Weighted Datalog and Levels of Trust *†

Stefano Bistarelli
Dipartimento di Scienze, Università "G. D'Annunzio"
Viale Pindaro 87, Pescara, Italy and
Istituto di Informatica e Telematica - CNR,
Via G. Moruzzi 1, Pisa, Italy
stefano.bistarelli@iit.cnr.it, bista@sci.unich.it

Fabio Martinelli
Istituto di Informatica e Telematica - CNR,
Via G. Moruzzi 1, Pisa, Italy
fabio.martinelli@iit.cnr.it

Francesco Santini
IMT School for Advanced Studies
Piazza San Ponziano 6, Lucca, Italy and
Istituto di Informatica e Telematica - CNR,
Via G. Moruzzi 1, Pisa, Italy
francesco.santini@iit.cnr.it, f.santini@imtlucca.it

Abstract

We extend the Datalog language (we call it Datalog^W) in order to deal with weights on ground facts and to consequently compute a feedback result for the goal satisfaction. The weights are chosen from a proper c-semiring. As a second step, in order to show the usefulness of the language, we use Datalog^W as the basis to give a uniform semantics to declarative RT^W (Trust Management) language family, in order to represent trust levels *based on c-semirings. In this way it is possible to manage a score corresponding to a preference or cost associated to the revealed credentials, instead of a plain "yes or no" authorization result. Clearly, such a solution is more informative and allows us to treat uncertainty of facts and rules application, or different preferences for the entity roles. Trust can be then derived by choosing the best chain. The approach is rather generic and could be applied to other trust management languages.*

1 Introduction and Motivations

Trust is a very slippery property that is gaining the attention of several researchers due to its crucial role for the developments of true open and pervasive systems. In particular, trust based on reputation [13] is a well established means to provide *soft security* mechanisms [18]. This kind of *social control* can be integrated with traditional *hard security* mechanisms, as passwords, authentication and access control, in order to improve the global security: soft security can be used to predict a bad behaviour (including also selfishness, and not only malice) and is more appropriate for open systems, while hard methods "deny" the presence of malicious and, once bypassed, they reveal everything [18]. Using soft mechanisms, a generic entity is instead accepted as long as its actions are not harming anyone else.

A Trust Management (TM) language is required to have the expressivity power to represent the trust-related facts of the considered dominion and a method to derive new assessments and decision starting from these base facts. For example, considering the approach to distributed authorization scenario in [6], an *authorizer* needs to decide wether to authorize a request coming from a *requester* or not. The requester presents some credentials signed by other parties to ensure the authenticity and integrity: these attributes usually represent a group membership (e.g. "I am a student"), a membership in a role within an organization or being delegates of a permission or role. In few words, the digitally signed credentials are used as an authenticated certification of the attributes of the entities that perform such a request. The *access rules* are instead used by the authorizer in order to process the requests, since they specify what conditions and what attributes are requested be authorized. For all these features, as the use of cryptography and certificates, rule-based systems can be seen instead as hard security mechanisms.

In this paper we second the view that facts of both policy

*The first and third authors are supported by the MIUR PRIN 2005-015491.

†The second author is supported by the EU projects GRIDtrust and SENSORIA.

statements and access rules are not so crisp in the real complex world. For example, each piece of information could have a confidence value associated with it and representing a reliability estimation, or a fuzzy preference level or a cost to be taken in account. The feedback final value, obtained by aggregating all the ground facts together, can be then used to improve the decision support system by basing on this preference level instead of a plain "yes or no" result (e.g, see [17, 9, 7]). In this scenario, a credential could state that the referred entity is a "student" or a "bright student" with a probability of 80% because her/his identity of student is based on what an acquaintance asserts (thus, it is not as certain as declared in IDs), or, in the second case, because the received marks need to be globally evaluated. In literature there are many examples where trust or reputation are computed by aggregating some values together [13], for example in PGP systems, or for generic trust propagation inside social networks. We think that similar quantitative measurements are useful also for trust languages, in order to have a more informative result. For this reason, we describe a weighted version of Datalog where the rules are enhanced with values taken from a proper c-semiring structure [1, 2], in order to model the preference/cost system; then, we use it as the basis to give declarative semantics to a Role-based Trust-management language according to the principles of RT_0 [16], and called here RT_0^W: the statements of RT_0^W are "soft", i.e. have a related c-semiring value. A similar improvement can be accomplished also for RT_1 [16], i.e. RT_0 extended with parameterized roles. Similar variations for RTML family languages where defined and implemented in by using different formal tools in [17]. There an initial comparison (and integration) between rule-based trust management (RTML) and reputation-based trust systems has been performed and a preliminary (ad-hoc) implementation presented in [12] for GRID systems. However, having a uniform semantics approach, as the weighted datalog we are advocating here, to model these languages could be very useful to provide a common understanding as well as a basis for systematic comparison and uniform implementation.

Indeed, there are good reasons to prefer a language that is declarative and has a formal foundation. In this sense, we are following a similar approach as done in [15] for RTML trust management languages, where Datalog with constraints have been proposed as a formal semantics for trust management languages. As it has been shown since trust is not necessarily crisp, soft Datalog could be used to give formal semantics to this languages for soft credentials. We have shown here an approach for RTML. However, it can be further extended to other credential based languages, whose semantics could be given in terms of soft Datalog. The contribution of this paper is thus to provide a formal semantics for such languages that could also bring to a uniform implementation approach, as well as to a comparison among these languages w.r.t. to others by using existing results on Datalog and soft constraints. Our systematic approach to give weights to facts and rules, contributes also towards bridging the gap between "rule-based" trust management (i.e. hard security mechanisms) and "reputation based" trust management [13] (i.e. soft security mechanisms).

In Sec. 2 we describe the background notions about trust languages and c-semirings. In Sec. 3 we present a weighted version of Datalog, i.e. Datalog^W, while Sec. 4 features the new trust languages based on Datalog^W, i.e. RT_0^W, and RT_1^W. At last, in 5 we present the final conclusions.

2 Background and Related Works

Some TM systems, such as KeyNote [5] and SPKI/SDI [10], use credentials to delegate permissions, but their delegation structure is too limited. Each credential delegates certain permissions from its issuer to its subject, acting like a capability, and therefore they do not address the distributed nature of authority in distributed systems. For instance, it is not possible to express the statement that anyone who is a student is entitled to a discount.

Datalog was originally developed as a query and rule language for deductive databases and is syntactically equivalent to a subset of the Prolog language. Several TM languages are based on Datalog, e.g., Delegation Logic [14], the RT (Role-based Trust-management) framework [16], KeyNote [5] and RBAC [19].

The RT framework is a family of Role-based Trust-management languages [16], whose most basic part is RT_0 which has been then extended to RT_1 with parameterized roles: *University.professorOf(student)* is a statement that can be used to name the professor of a student. An *entity* (or *principal*, e.g. A or B) in RT is a uniquely identified individual or process, which can issue credentials and make requests. RT assumes that an entity that issued a particular credential or a request can be determined through the use of public/private key pairs. A *role* in RT takes the form of an entity followed by a role name (e.g. R with subscripts), separated by a dot. A role defines a set of entities who are members of this role: each entity A has the authority to define who are the members of each role of the form $A.R$. Each statement defines one role to contain either an entity, another role, or certain other expressions that evaluate to a set of entities. More details will be given in Sec. 4.

An important extension that significantly enhances the expressivity of this kind of languages is presented in [15]. In that work, the authors present Datalog extended with constraints (denoted by Datalog$^{\mathcal{C}}$) in order to define access permissions over structured resources as trees.

Several approaches advocated the usage of trust levels

w.r.t. attributes, also stated directly in digital credentials. In addition to the works on the extension of RTML with weights and its relationships with other trust models as the Josang one already mentioned [17, 12], there is also the work on policy and reputation done in [7]. Here the PROTUNE policy language is extended to deal with trust and reputation levels. Also role based access control has been extended with trust levels in [9].

All these works use specific logics and approaches.

2.1 C-semirings

A c-semiring S [1, 2] (or simply semiring in the following) is a tuple $\langle S, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where S is a set with two special elements ($\mathbf{0}, \mathbf{1} \in S$) and with two operations $+$ and $\times$ that satisfy certain properties: $+$ is defined over (possibly infinite) sets of elements of S and thus is commutative, associative, idempotent, it is closed and $\mathbf{0}$ is its unit element and $\mathbf{1}$ is its absorbing element; $\times$ is closed, associative, commutative, distributes over $+$, $\mathbf{1}$ is its unit element, and $\mathbf{0}$ is its absorbing element (for the exhaustive definition, please refer to [1, 2]). The $+$ operation defines a partial order $\leq_S$ over S such that $a \leq_S b$ iff $a + b = b$; we say that $a \leq_S b$ if b represents a value *better* than a. Other properties related to the two operations are that $+$ and $\times$ are monotone on $\leq_S$, $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum, $\langle S, \leq_S\rangle$ is a complete lattice and $+$ is its lub. Finally, if $\times$ is idempotent, then $+$ distributes over $\times$, $\langle S, \leq_S\rangle$ is a complete distributive lattice and $\times$ its glb.

Varying the set S and the meaning of the $+$ and $\times$ operations, we can represent many different kinds of problems, having features like fuzziness, probability, and optimization. Moreover, in [2] the authors have shown that the cartesian product of two c-semirings is another c-semiring, and this can be fruitfully used to describe multi-criteria constraint satisfaction and optimization problems, e.g. the *path semiring* presented in Sec. 3.

3 A Weighted Extension of Datalog

Datalog is a restricted form of logic programming with variables, predicates, and constants, but without function symbols. Facts and rules are represented as Horn clauses in the generic form R_0 :- $R_1, \ldots, R_n$. A Datalog rule has the form:

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

where $R_0, \ldots, R_n$ are predicate (relation) symbols and each term $t_{i,j}$ is either a constant or a variable ($0 \leq i \leq n$ and $1 \leq j \leq k_i$). The formula $R_0(t_{0,1}, \ldots, t_{0,k_0})$ is called the head of the rule and the sequence $R_1(t_{1,1}, \ldots, t_{1,k_1}), \ldots, R_n(t_{n,1}, \ldots, t_{n,k_n})$ the body. If $n = 0$, then the body is empty and the rule is called a fact. Moreover, each program P in Datalog (i.e. a finite set of rules) must satisfy two *safety* conditions: *i)* all variables occurring in the head of a rule also have to appear in the body, and *ii)* every fact in P must be a ground fact.

```
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.
```

Table 1. A simple Datalog^W program.

We can now define our *Weighted Datalog*, or Datalog^W based on classical Datalog. While rules have the same form as in classical Datalog, a fact in Datalog^W has the form:

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

Therefore, the extension is obtained by associating to ground facts a value $s \in S$ taken from the semiring $\langle S, +, \times, \mathbf{0}, \mathbf{1}\rangle$. This value describes some properties of the fact, depending on the chosen semiring: for example, we can add together all these values by using the *Weighted* semiring $\langle R^+, min, +, \infty, 0\rangle$, trying to minimize the overall sum at the same time. Otherwise, we can find the best global preference level by using the *Fuzzy* semiring $\langle [0, 1], max, min, 0, 1\rangle$ or we can retrieve the highest resulting probability when we compose all the ground facts, by using the *Probability* semiring $\langle [0, 1], max, \times, 0, 1\rangle$.

Table 1 shows an example of Datalog^W program, for which we suppose to use the *Weighted* semiring. The intuitive meaning of a semiring value like 3 associated to the atom $r(a)$ (in Tab. 1) is that $r(a)$ costs 3 units. Thus the set R^+ contains all possible costs, and the choice of the two operations min and $+$ implies that we intend to minimize the sum of the costs. This gives us the possibility to select the atom instantiation which gives the minimum cost overall. Given a goal like $s(x)$ to this program, the operational semantics collects both a substitution for x (in this case, $x = a$) and also a semiring value (in this case, 2) which represents the minimum cost among the costs for all derivations for $s(x)$. To find one of these solutions, it starts from the goal and uses the clauses as usual in logic programming, except that at each step two items are accumulated and combined with the current state: a substitution and a semiring value (both provided by the used clause). The combination of these two items with what is contained in the current goal is done via the usual combination of substitutions (for the substitution part) and via the multiplicative operation of the semiring (for the semiring value part), which in this exam-

ple is the arithmetic $+$. Thus, in the example of goal $s(X)$, we get two possible solutions, both with substitution $X = a$ but with two different semiring values: 2 and 3. Then, the combination of such two solutions via the min operation give us the semiring value 2.

To compute trust, in Sec. 4.1 we will use the *path semiring* [21]: $S_{trust} = \langle\langle[0,1],[0,1]\rangle, +_p, \times_p, \langle 0,0\rangle, \langle 1,1\rangle\rangle$, where

$$\langle t_i, c_i\rangle +_p \langle t_j, c_j\rangle = \begin{cases} \langle t_i, c_i\rangle & \text{if } c_i > c_j, \\ \langle t_j, c_j\rangle & \text{if } c_i < c_j, \\ \langle max(t_i, t_j), c_i\rangle & \text{if } c_i = c_j. \end{cases}$$

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

In this case, trust information is represented by a couple of values $\langle t, c\rangle$: the second component represents a trust value in the range $[0, 1]$, while the first component represents the accuracy of the trust value assignment (i.e. a *confidence* value), and it is still in the range $[0, 1]$. This parameter can be assumed as a *quality* of the opinion represented instead by the trust value; for example, a high confidence could mean that the trustor has interacted with the target for a long time and then the correlated trust value is estimated with precision.

3.1 Finite Computation Time

Being the Datalog^W language a subset of the *Soft Constraint Logic Programming* language [4] with no functions, we can can use the results in [4] to prove that, considering a fixed Datalog^W program, the time for computing the value of any goal for this program is finite and bounded by a constant. The reason is that we just have to consider a finite subclass of refutations (i.e. *simple refutations*) with a bounded length. After having considered all these refutations up to that bounded length, we have finished computing the semiring value of the given goal. Given a refutation tree, a path from the root to a leaf is called *simple* if all its nodes have different labels up to variable renaming. A refutation is a simple refutation if all paths from the root to a leaf in its refutation tree are simple.

Theorem 1 (Finite Set of Simple Refutations) . *Given a Datalog^W program P and a goal C, consider the set $SR(C)$ of simple refutations starting from C and building the empty substitution. Then $SR(C)$ is finite. Moreover, each refutation in $SR(C)$ has length at most N, where $N = \sum_{i=1}^{c} b_i$, c is the number of all clauses with head instantiated in all possible ways, and b is the greatest number of atoms in the body of a clause in program P.*

This proof of the theorem is identical to the one given in [4]. Therefore, solving a goal C takes $O(SR(C) \cdot \sum_{i=1}^{c} b_i)$ steps. Moreover, other finite complexity results for a weighted Datalog language are shown in [20].

4 Extending the RT Family with Datalog^W

We describe four kinds of credentials for defining roles in a TM language family, here called RT^W, which is based on Datalog^W (see Sec. 3). This family uniformly extends the classical RT family [16] by associating a weight, or better, a semiring value to the basic role definition. Therefore, all the following credentials must be parameterized with a chosen $\langle S, +, \times, \mathbf{0}, \mathbf{1}\rangle$ semiring in order represent preference/cost or fuzzy information associated to the statements. For every following RT_0^W credential, we describe how it can be translated in a corresponding Datalog^W rule. Then we will suggest how to extend RT_0^W with parameterized roles, obtaining the RT_1^W language.

Rule 1 $A.R \longleftarrow \langle B, s\rangle$ where A and B are (possibly the same) entities, and R is a role name. This means that A defines B to be a member of A's R role. This statement can be translated to Datalog^W with the rule $r(A, B)$:- s, where s is the semiring value associated with the related ground fact, i.e. $s \in S$.

Rule 2 $A.R \longleftarrow B.R_1$ This statement means that A defines its R role to include (all members of) B's R_1 role. The corresponding Datalog^W rule is $r(A, x)$:- $r_1(B, x)$.

Rule 3 $A.R \longleftarrow A.R_1.R_2$, where $A.R_1.R_2$ is defined as *linked role* [16] and it means that A defines its R role to include (the members of) every role $B.R_2$ in which B is a member of $A.R_1$ role. The mapping to Datalog^W is $r(A, x)$:-$r_1(A, y), r_2(y, x)$.

Rule 4 $A.R \longleftarrow B_1.R_1 \cap B_2.R_2 \cap \cdots \cap B_n.R_n$. In this way, A defines its R role to include the intersection of the n roles. It can be translated to Datalog^W with $r(A, x)$:-$r_1(B_1, x), r_2(B_2, x), \ldots, r_n(B_n, x)$.

The semantics of a program using these rules will find the best credential chain according to the $+$ operator of the chosen semiring, which defines a partial order $\leq_S$. Notice that only the basic role definition statement (i.e. **Rule 1**) is enhanced with the semiring value $s \in S$, since the other three rules are used to include one role into another or to obtain the intersection of different roles.

Notice that having a semiring value associated only with ground facts does not prevent us from giving a weight also to rules. This can be accomplished by slightly changing the syntax of the credentials used to compose the roles together (i.e. rules 2-3-4), by associating a semiring value also to them. Then, in the Datalog translation, a new ground fact can be added in the body of the rule, whose weight models the use of that specific rule. For example, **Rule 2** becomes $A.R \longleftarrow B.R_1, s$ (where s is a value taken from the same S semiring set), and its Datalog translation

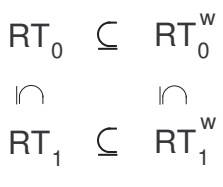

Figure 1. A hierarchy of RT^W languages, compared with the classical RT one.

is $r(A, x)$:-$r(B, x), rule\_weight$, where $rule\_weight$:- s is the ground fact that gives a weight to the rule. Clearly, nothing changes from the computational point of view (see Sec 3.1).

It is easy to extend this language in order to enhance it with parameterized roles, thus obtaining a RT_1^W language following the hierarchy presented in [16]. This parametrization can be used to represent relationships among entities, e.g. *University.professorOf(student)* to name the professor of a student, but also to represent attributes that have fields, e.g. the number of exams or the enrollment academic year and so on. With respect to the previous four rules, in RT_1^W the *head* of a credential has the form $A.R(h_1, \ldots, h_n)$, in which A is an entity, and $R(h_1, \ldots, h_n)$ is a role name (R is a role identifier). For each $i \in 1 \ldots n$, h_i is a data term having the type of the ith parameter of R. For example, **Rule 1** can be rewritten in RT_1^W as $A.R(h_1, \ldots, h_n) \longleftarrow \langle B, s \rangle$, and mapped to Datalog^W as $r(A, B, h_1, \ldots, h_n)$:- s. Our intention is to extend the RT^W family according to the guidelines explained in [16] (see Sec. 5).

Since Datalog is a subset of first-order logic, the semantics of a TM language based on it is declarative and unambiguous. While The $\times$ operator of the semiring is used to compose the preference/cost values associated to the statements, the $+$ is used to let the framework select the best derivation chain with more chances to authorize the requester (among all the credentials revealed by her/him).

Theorem 2 (Language Family Inclusion) *For each S set of statements in the RT_0 or RT_1 language, we can find a corresponding S^W set of statements respectively represented in RT_0^W or RT_1^W, and whose semantics is the same.*

In Fig. 1 we show the theorem result, i.e. the horizontal inclusions; the vertical ones are explained in [16] (for RT) and in this paper (for RT^W). As a suggestion, it can be proved by using the *Boolean* semiring $\langle \{0, 1\}, \vee, \wedge, 0, 1 \rangle$ and by assigning a weight of 1 (i.e. the *true* value) to all the ground facts. In this way we obtain a set of crisp statements and the semantics returns all the possible derivation paths, as the corresponding RT set of statements would do.

4.1 Some Examples with Levels of Trust

We can start by adding levels to the classical RT_0 example presented in many RT related papers (e.g. [16]). To solve the example in Tab. 2, we use a *Fuzzy* semiring $\langle [0, 1], max, min, 0, 1 \rangle$, where the elements in $[0, 1]$ represents the truth degree connected to a credential and evaluated by the entity which signs and issues it: for example, `StateU.highMarks ⟵ ⟨ Alice, 0.8 ⟩` in Tab. 2 certifies that Alice has obtained a good number of high marks (since the value is 0.8) for the exams completed at the StateU university (the credential is issued by StateU).

```
EPub.disct ⟵ EPub.preferred ∩
      EPub.brightStudent.
EPub.preferred ⟵ EOrg.highBudget ∩
      EOrg.oldCustomer.
    EPub.brightStudent ⟵
  EPub.goodUniversity.highMarks.
EPub.goodUniversity ⟵ ABU.accredited.
ABU.accredited ⟵ ⟨ StateU, 0.9 ⟩.
StateU.highMarks ⟵ ⟨ Alice, 0.8 ⟩.
EOrg.highBudget ⟵ ⟨ Alice, 0.6 ⟩.
EOrg.oldCustomer ⟵ ⟨ Alice, 0.7 ⟩.
```

Table 2. An example in RT_0^W, with fuzzy values associated to the credentials.

```
EPub.disct ⟵ EPub.preferred ∩
      EPub.brightStudent.
        EPub.disct ⟵
  EOrg.famousProf.goodRecLetter.
EPub.preferred ⟵ EOrg.highBudget ∩
      EOrg.oldCustomer.
    EPub.brightStudent ⟵
  EPub.goodUniversity.highMarks.
EPub.goodUniversity ⟵ ABU.accredited.
EOrg.famousProf ⟵ ⟨ ProfX, ⟨ 0.9, 0.9 ⟩⟩.
ProfX.goodRecLetter ⟵ ⟨Alice, ⟨ 0.9, 0.8 ⟩⟩.
ABU.accredited ⟵ ⟨ StateU, ⟨ 0.9, 0.8 ⟩⟩.
StateU.highMarks ⟵ ⟨ Alice, ⟨ 0.8, 0.9 ⟩⟩.
EOrg.highBudget ⟵ ⟨ Alice, ⟨ 0.6, 0.5 ⟩⟩.
EOrg.oldCustomer ⟵ ⟨ Alice, ⟨ 0.7, 0.7⟩⟩.
```

Table 3. An extension of the example in Tab. 2, using the *path semiring*.

The example in Tab. 2 describes a fictitious Web publishing service, *EPub*, which offers a discount to anyone who is both a preferred customer and a bright student. EPub delegates the authority over the identification of preferred customers to its parent organization, *EOrg*. In order to be

evaluated as a preferred customer, EOrg must issues two different types of credentials stating that the customer is not new (i.e. *EOrg.oldCustomer*) and has already spent some money in the past (i.e. *EOrg.highBudget*). EOrg assigns a fuzzy value to both these two credentials to quantify its evaluation. EPub delegates the authority over the identification of bright students to the entities that are accredited universities. To identify such universities, EPub accepts accrediting credentials issued by the fictitious *Accrediting Board for Universities* (*ABU*). ABU evaluates a university with a fuzzy score and each university evaluates its enrolled students. A student is bright if she/he is both enrolled in a good university and has high marks. The final fuzzy score, obtained by composing together all the values of the used credentials, can be compared with a threshold to authorize the discount: e.g. only entities whose set of credentials produced a score greater than 0.7 are authorized. Otherwise, the final fuzzy result can be used to derive a proportional discount amount: for example a score of 0.8 could authorize a discount that is twice the discount allowed with a score of 0.4. The following credentials prove that Alice is eligible for the discount with a score of 0.6, determined by the fact that she has not a very high budget spent at EOrg (i.e. her $EOrg.highBudget$ credential has a value of 0.6).

In Tab. 3 we extend the example of Tab. 2 in order to represent also a case where the authorization can be accomplished by following different derivation chains. For example, a customer could be allowed to have a discount even if she/he presents a good recommendation letter written by a famous professor (i.e. *EPub.famousProf.goodRecLetter*). In Tab. 3 we use the *path semiring* presented in Sec. 3, thus a semiring value consists in a couple of trust/confidence feedbacks. The best derivation chain corresponds to the criteria defined by the $+_p$ (i.e. confidence is more important).

4.2 An implementation in CIAO Prolog

In this Section we implement and solve the authorization problem described in Tab. 3. Since Datalog is a subset of Prolog, we represent the credentials and the access rules with a program in *CIAO Prolog* [8], a system that offers a complete Prolog system supporting ISO-Prolog , but, at the same time its modular design allows both restricting and extending the basic language (with predicate abstractions, constraints, objects, concurrency, etc).

In Fig. 2 we can see the statements in Tab. 3 translated into CIAO Prolog clauses. The *times* and *plus* clauses are used to model the $\times_p$ and $+_p$ operators of the *path semiring* presented in Sec. 3. Querying the interpreter with $trust(alice, [T, C])$, we obtain the best refutation tree (according to $+_p$) among the two possible ones, with $[T, C]$ respectively instantiated to $\langle 0.81, 0.72\rangle$ and $\langle 0.3, 0.25\rangle$. The best choice is by using

```
:- module(rtprogram,_,_).
:-use_module(library(aggregates)).
:-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,
plus(Rest, [MT,MC], 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, 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]) .
```

Figure 2. The CIAO Prolog program representing the statements in Tab. 3

the *EOrg.famousProf.goodRecLetter*, i.e. $\langle 0.81, 0.72\rangle$.

5 Conclusions and Future Work

We have proposed a weighted extension of Datalog and a trust language family based on it. These languages can be used to deal with vague and imprecise security policies or credentials, and preference or costs associated to each credential. In practice, we can manage and combine together different levels of truth, preference or costs associated to the statements and finally have a single feedback value on which to authorize a trust request. We have used weighted datalog to give semantics to an extension of the RT family of languages [16] and we we have shown that classical RT_0 and RT_1 are respectively included in our RT_0^W and RT_1^W. It is worthy to notice that our extension is completely orthogonal w.r.t. the RT extension proposed in [15], i.e. RT^C, where the supporting Datalog^C language allows

first-order formulas in tractable constraint domains. The constraints are introduced to represent the access permissions over structured resources, e.g., tree domains and range domains. Our aim is instead the representation of trust levels modelling cost/preference or fuzziness of credentials.

As a first future work, we will extend the RT^W family by considering the other languages proposed in [16], as RT_2 that provides the notion of o-sets (grouping logically related objects together), RT^T that is improved with manifold roles and role-product operators, or RT^D, that provides delegation of role activations. On more future improvement could be to leave to the programmer the opportunity to take more decisions inside the rules, for example based on the current aggregated semiring value (the process is called *reification* of the values, i.e. make them visible to the programmer); from its evaluation, some rules could be enabled and others could be ignored, influencing the derivation process and the final result. We plan to investigate the complexity of tractable soft constraints classes [11] in order to cast them in a tractable Datalog-based language. Therefore, we want to extend also the RT^C language [15] (based on Datalog enhanced with crisp constraints) in its RT^W_C soft version, as done in this paper for RT_0 and RT_1.

We want also to study the feasibility of using *Soft Concurrent Constraint Programming*-like languages [1, 3] to represent the same authorization process: the soft constraints, added to the σ constraint store by the participating parties, will represent the credentials and the access rules proposed for the request, each statement with its own preference value. In fact, soft constraints [1] extend classical constraints to represent multiple consistency levels, and thus provide a way to express preferences, fuzziness, and uncertainty. Then, a query can be modelled with a simple $ask(c) \longrightarrow^a$ action [1, 3], where the c constraint represents the fact that we want to check if it is implied by the store: the ask succeeds (i.e. the request is accepted) if the store σ entails the constraint c and also if the store is "consistent enough" w.r.t. the threshold a on the transition arrow, which represents the minimum requested consistency (or preference) level to be satisfied by the query.

Acknowledgments. We would like to thank the anonymous referees for their helpful comments.

References

[1] S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *Lecture Notes in Computer Science*. Springer, 2004.

[2] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, 1997.

[3] S. Bistarelli, U. Montanari, and F. Rossi. Soft concurrent constraint programming. *ACM Trans. Comput. Logic*, 7(3):563–589, 2006.

[4] S. Bistarelli and F. Rossi. Semiring-based constraint logic programming: syntax and semantics. *ACM Trans. Program. Lang. Syst.*, 23(1):1–29, 2001.

[5] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The keynote trust-management system, rfc2704, 1999.

[6] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *SP '96: Proc. of Security and Privacy*, page 164. IEEE Computer Society, 1996.

[7] P. Bonatti, C. Duma, D. Olmedilla, and N. Shahmehri. An integration of reputation-based and policy-based trust management. In *Semantic Web Policy Workshop*, 2005.

[8] F. Bueno, D. Cabeza, M. Carro, M. Hermenegildo, P. López-García, and G. Puebla. The CIAO prolog system: reference manual. Technical Report CLIP3/97.1, School of Computer Science, Technical University of Madrid (UPM), 1997.

[9] S. Chakraborty and I. Ray. Trustbac: integrating trust relationships into the rbac model for access control in open systems. In *SACMAT '06: Proc. of Access control models and technologies*, pages 49–58. ACM, 2006.

[10] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *J. Comput. Secur.*, 9(4):285–322, 2001.

[11] D. A. Cohen, M. C. Cooper, P. G. Jeavons, and A. A. Krokhin. The complexity of soft constraint satisfaction. *Artif. Intell.*, 170(11):983–1016, 2006.

[12] M. Colombo, F. Martinelli, P. Mori, M. Petrocchi, and A. Vaccarelli. Fine grained access control with trust and reputation management for globus. In *OTM Conferences (2)*, pages 1505–1515, 2007.

[13] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[14] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.

[15] N. Li and J. C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *PADL '03: Proc. of Practical Aspects of Declarative Languages*, pages 58–73. Springer-Verlag, 2003.

[16] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *SP '02: Proc. of Security and Privacy*, page 114. IEEE Computer Society, 2002.

[17] F. Martinelli and M. Petrocchi. A uniform approach for the modeling of security and trust on protocols and services. In *ICS '06: International Workshop on Computer Security*, 2006.

[18] L. Rasmusson and S. Jansson. Simulated social control for secure internet commerce. In *NSPW '96: Proc. of New security paradigms*, pages 18–25. ACM, 1996.

[19] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[20] T. Stüber and H. Vogler. Weighted monadic datalog. In *Workshop on Theorietag Automaten und Formale Sprachen*, pages 126–130. Informal Proc., 2007.

[21] G. Theodorakopoulos and J. S. Baras. Trust evaluation in ad-hoc networks. In *WiSe '04: Workshop of Wireless security*, pages 1–10. ACM, 2004.