

Symbolic Constraint Solving

Ralf Treinen

Laboratoire Spécification et Vérification (LSV)

École Normale Supérieure de Cachan – CNRS – INRIA Futurs
France

Plan

1. Constraints
2. Feature trees and feature constraints
3. Methods for constraint solving:
 - (a) Simplification of constraints
 - (b) Translation to other problems
 - i. Subtyping constraints
 - (c) Reachability in inference systems
 - i. Subsumption constraints on feature trees
 - ii. Symbolic analysis of cryptographic protocols
4. Conclusions and future work

What are Constraints good for?

A **constraint system** is a means to specify tuples of values, and

tuples of values are everywhere in computer science:

- stateful programming : *states*
- logic programming : *atoms*
- functional programming : *environments*
- data bases : *rows*
- ...

Constraints are employed for ...

- **Extending** and **generalizing** a computational mechanism that operates on *sets of tuples*:
 - from *logic programming* to *constraint logic programming* [Jaffar&Lassez '86]
 - from *deduction systems* to *constrained deduction systems* [Kirchner&Kirchner&Rusinowitch '90]
- **Analyzing** a computational mechanism:
 - Set-based program analysis [Heintze&Jaffar '90]
 - Symbolic verification of protocols (see later)
 - Redundancy in constrained inference systems [Kirchner&Kirchner&Rusinowitch '90]

Constraints are Logical Formulae because ...

... because predicate logic gives us the the means to

- address components of tuples: *variables*
- intersect specifications: *conjunction*
- restrict the name space: *existential quantification*
- extend the name space: *cylindrification*

Constraint Systems

A **constraint system** is given by

- a **language** of predicate logic: function and predicate symbols
- a **model**: a domain of values, and an interpretation of the symbols
- a set of **constraints**: set of first-order formulae, closed under conjunction.

Computational Service Required from a Constraint System

Basic question: **Constraint Solving**

Does a given constraint ϕ have a solution ?

Possible sets of constraints

- **Basic Constraint**: conjunction of atomic constraints

$$a_1 \wedge \dots \wedge a_n$$

- **Entailment**: implication of basic constraints

$$\forall \bar{X} (\phi_1 \rightarrow \exists \bar{Y} \phi_2)$$

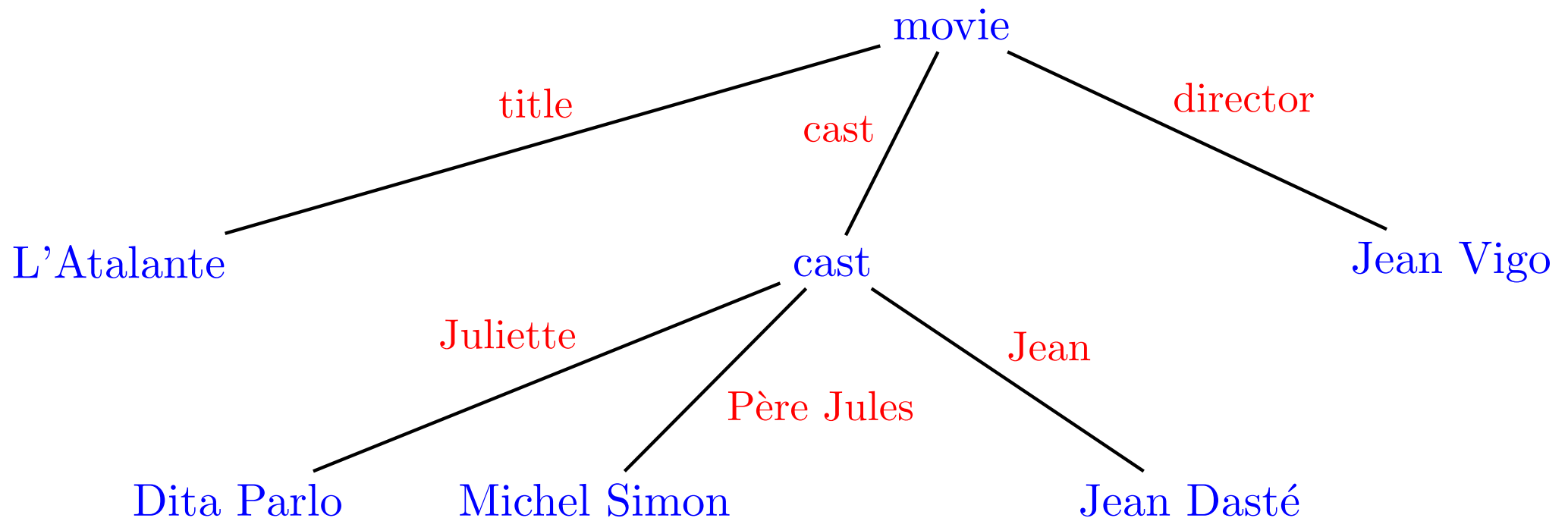
- All **first-order formulas**

Plan

1. Constraints
2. Feature trees and feature constraints
3. Methods for constraint solving
 - (a) Simplification of constraints
 - (b) Translation to other problems
 - i. Subtyping constraints
 - (c) Reachability in inference systems
 - i. Subsumption constraints on feature trees
 - ii. Symbolic analysis of cryptographic protocols
4. Conclusions and future work

Symbolic Constraint Systems: Feature Constraints

[Aït-Kaci, Podelski, Smolka '92]: Feature Trees model data as trees with **labeled nodes** and **labeled edges**:



Feature Trees

- definition builds on a notion **feature structures** well-established in computational linguistics
- fine-grained constraints, i.e.:
 $x[f]y$: there is an edge f from x to y .
- structure should be self-explanatory
(c.f. *semi-structured data*, *XML*)
- features are **functional**: from any node at most one edge with a given feature symbol.

Plan

1. Constraints
2. Feature trees and feature constraints
3. Methods for constraint solving
 - (a) Simplification of constraints
 - (b) Translation to other problems
 - i. Subtyping constraints
 - (c) Reachability in inference systems
 - i. Subsumption constraints on feature trees
 - ii. Symbolic analysis of cryptographic protocols
4. Conclusions and future work

Constraint Solving by Simplification

Rewrite rules on constraints such that

- **Soundness**: transformations preserve validity
- **Termination**: every reduction sequence finite
- **Completeness**: normal forms: satisfiability easy to decide

Example: rule for **subtree constraints** $x[f]y$:

$$\text{(Fun)} \quad \frac{x[f]y \wedge x[f]z}{x[f]z \wedge y = z}$$

Variations of Constraint Simplification

- **Saturation** of constraints: ordering constraints
- **Quantifier elimination**: Dealing with arbitrary first-order formula by rewriting
- **Weak quantifier elimination**: Simplifying quantification structure of arbitrary first-order formulae

Results Constraint Solving by Simplification

- **CFT**: label, feature and arity constraints
[Smolka&*RT* '92; Backofen&*RT* '94]
- **EF**: features as first-class values
[*RT* '93]
- **FX**: parameterized by the structure of feature symbols
[*RT* '97])
- **MT**: membership constraints for infinite trees
[Niehren&Podelski&*RT* '93]
- **FT**: label and feature constraints
[*RT* '03]

Plan

1. Constraints
2. Feature trees and feature constraints
3. Methods for constraint solving:
 - (a) Simplification of constraints
 - (b) Translation to other problems
 - i. Subtyping constraints
 - (c) Reachability in inference systems
 - i. Subsumption constraints on feature trees
 - ii. Symbolic analysis of cryptographic protocols
4. Conclusions and future work

Constraint Solving by Translation

Show that a structure is **automatic** [Blumensath&Grädel '00]:

the set of solutions of any atomic constraint
is a **regular language**

⇒ any definable relation is (represented by) a regular language

⇒ first-order theory decidable

Example: Presburger arithmetic.

Example: **Non-structural** Subtype constraints

- Language: $\perp, \top, f_1, f_2, f_3, \dots \leq$.
- Model:
 - Domain: finite terms over $\perp, \top, f_1, f_2, \dots$
 - Interpretation of $\leq$:

$$\perp \leq x \quad \text{for all } X$$

$$x \leq \top \quad \text{for all } X$$

$$f_i(t_1, \dots, t_n) \leq f_i(s_1, \dots, s_n) \quad \text{iff } t_1 \leq s_1, \dots, t_n \leq s_n$$

Satisfiability of basic constraints: n^3

[Palsberg, Wand, O’Keefe ’97]

Entailment: open question (for one binary f)

Subtyping Constraints in Automata

Results shown in [Su&Aiken&Niehren&Priesnitz&RT '02]

- Monadic signature: automatic structure (first-order theory decidable)
- General case: first-order theory captured by *tree automata with component-wise tests*
- General case: first-order theory undecidable

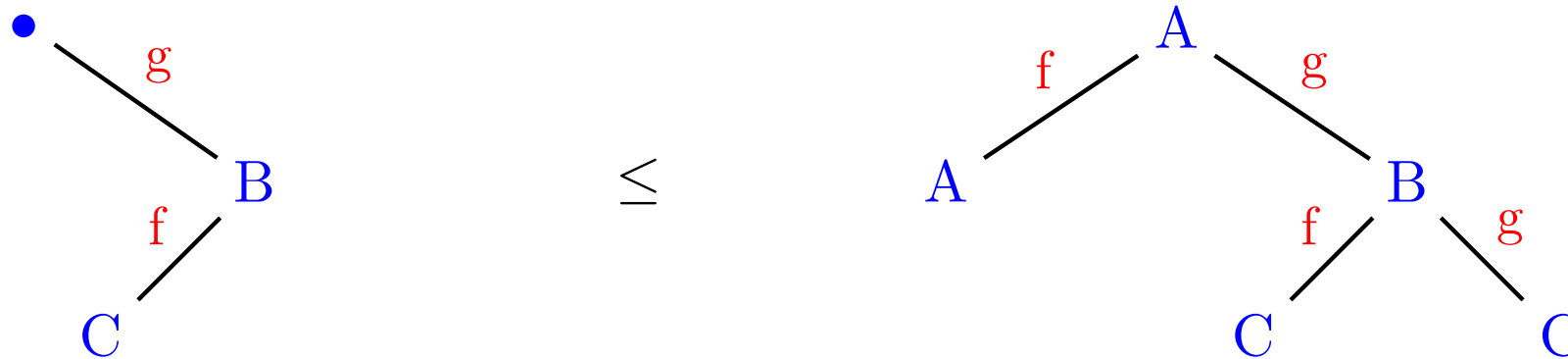
Tree automata with component-wise tests: emptiness undecidable
[RT '00; Su&Aiken&Niehren&Priesnitz&RT '05]

Plan

1. Constraints
2. Feature trees and feature constraints
3. **Methods for constraint solving**
 - (a) Simplification of constraints
 - (b) Translation to other problems
 - i. Subtyping constraints
 - (c) **Reachability in inference systems**
 - i. **Subsumption constraints on feature trees**
 - ii. Symbolic analysis of cryptographic protocols
4. Conclusions and future work

Reachability in Inference Systems: Subsumption Constraints $FT \leq$

Weak subsumption between partially labeled feature trees:



- For any node label symbol A :
 $Ax : x$ has root label A
- For any feature symbol f :
 $x[f]y : \text{edge } f \text{ from the root of } x \text{ to the root of } y$

- Satisfiability of basic constraints: n^3
[Müller, Niehren, Podelski '97]
- Entailment without quantifiers: n^3
[Müller, Niehren, Podelski '97]

Our results in [Müller&Niehren&*RT* '01]

- Entailment $\forall \bar{X} (\phi \rightarrow \exists \bar{Y} \psi)$: PSPACE-complete
- First-order theory undecidable

Path constraints for $FT \leq$

Set of **path constraints** Π :

- $x?[\pi] \leq y?[\omega]$: if $\pi \in Dom(x), \omega \in Dom(y)$ then $x[\pi] \leq y[\omega]$.
- ...

Lemma: *If $\phi \wedge \psi$ satisfiable then*

$$FT \leq \models \forall \bar{X} (\phi \rightarrow \exists \bar{Y} \psi)$$

iff

$$\{p \in \Pi \mid \exists \bar{Y} \psi \models p\} \subseteq \{p \in \Pi \mid \phi \models p\}$$

Recognizing Implied Path Constraints

Lemma:

$$\psi \models p \iff \psi \vdash p$$

Lemma: *Concrete syntax of $\{p \in \Pi \mid \psi \vdash p\}$ is recognized by an automaton of size $\text{poly}(|\psi|)$.*

Concrete syntax: factor out the longest common suffix of paths.

$$x?[f f h k] \leq y?[g h k] \Rightarrow \mathbf{x}[f f] \leq \mathbf{y}[g] \# h k$$

Theorem: *[Müller & Niehren & RT '98] Entailment*

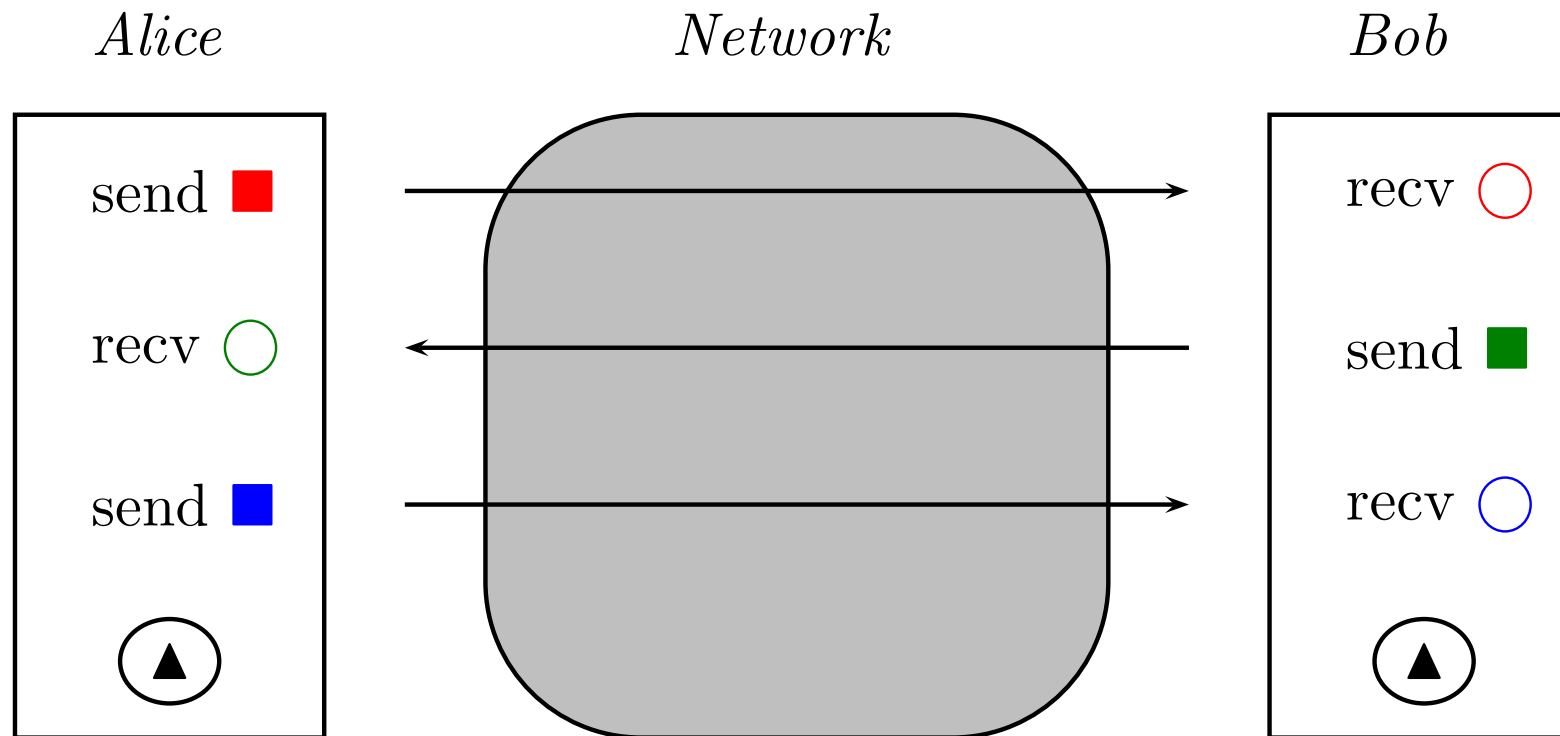
$\forall \bar{X} (\phi \rightarrow \exists \bar{Y} \psi)$ in $\text{FT} \leq$ is PSPACE-complete.

Plan

1. Constraints
2. Feature trees and feature constraints
3. **Methods for constraint solving**
 - (a) Simplification of constraints
 - (b) Translation to other problems
 - i. Subtyping constraints
 - (c) **Reachability in inference systems**
 - i. Subsumption constraints on feature trees
 - ii. **Symbolic analysis of cryptographic protocols**
4. Conclusions and future work

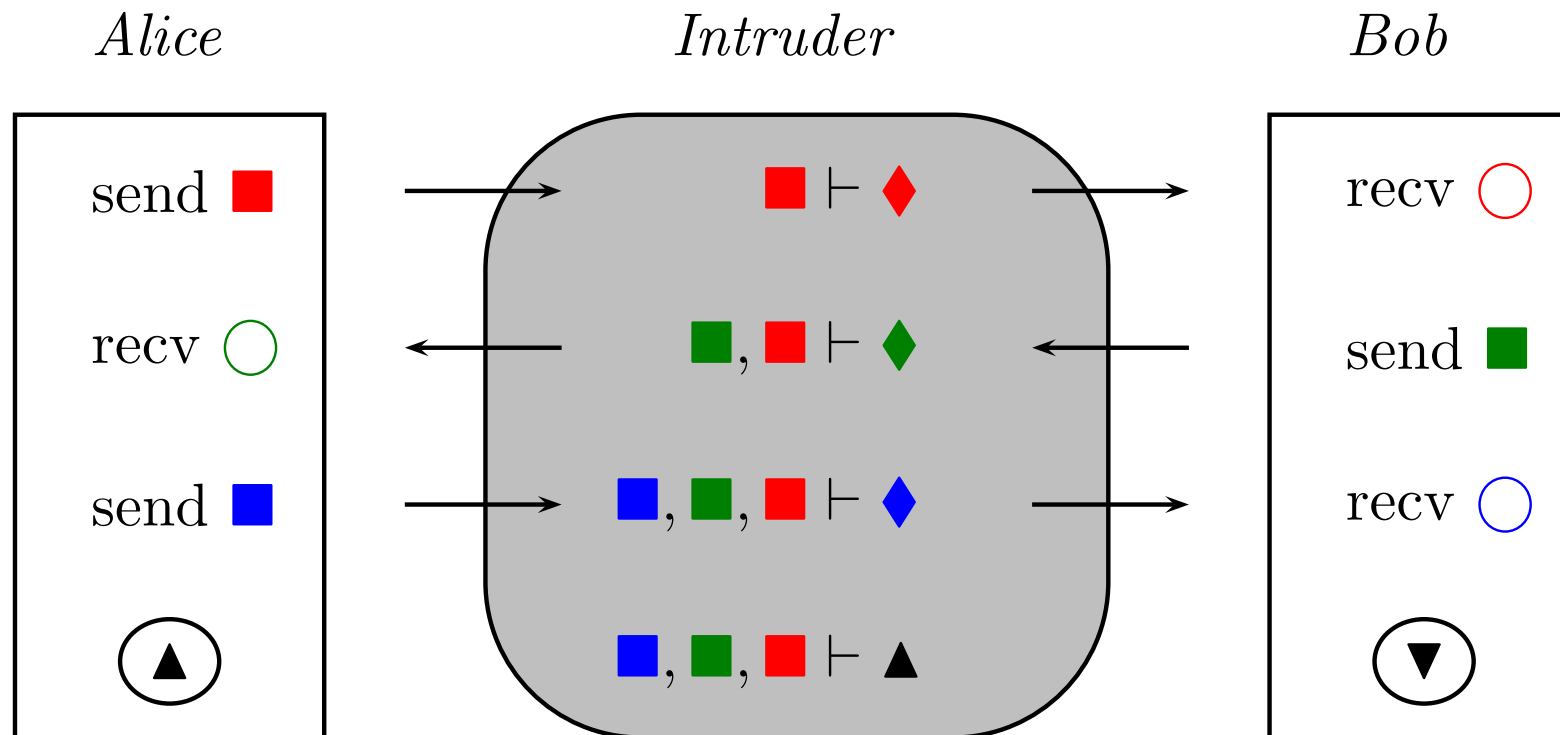
Symbolic Verification of Cryptographic Protocols

Faithful execution of a cryptographic protocol:



The Network is the Intruder

The attacker may replace messages:



The Dolev-Yao Model of Intruder Capabilities

Messages are modeled as terms in a free term algebra

$$(A) \quad \frac{u \in T}{T \vdash u}$$

$$(UL) \quad \frac{T \vdash \langle u, v \rangle}{T \vdash u}$$

$$(P) \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle}$$

$$(UR) \quad \frac{T \vdash \langle u, v \rangle}{T \vdash v}$$

$$(C) \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \{u\}_v}$$

$$(D) \quad \frac{T \vdash \{u\}_v \quad T \vdash v}{T \vdash u}$$

$$(F) \quad \frac{T \vdash u_1 \quad \dots \quad T \vdash u_n}{T \vdash f(u_1, \dots, u_n)} \quad f \in \Sigma^-$$

Modeling Attacks by Constraints

- fix the instances of the protocol
- guess a linearisation of the execution

$$\begin{array}{ccc} t_0, \dots, t_n & \vdash & u_1 \\ \vdots & & \vdots \\ t_0, \dots, t_n, \dots, t_k & \vdash & u_{k+1} = \textit{secret} \end{array}$$

Satisfiability is NP-complete [Rusinowitch, Turuani '03].

Perfect Cryptography Assumption

Knowledge about some encrypted message can only be obtained when encryption key is known.

In reality, cryptographic primitives have properties which

- are necessary for the protocol
- can be exploited by an intruder

Algebraic properties: expressed by equational axioms

Some Important Algebraic Properties

Properties of **exclusive or** (symmetric encryption)

associativity $(x \oplus y) \oplus z = x \oplus (y \oplus z)$

commutativity $x \oplus y = y \oplus x$

unit $0 \oplus x = x$

nilpotence $x \oplus x = 0$

Homomorphism properties

(asymmetric encryption, block chaining modes)

homomorphic hash functions $h(x \oplus y) = h(x) \oplus h(y)$

distributive encryption $\{x \oplus y\}_k = \{x\}_k \oplus \{y\}_k$

Weakening the Perfect Cryptography Assumption

Extend the Dolev-Yao deduction system by

$$(E) \quad \frac{T \vdash u \quad u =_E v}{T \vdash v}$$

In reality:

represent $=_E$ by a **canonical rewrite system** modulo some background equational theory

The Ground Case: Intruder Deduction Problem

- Deciding validity of ground constraints $t_1, \dots, t_n \vdash t$
- Prerequisite to resolution of non-ground constraints

[McAllester '93]:

- A proof of $T \vdash u$ is **local** if every intermediate node is a *subterm* of T, u .
- An inference system is **local** if every provable theorem has a local proof.

Theorem: [McAllester'93]

Provability in local inference systems is decidable in PTIME.

Intruder Deduction Problem modulo some Equational Theories

- Distributive encryption: PTIME (*locality*)
[Comon&RT '03]
- ExOr + distributive encryption: EXPTIME (*locality*)
PTIME in the binary case (*prefix rewrite system* [Caucal'90])
[Lafourcade&Lugiez&RT '05]
- ExOr/AG + homomorphic hash function: PTIME
(*locality + linear equations over polynomial rings*)
[Delaune'05]

Intruder Constraints modulo ExOr + homomorphism

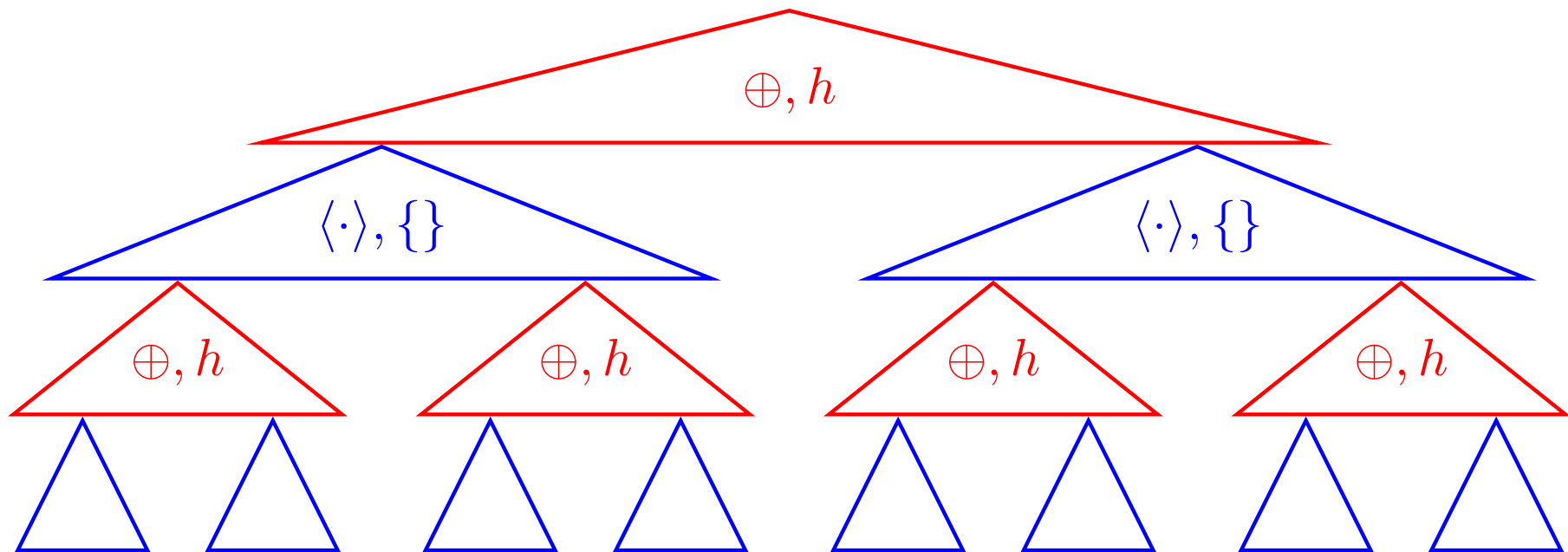
- ExOr: [Comon, Shmatikov'03; Chevalier, Küsters, Rusinowitch, Turuani '03]
- ExOr / Abelian groups: [Millen, Shmatikov '05].

Theorem: *[Delaune&Lafourcade&Lugiez&RT '05] Satisfiability of intruder constraints modulo ExOr + homomorphism is decidable under mild assumptions.*

Ingredients of our proof:

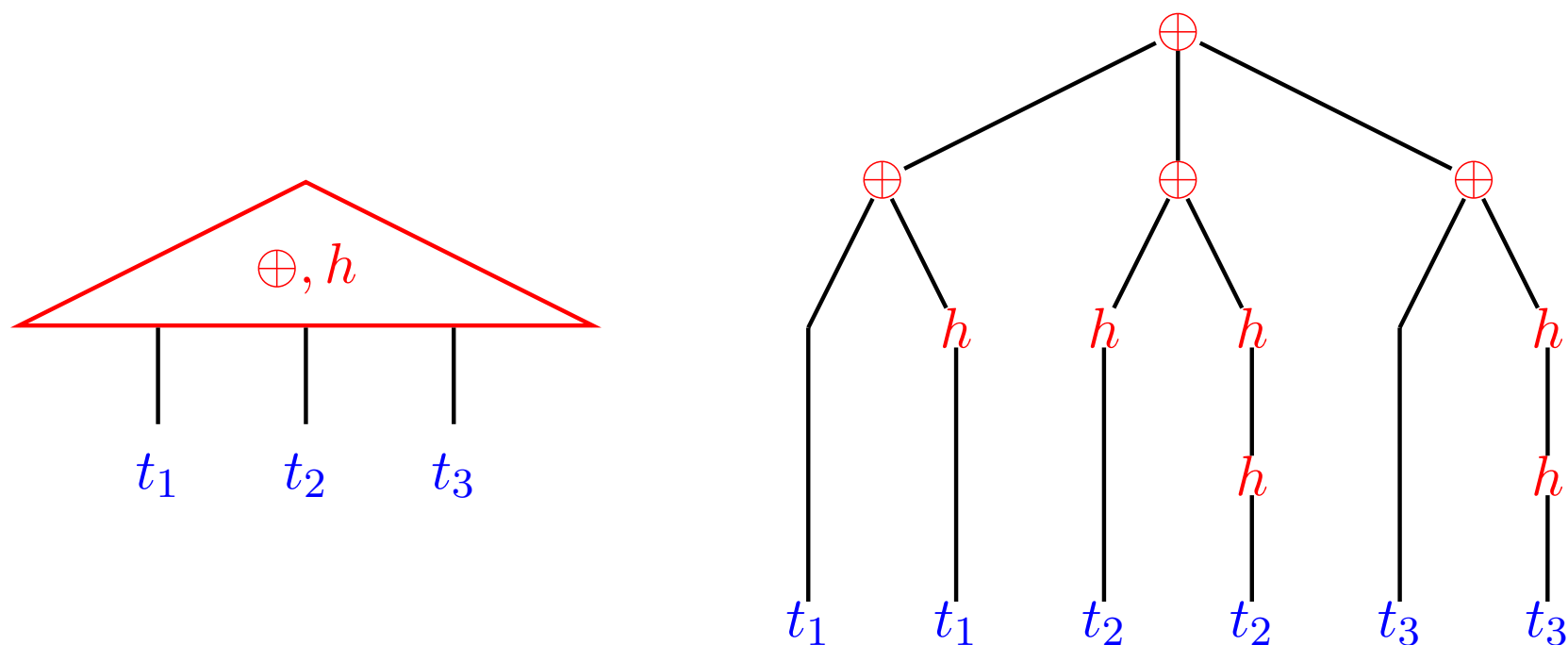
- Decidability of ground constraints
- General unification modulo ExOr + homomorphism
- Equation systems over integral domains

Stratification of Terms



- Non-standard layer: equational theory
- Standard layer: not subject to equational theory

What's in a Non-standard Layer?



Linear combination of t_1 , t_2 , t_3 with coefficients in $\mathbb{Z}/2\mathbb{Z}[h]$

$$(1 \oplus h) \odot t_1 \oplus (h \oplus h^2) \odot t_2 \oplus (1 \oplus h^2) \odot t_3$$

How to solve $\vdash$ constraint systems for ExOr + homomorphism

1. from $\vdash$ constraints to $\vdash_1$ constraints
generalisation of the locality of $\vdash$
2. from $\vdash_1$ constraints to $\vdash_{M_E}$ constraints
general ExOr+h-unification is decidable and finitary
[Delaune&Lafourcade&Lugiez&RT '05]
3. abstract **subterms** by constants
4. from $\vdash_{M_E}$ to ground $\vdash_{M_E}$ constraints
solvable system admits small contexts
determine value of variables from the contexts
5. check satisfiability of ground $\vdash_{M_E}$ constraint system
[Lafourcade&Lugiez&RT '05; Delaune '05]

Conclusion and Future Work (1)

Constraint solving methods:

- Simplification: local method
profits from the term rewriting tool box
- Translation: global method
profits from automata techniques
- Inference systems: global method
profits from results from automated deduction

Conclusion and Future Work (2)

Open questions:

- a methodology for constraint solving by analysis of inference systems
- using automata techniques for restricted sets of constraints
- investigate relation between the three proof techniques

Conclusion and Future Work (3)

Symbolic verification of cryptographic protocols:

- Constraint solving for richer equational theories:
 - ExOr + distributive encryption
 - AG + homomorphism / distributive encryption
- Verification of richer protocols
 - global variables
 - conditionals, case distinctions
 - iterative constructs