

A Semantic Foundation for Trust Management Languages with Weights: An Application to the *RT* Family

S. Bistarelli ^{1,2}, F. Martinelli ², F. Santini ¹

¹Dipartimento di Matematica e Informatica, Perugia

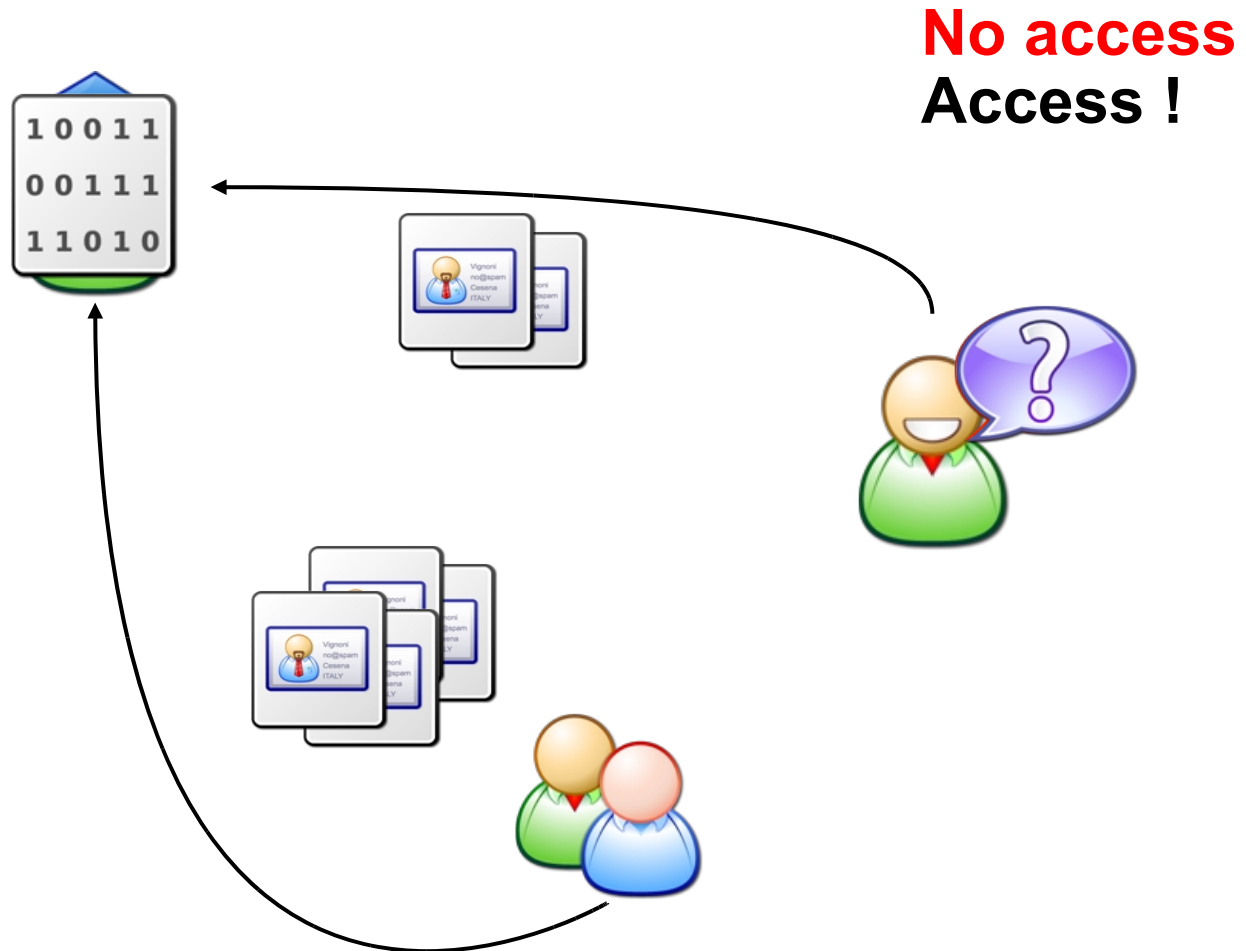
²Istituto di Informatica e Telematica CNR, Pisa



Abstract

- *We extend the Datalog language (we call it Datalog^w) in order to deal with weights on ground facts*
- *The weights are chosen from a proper c-semiring*
- *We use Datalog^w as the basis to give a uniform semantics to the RT (Role-based Trust Management) language family: the new family is called RT^w*
- *Each credential is associated with a value*
- *The value corresponds to a preference or cost associated to the credentials (instead of a plain “yes or no”)*
 - *More informative*
 - *Uncertainty*
- *The approach is rather generic and can be applied to other trust management languages*

The Authorization Scenario



What's new?

➤ The credentials are “weighted” according to some criteria



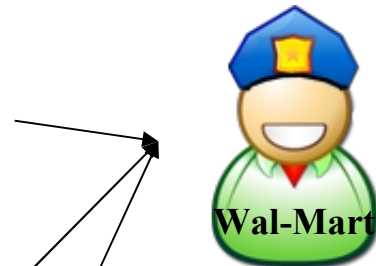
I am an old customer



I have already spent a lot of money



I bought many electronic devices



iPod = ~~249~~ €

= 200 €

Benefits of the Approach

- A declarative language with a formal foundation
- In this talk we use it to extend the RT family but...
 - Delegation Logic
 - Binder
- The credentials are *soft*:
 - Cost
 - Preference
 - Uncertainty
- c-semirings are flexible and parametric
 - Each metric that can be represented with it, is good!
- Not-weighted credentials can be represented as well

A bridge between “rule-based” trust management and
“reputation-based” trust management

Weighted Datalog

- Rules are the same as in Datalog

$$R_0(t_{0,1}, \dots, t_{0,k_0}) :- R_1(t_{1,1}, \dots, t_{1,k_1}), \dots, R_n(t_{n,1}, \dots, t_{n,k_n})$$

- Facts slightly change; they are in the form:

$$R_i(x_{i,1}, \dots, x_{i,k_i}) :- a$$

- where $a \in A, S = \langle A, +, \times, 0, 1 \rangle$

- A is the set of values (e.g. preferences, costs)
- + defines a partial order \leq_S over S: $a \leq_S b$ means b is better
- \times is used to compose the values
- 0 and 1 are the bottom and top elements

- A very simple program example

```
s(X)      :- p(X, Y) .  
p(a, b)   :- q(a) .  
p(a, c)   :- r(a) .  
q(a)      :- t(a) .  
t(a)      :- 2 .  
r(a)      :- 3 .
```

Extending RT_0 : RT_0^W

➤ A *role* in RT_0 (and RT_0^W) takes the form of an entity (A) followed by a role name (R) separated by a dot = $A.R$

➤ Each entity A has the authority to define who are the members of each role of the form $A.R$

- IEEE.member \leftarrow Alice

➤ Each statement defines one role to contain:

- An entity
- Another role: IEEE.member \leftarrow IEEE.studentMember
- Other expressions that evaluate to a set of entities

```
{
  EPub.disct  $\leftarrow$  EPub.preferred  $\cap$  EPub.student
  EPub.preferred  $\leftarrow$  EOrg.preferred
  EOrg.preferred  $\leftarrow$  IEEE.member
  EPub.student  $\leftarrow$  EPub.university.stuID
  EPub.university  $\leftarrow$  ABU.accredited
  ABU.accredited  $\leftarrow$  StateU
  StateU.stuID  $\leftarrow$  Alice, IEEE.member  $\leftarrow$  Alice
}
```

4 Rules for RT_0^W

- Rule 1: $A.R \leftarrow B \equiv RT_0$ $A.R \leftarrow \langle B, s \rangle \equiv RT_0^W$
- Rule 2: $A.R \leftarrow B.R_1$ $A.R \leftarrow A.R_1.R_2$
- Rule 3: $A.R \leftarrow A.R_1.R_2$ $A.R \leftarrow B_1.R_1 \cap B_2.R_2 \cap \dots \cap B_n.R_n$
- Rule 4: $A.R \leftarrow B_1.R_1 \cap B_2.R_2 \cap \dots \cap B_n.R_n$

➤ Translated in Datalog^W as:

- Rule 1 $r(A, B) :- s$
- Rule 2 $r(A, x) :- r_1(B, x)$
- Rule 3 $r(A, x) :- r_1(A, y), r_2(y, x)$
- Rule 4 $r(A, x) :- r_1(B_1, x), r_2(B_2, x), \dots, r_n(B_n, x)$

An Example

- EPub.disct \leftarrow EPub.preferred \cap EPub.brightStudent.
- EPub.disct \leftarrow EOrg.famousProf.goodRecLetter.
- EPub.preferred \leftarrow EOrg.highBudget \cap EOrg.oldCustomer.
- EPub.brightStudent \leftarrow EPub.goodUniversity.highMarks.
 - EPub.goodUniversity \leftarrow ABU.accredited.
 - EOrg.famousProf \leftarrow \langle ProfX, \langle 0.9, 0.9 $\rangle\rangle$.
 - ProfX.goodRecLetter \leftarrow \langle Alice, \langle 0.9, 0.8 $\rangle\rangle$.
 - ABU.accredited \leftarrow \langle StateU, \langle 0.9, 0.8 $\rangle\rangle$.
 - StateU.highMarks \leftarrow \langle Alice, \langle 0.8, 0.9 $\rangle\rangle$.
 - EOrg.highBudget \leftarrow \langle Alice, \langle 0.6, 0.5 $\rangle\rangle$.
 - EOrg.oldCustomer \leftarrow \langle Alice, \langle 0.7, 0.7 $\rangle\rangle$.

$$S_{trust} = \langle \langle [0, 1], [0, 1] \rangle, +_p, \times_p, \langle 0, 0 \rangle, \langle 1, 1 \rangle \rangle$$
$$\langle t_i, c_i \rangle +_p \langle t_j, c_j \rangle = \begin{cases} \langle t_i, c_i \rangle & \text{if } c_i > c_j, \\ \langle t_j, c_j \rangle & \text{if } c_i < c_j, \\ \langle \max(t_i, t_j), c_i \rangle & \text{if } c_i = c_j. \end{cases}$$

$$\langle t_i, c_i \rangle \times_p \langle t_j, c_j \rangle = \langle t_i t_j, c_i c_j \rangle$$

The Prolog Implementation

```
:- module(rtprogram,_,_).  
:-use_module(library(aggregate)).  
:-use_module(library(sort)).
```

```
times([T1,C1], [T2,C2], [T,C]) :-  
T is (T1 * T2),  
C is (C1 * C2).
```

```
plus([], MaxSoFar, MaxSoFar).
```

```
plus([[T,C]|Rest], [MT,MC], Max):-  
C > MC,  
plus(Rest, [T,C], Max).
```

```
plus([[T,C]|Rest], [MT,MC], Max):-  
C = MC, T > MT,  
plus(Rest, [T,C], Max).
```

```
plus([[T,C]|Rest], [MT,MC], Max):-  
C < MC,  
plus(Rest, [MT,MC], Max).
```

```
plus([[T,C]|Rest], [MT,MC], Max):-  
C = MC, T < MT,  
plus(Rest, [MT,MC], Max).
```

```
trust(X, Max):- findall([T,C],disct(ePub, X, [T,C]), L1), plus(L1,[0,0],Max).
```

```
disct(ePub, X, [T,C]):- preferred(ePub, X, [T1,C1]),  
brightStudent(ePub, X, [T2,C2]),  
times([T1,C1], [T2,C2], [T,C]).
```

```
disct(ePub, X, [T,C]):- famousProf(eOrg, Y, [T1,C1]),  
goodRecLetter(Y, X, [T2,C2]),  
times([T1,C1], [T2,C2], [T,C]).
```

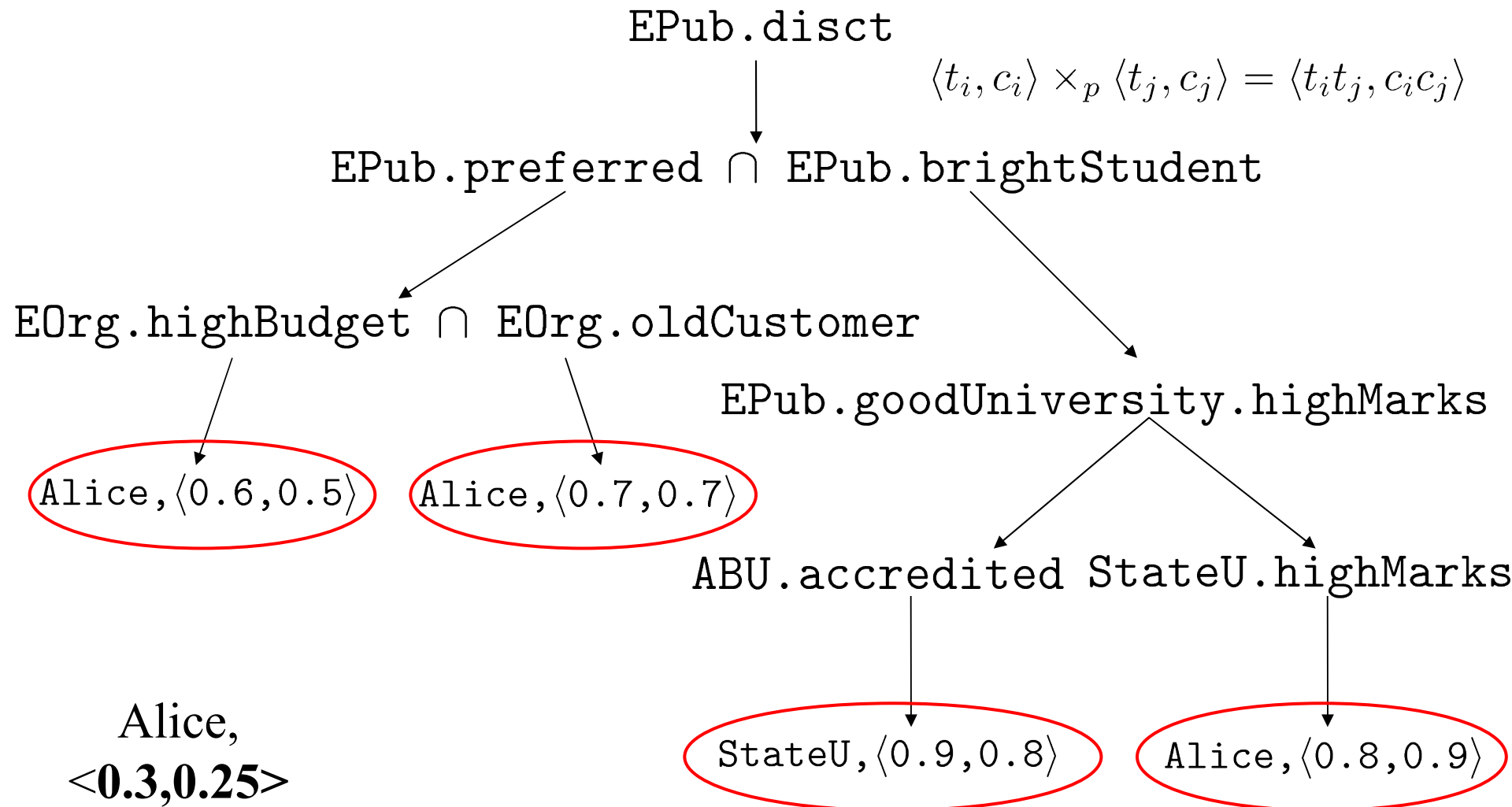
```
preferred(ePub, X, [T,C]):- highBudget(eOrg, X, [T1,C1]),  
oldCustomer(eOrg, X, [T2,C2]),  
times([T1,C1], [T2,C2], [T,C]).
```

```
brightStudent(ePub, X, [T,C]):- goodUniversity(ePub, Y, [T1,C1]),  
highMarks(Y, X, [T2,C2]), times([T1,C1], [T2,C2], [T,C]).
```

```
goodUniversity(ePub, X, [T,C]):- accredited(aBU, X, [T,C]).
```

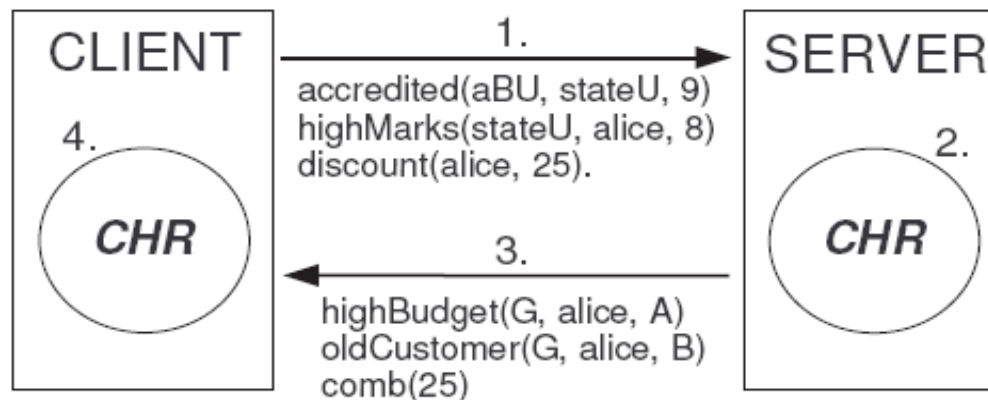
```
famousProf(eOrg, profX, [0.9,0.9]).  
goodRecLetter(profX, alice, [0.9,0.8]).  
accredited(aBU, stateU, [0.9,0.8]).  
highMarks(stateU, alice, [0.8,0.9]).  
highBudget(eOrg, alice, [0.6,0.5]).  
oldCustomer(eOrg, alice, [0.7,0.7]) .
```

The Solution (1)



Future Work

- **Abduction:** given a policy and a access request, it consists in finding the credentials/events that would grant access, i.e. the minimal set of facts that added to the policy would make the request a logical consequence.
- **The tool:** *Soft Constraint Logic Programming*, a superset of Datalog^W



Thank you for your time!

Contacts:

francesco.santini@iit.cnr.it

