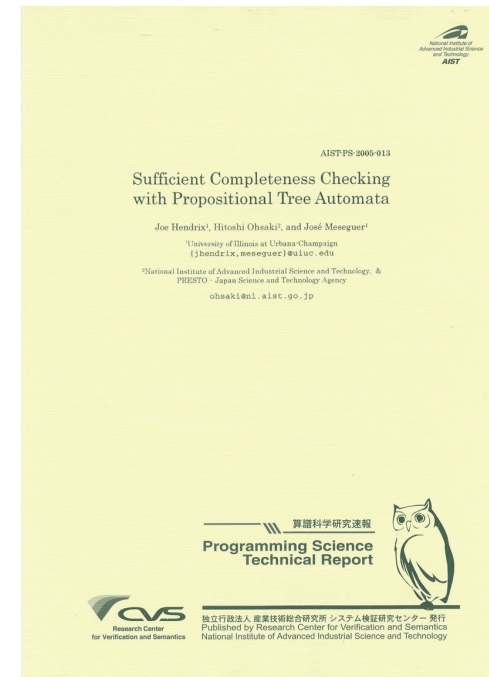
ENTCS 124, pp. 97–111

[5] Sufficient Completeness Checking with Propositional Tree Automata

Joe Hendrix & Hitoshi Ohsaki & José Meseguer

Technical Report AIST-PS-2005-013

AIST/CVS, 2005 (**new!**)



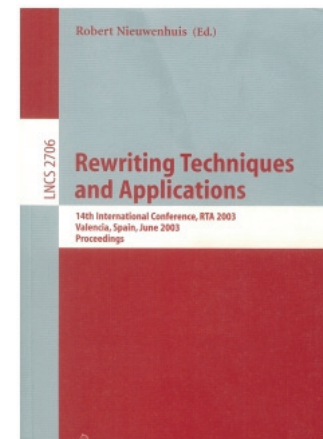
Publications II: equational tree automata

[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



©Springer-Verlag

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)



The screenshot shows the homepage for CETA: A Library for Equational Tree Automata. The page is titled "Ceta: A Library for Equational Tree Automata" and is displayed in a Mozilla browser window. The content is organized into several sections:

- Summary:** Describes Ceta as a library for reasoning about tree languages, based on propositional tree automata. It mentions its development at the University of Illinois at Urbana-Champaign and its joint project with the National Institute of Advanced Industrial Science and Technology in Japan. The primary software developer is Joe Hendrix.
- Download:** States that the source code for Ceta Alpha 1 is available as a gzipped tar file. It lists requirements: a C++ compiler (gcc 3.3 or later recommended), Boost C++ Libraries (version 1.32 or later recommended), and Doxygen (version 1.4 or later recommended). A table lists the available files for download:

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

- Documentation:** States that Ceta's documentation is automatically generated using Doxygen and is available online.
- Announcement Mailing list:** Encourages users to subscribe to the ceta-announce mailing list for updates.
- Development:** Provides information on how to track development using the latest source, available as a darcs repository.

Logos for W3C, XHTML, CSS, and Doxygen are visible at the bottom right of the page.

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

Copyright © 2005 Hitoshi Ohsaki

All rights reserved. No part of this lecture note may be reproduced or stored in a retrieval system, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior consent of the publisher.