In *Proc. of the 8th Symposium on Principles and Practice of Declarative Programming (PPDP'06)*, pages - . © ACM.

# Inferring Dynamic Credentials for Rôle-based Trust Management

Daniele Gorla
Dip. di Informatica,
Univ. di Roma "La Sapienza"
gorla@di.uniroma1.it

Matthew Hennessy
Dep. of Informatics, Univ. of Sussex
matthewh@susx.ac.uk

Vladimiro Sassone
ECS, Univ. of Southampton

## Abstract

The topic of this paper is the rôle-based trust-management language $RT_0$, a formalism inspired by logic programming that handles trust in large scale, decentralised systems. We provide a purely operational semantics for the language in which credentials can be established using a simple set of inference rules. We then extend $RT_0$ to include time validity and boolean guards that control the availability of credentials. In such an extended framework, credentials are *conditional* on the availability of supporting credentials in the execution context. In addition to a set-theoretic and a logic-programming semantics, we develop for the extended language a series of increasingly powerful inference systems for establishing these conditional credentials. By means of simple but realistic examples, we demonstrate the expressiveness and usability of our language, warranting its integration into existing trust-management tools.

***Categories and Subject Descriptors*** D.3.1 [*Programming Languages*]: Formal Definitions and Theory; F.3.2 [*Logics and Meaning of Programs*]: Semantics of Programming Languages; D.4.6 [*Operating Systems*]: Security and Protection

***General Terms*** Languages, Security.

***Keywords*** trust-management, rôle-based access control, inference systems, logic programming with negation.

## 1. Introduction

One of the current challenges in computer science is the development of theoretically-based and practically-implementable approaches to access control and authorisation in large-scale, distributed systems. Such problems arise, for example, when independent users or organisations collaborate to achieve common goals, since collaborations are highly dynamic and usually heterogeneous: membership, resources and policies vary in time and are usually locally controlled by each collaborating principal; normally no form of centralisation exists.

*Trust-management* [6] is an approach to distributed access control where decisions are based on policy statements made by multiple principals. A key aspect of trust-management is *delegation*: a principal may transfer limited authority on one or more resources to other principals. Usually, this is done by means of *credentials*, i.e. pieces of information, passed from one principal to another and used to establish the sending principal's access rights. A chain of one or more credentials acts as a capability, granting permissions to principals.

Traditionally, access control takes decisions by relying on the identity of the resource requester. Unfortunately, when resource owner and requester are unknown to each other, such a form of access control does not work. For this reason, in [20] trust-management has been integrated with *rôle-based access control* (RBAC) [27]. RBAC is a policy-neutral access control technology, whose flexibility and expressiveness arise from the notion of *rôle*, interposed in the assignment of permissions to users. Users are authorised to use the permissions assigned to the rôles they belong to; thus, in contrast with traditional access control mechanisms, RBAC regulates access to resources on the basis of the activities users execute in the system, and not on their identity.

***A Rôle-based Trust-Management Language.*** $RT$ is a family of rôle-based trust-management languages able to express statements on policies in a succinct and intuitive way. It is inspired by trust-management languages such as SPKI/SDSI [9, 8] and includes basic operations to perform complex forms of delegation. $RT_0$ is the most basic language of the family; it is the "core" language, in that it only includes the key aspects of $RT$ and ignores programming features, such as data types or constraints, whose only aim is to make the task of programming more flexible.

We present the key features of $RT_0$ via the following simple example.

EXAMPLE 1. An auditor can inspect an enterprise only if he is a member of a society authorised by the government. In the UK, auditing societies are chosen among those which are legally registered and fair. Assuming that *B* is a member of a society *BSoc* that is both legally registered and fair for the UK standards, then $B$ can become an auditor for an enterprise *Ent*. This scenario can be modelled by the following $RT_0$-credentials:

$$\mathit{Ent.auditor} \leftarrow \mathit{UK.auditor} \quad (1)$$
$$\mathit{UK.auditor} \leftarrow \mathit{UK.authSoc.member} \quad (2)$$
$$\mathit{UK.authSoc} \leftarrow \mathit{UK.legalSoc} \sqcap \mathit{UK.fairSoc} \quad (3)$$
$$\mathit{UK.legalSoc} \leftarrow \mathit{BSoc} \quad (4)$$
$$\mathit{UK.fairSoc} \leftarrow \mathit{BSoc} \quad (5)$$
$$\mathit{BSoc.member} \leftarrow \mathit{B} \quad (6)$$

□

The basic statement of $RT_0$ takes the form $A.r \leftarrow B$ (cf. (4), (5) and (6) in Example 1): it states that the principal $B$ belongs to the rôle $r$ governed by principal $A$. The basic form of delegation is expressed in $RT_0$ by means of a credential $A.r \leftarrow B.s$ (cf. (1) in Example 1), stating that all members of rôle $s$ governed by $B$ also belong to rôle $r$ governed by $A$. Delegation can also be partial, viz.

in $A.r \leftarrow B.s \sqcap C.t$ (cf. (3) above): in this case, only the members of both $B$'s rôle $s$ and of $C$'s rôle $t$ belong to rôle $r$ governed by $A$. Finally, delegation itself can also be delegated, viz. in $A.r \leftarrow B.s.t$ (cf. (2)): in this case, all members of $C$'s rôle $t$ also belong to rôle $r$ governed by $A$, for every $C$ belonging to $B$'s rôle $s$.

Two equivalent semantics of $RT_0$ are presented in [22, 20]: in the first paper, the semantics of a set of $RT_0$-credentials is given via a set-theoretic interpretation; this resembles a denotational semantics and is explicitly based on a fixpoint construction. In the second paper, the semantics is given indirectly: $RT_0$ credentials are translated into a logic program and their semantics is obtained as the minimal Herbrand model of the translation. The main intention of this second approach is to provide an implementation of credential resolution.

The first contribution of our paper is a purely operational interpretation of $RT_0$ credentials; we give a simple set of inference rules for deriving credentials from a set of $RT_0$ statements. This inference system is an explicit formalisation of the intuitive meaning of $RT_0$ statements and provides a convenient way of working with them. The judgements of the inference system take the form $\mathcal{P} \succ c$, where $\mathcal{P}$ is a set of $RT_0$ credentials and $c$ is a $RT_0$-credential. Thus, in the context of Example 1, we will be able to derive that $\{(1), (2), (3), (4), (5), (6)\} \succ$ *Ent.auditor* $\leftarrow$ *B*.

$RT_0$ is intended as a model for real-life trust management; it is therefore desirable to extend it with realistic features, while preserving its nature of "core" formalism. The main contribution of this paper is in this direction: we add time validity and (possibly negative) boolean conditions to limit the use of $RT_0$-credentials. We call the resulting formalism *context-dependent credentials* (CDCs, for short), since the availability of a $RT_0$-credential now depends on the context where it is exhibited, that is the time of evaluation and the information inferable from other credentials available in the execution context.

***Context-dependent credentials, informally.*** Example 1 can be made more realistic by including *timing information*; indeed, several authors advocate credentials that are valid only for some fixed periods of time (see, e.g., [28, 23, 26]). In our auditing scenario, it is quite natural to assume that *B* is a member of *BSoc* only for a fixed period of time, say $v_2$; moreover, UK's fairness certificates are usually valid only for a period of time, say $v_1$; finally, *BSoc* becomes a legal society only after its registration that happens, say, at time $\tau$. Thus, credentials (4), (5) and (6) should be generalised to

$$\textit{UK.legalSoc} \leftarrow \textit{BSoc} \textbf{ in } [\tau, +\infty) \tag{7}$$

$$\textit{UK.fairSoc} \leftarrow \textit{BSoc} \textbf{ in } v_1 \tag{8}$$

$$\textit{BSoc.member} \leftarrow \textit{B} \textbf{ in } v_2 \tag{9}$$

stating that (4), (5) and (6) are only available after $\tau$, during $v_1$ and during $v_2$, respectively. On the other hand, credentials (1), (2) and (3) are always valid, as they express some time-independent facts. Now, by using (1), (2), (3), (7), (8) and (9), we want to be able to derive that *B* can be an auditor for *Ent* during all of the period

$$v_1 \cap v_2 \cap [\tau, +\infty). \tag{10}$$

Another powerful feature which would be useful to model more realistic policies is the ability to parameterise the validity of a credential on the the availability/non-availability of other credentials in the execution context. This can be useful to enforce, e.g., *mutual exclusion* or *separation of duties*. These principles are easy to implement in RBAC [12] and thus it is then desirable to have them also in $RT$.

For example, in Example 1 it would be very natural to require that the auditor to inspect *Ent* is not one of *Ent*'s employees. Thus, credential (1) should be replaced by

$$\begin{array}{l}\textbf{if } \textit{B} \in \textit{UK.auditor} \wedge \textit{B} \notin \textit{Ent.employees} \\ \textbf{then } \textit{Ent.auditor} \leftarrow \textit{B}\end{array} \tag{11}$$

Now, from (2), (3), (7), (8), (9) and (11), we will be able to infer that *B* can inspect *Ent* during all $v_1 \cap v_2 \cap [\tau, +\infty)$ only if the execution context does not provide any credential proving that *B* is an employee of *Ent* in such a period, i.e., only if it is *not* possible to infer from the context the $RT_0$-credential

$$\textit{Ent.employees} \leftarrow \textit{B} \tag{12}$$

The presence of negative premises makes the theory of CDCs considerably more complex than that of $RT_0$. We avoid potential inconsistencies by following the well-known path put forward by the *stable model semantics* [14]; using this technique we extend the standard semantics of $RT_0$ by providing both a set-theoretic semantics for CDCs and a translation into logic programming. We then adapt our inference system for $RT_0$ to CDCs; as with $RT_0$, we believe that this approach gives a very intuitive interpretation to CDCs. However, since the inference system has negative premises, we have to be careful to avoid the same unfoundedness problems present in the logic programming and in the set theoretic approach. Following the ideas in [7], we define an inference system allowing negative premises by following the construction of stable models for general logic programs (i.e., logic programs with negative atoms). Then because the same construction is used in all the three semantics for CDCs, we can claim and prove that all these approaches do coincide.

The extended inference system now relies on judgements of the form $\aleph \vdash_\tau c$, where $\aleph$ is a set of CDCs and $c$ is a $RT_0$-credential. Intuitively, it means that $c$ can be inferred, at time $\tau$, from $\aleph$, meaning that $\aleph$ has enough information to satisfy all the positive guards of the CDCs used in the inference and none of their negative guards. For example, we will have that $\{(2), (3), (7), (8), (9), (11)\} \vdash_\tau$ *Ent.auditor* $\leftarrow$ *B*, for every $\tau \in v_1 \cap v_2 \cap [\tau, +\infty)$. On the other hand, it will not be possible to derive *Ent.auditor* $\leftarrow$ *B* from $\{(2), (3), (7), (8), (9), (11), (12)\}$ at any time.

As a final contribution, we then enhance the judgements $\aleph \vdash_\tau c$ in two ways. Firstly, we define the judgement $\aleph \Vdash_v c$, stating that $c$ can be inferred from $\aleph$ at any time $\tau \in v$; in other words, we calculate the maximal set of times in which the inference $\aleph \vdash_\tau c$ holds. Secondly, we define the judgement $\aleph \Vdash^\phi_\tau c$, stating that $c$ can be inferred from $\aleph$ at time $\tau$ in any context that provides enough information to satisfy the logical formula $\phi$; essentially $\phi$ describes what can be inferred from the context and what cannot. This is useful in distributed systems, where it would be unreasonable to assume that all users know at all times the credentials currently available. Thus, when a user wants to construct a certificate chain, he can rely on the credentials he owns (typically, those created by himself and those granted to him by someone else); these are, however, rarely sufficient to complete the chain. So, $\aleph \Vdash^\phi_\tau c$ could be used to describe the contexts in which the user can obtain the privilege desired.

EXAMPLE 2. Let us clarify this last point by means of a simple example. The access to *Alice*'s mail can be described by the following CDC:

$$\begin{array}{l}\textbf{if } \textit{Alice} \notin \textit{Ent.active} \\ \textbf{then } \textit{Alice.readMail} \leftarrow \textit{Ent.secr}\end{array} \tag{13}$$

It states that whenever *Alice* is not an active worker of the enterprise (maybe because she is on vacation or because she is no longer working for it), the enterprise's secretary can read *Alice*'s mail. Clearly, *Alice*'s status is described by the execution context where the credential is used. Hence, the credential *Alice.readMail* $\leftarrow$ *Ent.secr* is available if the execution context provides a set of

credentials where *Ent.active* $\leftarrow$ *Alice* cannot be inferred. For example, imagine that the execution context provides the credential

$$\textit{Ent.active} \leftarrow \textit{Alice}\ \textbf{in}\ v \tag{14}$$

where $v$ is a proper time validity. Then, the credential set $\{(13), (14)\}$ makes available the credential *Alice.readMail* $\leftarrow$ *Ent.secr* at any time not included in $v$. Now, what are the credentials a user *Bob* should provide to read *Alice*'s mail? In other words, we need to find a set of CDCs that, together with (13) and (14), allows *Bob* to derive the following goal:

$$\textit{Alice.readMail} \leftarrow \textit{Bob} \tag{15}$$

A possibility would be the context

$$\textit{Ent.secr} \leftarrow \textit{Bob} \quad \text{and} \quad \tau \in (-\infty, +\infty) \setminus v \tag{16}$$

i.e., *Bob* can read *Alice*'s mail provided that *Bob* is one of the enterprise's secretaries while *Alice* is not an active worker of the enterprise (by (14), $Alice$ is an active worker only during $v$; thus, she is not an active worker during $(-\infty, +\infty) \setminus v$). Notice that context (16) works well only if $v \subset (-\infty, +\infty)$; otherwise credential (13) cannot be used because credential (14) states that *Alice* is a permanently active worker. In this latter case, we can only rely on a context providing a credential set that permits to directly infer *Alice.readMail* $\leftarrow$ *Bob*. □

Finally, we also discuss the close correspondence between our enhanced inference systems for inferring CDCs and *abductive logic programming* [16]. This is a variant of logic programming which, given a logic program (corresponding, in our case, to a set of CDCs) and a goal (corresponding, in our case, to the privilege desired), returns the minimal set of facts (corresponding, in our case, to the execution context) enabling the derivation of the goal.

***Structure of the paper.*** The paper is structured as follows. In Section 2 we recall some elements from the theory of logic programming, since, as we have already mentioned, several constructions are taken from this field. In Section 3 we briefly recall $RT_0$ and its two known semantics, namely the set-theoretic and the logic-programming one. Then, we present our inference system and state its equivalence with respect to the previous two semantics. In Section 4 we present CDCs and their set-theoretic, logic-programming and inference system semantics. In Section 5 we present the enhanced inference systems for CDCs and their abductive logic programming counterpart. Finally, in Section 6 we conclude the paper by touching upon related work. Due to space limitations, some proofs are omitted and can be found in a technical report [15].

## 2. Elements of Logic Programming

We briefly recall the main definitions from the field of logic programming used in the paper. For the sake of simplicity, several definitions will be tailored to our needs; for a full presentation, see [2, 4].

We assume three numerable and pair-wise disjoint sets: *variables*, ranged over by $\xi, \zeta, \ldots$; *constants*, ranged over by $A, B, C, \ldots$; and *(binary) relations*, ranged over by $r, s, t, \ldots$. *Atoms*, ranged over by $\alpha$, are triples consisting of a relation and two variables or constants; for example, $r(A, \xi)$ is an atom. Atoms are called *ground* whenever they do not contain any variable. *Literals*, ranged over by $\lambda$, are either atoms or negated atoms. *General logic clauses* take the form

$$\alpha :\!- \lambda_1, \cdots, \lambda_k$$

for $k \geq 0$; when $k = 0$, we use the term *unit clause*. All variables occurring in a clause are meant to be universally quantified and a comma between two literals stands for their conjunction. Thus, for example,

$$r(A, \xi) :\!- s(B, \xi), \neg t(D, C)$$

denotes the following first-order logic formula

$$\forall \xi\ .\ \neg t(D, C) \wedge s(B, \xi) \Rightarrow r(A, \xi)$$

*General logic programs*, ranged over by $\mathtt{P}$, are finite sets of general logic clauses. A *logic program* is a finite set of clauses that do not contain negated atoms.

The semantics of a general logic program is given by its *models*. A model is a triple formed by a nonempty set $\mathtt{U}$, called *universe*, and two functions that associate an element of the universe to each constant of the language and a subset of $\mathtt{U} \times \mathtt{U}$ to each relation of the language, respectively; moreover, such associations must respect the logical constraints imposed by the program clauses. The canonical model of a program is usually chosen among its *Herbrand models*. These are models whose universe is the set of constants of the language and whose constants are interpreted by the identity function. A Herbrand model is completely determined by the ground atoms that are true in it; we shall usually identify it with the set of these atoms. A Herbrand model is *minimal* if no proper subset of it is a Herbrand model of the program.

The semantics of a logic program (without negations) $\mathtt{P}$ is given by means of its *minimal Herbrand model*, written $\Psi(\mathtt{P})$, that always exists and is unique [30]. However, for general logic programs the Herbrand model may not be unique or may not even exist at all. Problems arise in programs such as

$$\{\ r(A, B) :\!- \neg\, r(A, B)\ \} \tag{17}$$

$$\begin{array}{l}\{\ r(A, B) :\!- \neg\, s(C, D)\ , \\ \quad s(C, D) :\!- \neg\, r(A, B)\ \}\end{array} \tag{18}$$

where validity of $r(A, B)$ (possibly indirectly) relies on the non-validity of $r(A, B)$ itself. Several proposals have been appeared in literature to give a semantics to general logic programs. However, as clearly stated in [14], "researchers seem to agree that there can be no useful definition of canonical models for arbitrary programs." Among all the available proposals, one of the most general and intuitive is the *stable model semantics* [14].

DEFINITION 2.1 (Stable Model Semantics). *Let $K$ be a set of ground atoms from $\mathtt{P}$.*[1] *Then,*

1. *the logic program $\mathtt{P}|_K$ is obtained from $\mathtt{P}$ by deleting*
   - *each clause that has a negative literal $\neg\alpha$ with $\alpha \in K$, and*
   - *all negative literals in the remaining clauses;*
2. *$K$ is a* stable model *of $\mathtt{P}$ if $K = \Psi(\mathtt{P}|_K)$;*
3. *the* stable model semantics *of $\mathtt{P}$ is $\Psi(\mathtt{P}|_K)$, provided that it exists and it is the only stable model of $\mathtt{P}$.*

The stable model semantics is one of the most general semantics for general logic programs in the sense that it assigns a semantics to quite a large number of programs. Indeed, it has been proved [14, 4] that stable models subsume the iterated fixpoint semantics of stratified programs [3], the well-founded semantics [31] and the perfect models of locally stratified programs [24], while it overlaps with perfect models for programs that are not locally stratified. Moreover, it coincides with the well-supported model semantics of [11]; this is a very reasonable semantics as it only allows to infer atoms whose 'explanation' does not rely on themselves. However, stable models still leave some programs without a semantics. For example, (17) has no semantics, as it has no stable model, whereas (18) has no semantics, as it has two different stable models (viz., $\{r(A, B)\}$ and $\{s(C, D)\}$).

[1] A set of ground atoms from a general logic program $\mathtt{P}$ is any subset of $\mathtt{R} \times \mathtt{C} \times \mathtt{C}$, where $\mathtt{R}$ and $\mathtt{C}$ are the relations and the constants occurring in $\mathtt{P}$.

| | |
|---|---|
| ENTITY NAMES | $A, B, C, \cdots \in \mathcal{E}$ |
| RÔLE NAMES | $r, s, t, \cdots \in \mathcal{R}$ |
| RÔLES | $A.r, B.s, C.t, \cdots \in \mathcal{E} \times \mathcal{R}$ |
| RÔLE EXPRESSIONS | $e ::= B \mid B.s \mid B.s.t \mid B.s \sqcap C.t$ |
| CREDENTIALS | $c ::= A.r \leftarrow e$ |

**Table 1.** Syntax of $RT_0$

## 3. The Language RT$_0$

We start by briefly recalling the language $RT_0$ from [21] which, in turn, is a generalisation of SPKI/SDSI [9, 8]. We then present an inference system for $RT_0$ that captures precisely the existing semantics of the language, given by both a set-theoretic interpretation and a logic programming interpretation.

***Syntax.*** The syntax of $RT_0$ is depicted in Table 1. We assume two countable and pair-wise disjoint sets, $\mathcal{E}$ and $\mathcal{R}$, of *entity* and *rôle names*, respectively. Entity names start with (or simply are) capital letters, while rôle names start with (or simply are) lower case letters. *Rôles* are compound entities made up of an entity name and a rôle name, separated by a dot.

In $RT_0$ there are four kinds of *rôle expressions*, ranged over by $e$, that yield four kinds of *credentials*, ranged over by $c$. Intuitively, credential $A.r \leftarrow B$ means that $A$ defines $B$ to be a member of $A$'s $r$ rôle. Credential $A.r \leftarrow B.s$ means that $A$ defines its $r$ rôle to include (all members of) $B$'s $s$ rôle; this represents a *delegation* from $A$ to $B$, since $B$ may affect who is a member of the rôle $A.r$ by issuing statements. Credential $A.r \leftarrow B.s.t$ means that $A$ defines $A.r$ to include every member of $C.t$, for every $C$ that is a member of $B.s$. Finally, credential $A.r \leftarrow B.s \sqcap C.t$ means that $A$ defines $A.r$ to include every principal who is a member of both $B.s$ and $C.t$; this represents *partial delegations* from $A$ to $B$ and to $C$.

In what follows, $\mathcal{P}$ denotes a finite set of $RT_0$-credentials.

***Semantics.*** We now recall from [19] two equivalent ways of giving a semantics to $RT_0$.

DEFINITION 3.1. *The* set-theoretic semantics of $\mathcal{P}$, *denoted as* $[\![\mathcal{P}]\!]$, *is the least fixpoint (w.r.t. point-wise set inclusion) of the following sequence of functions, mapping rôles to sets of entity names:*

$$R_0 \quad \textit{is the function mapping every rôle to } \emptyset$$
$$R_{i+1} \triangleq \bigoplus_{c \in \mathcal{P}} f(R_i, c)$$

*where '$\oplus$' is the point-wise extension of a function and $f$ is a function that, given a (partial) semantics $R_i$ and a credential $A.r \leftarrow e$, returns all the entity names that should be added to $R_i(A.r)$, as governed by $e$:*

$$\begin{aligned} f(R_i, A.r \leftarrow B) &\triangleq \{A.r \mapsto \{B\}\} \\ f(R_i, A.r \leftarrow B.s) &\triangleq \{A.r \mapsto R_i(B.s)\} \\ f(R_i, A.r \leftarrow B.s.t) &\triangleq \{A.r \mapsto \textstyle\bigcup_{C \in R_i(B.s)} R_i(C.t)\} \\ f(R_i, A.r \leftarrow B.s \sqcap C.t) &\triangleq \{A.r \mapsto R_i(B.s) \cap R_i(C.t)\} \end{aligned}$$

DEFINITION 3.2. *The* logic-programming semantics of $\mathcal{P}$, *denoted as* $\langle\!|\mathcal{P}|\!\rangle$, *is the minimal Herbrand model of* $\mathtt{LP}(\mathcal{P})$, *the logic program defined as*

$$\mathtt{LP}(\mathcal{P}) \triangleq \bigcup_{c \in \mathcal{P}} \mathtt{lc}(c)$$

$$\text{(RT}_1\text{)}\ \frac{c \in \mathcal{P}}{\mathcal{P} \succ c} \qquad \text{(RT}_2\text{)}\ \frac{\mathcal{P} \succ A.r \leftarrow B.s \quad \mathcal{P} \succ B.s \leftarrow C}{\mathcal{P} \succ A.r \leftarrow C}$$

$$\text{(RT}_3\text{)}\ \frac{\mathcal{P} \succ A.r \leftarrow B.s.t \quad \mathcal{P} \succ B.s \leftarrow C \quad \mathcal{P} \succ C.t \leftarrow D}{\mathcal{P} \succ A.r \leftarrow D}$$

$$\text{(RT}_4\text{)}\ \frac{\mathcal{P} \succ A.r \leftarrow B.s \sqcap C.t \quad \mathcal{P} \succ B.s \leftarrow D \quad \mathcal{P} \succ C.t \leftarrow D}{\mathcal{P} \succ A.r \leftarrow D}$$

**Table 2.** An Inference System for $RT_0$

*where function* $\mathtt{lc}(\cdot)$ *translates every credential to a logic program clause as follows:*

$$\begin{aligned} \mathtt{lc}(A.r \leftarrow B) &\triangleq r(A, B) :- \\ \mathtt{lc}(A.r \leftarrow B.s) &\triangleq r(A, \xi) :- s(B, \xi) \\ \mathtt{lc}(A.r \leftarrow B.s.t) &\triangleq r(A, \xi) :- s(B, \zeta), t(\zeta, \xi) \\ \mathtt{lc}(A.r \leftarrow B.s \sqcap C.t) &\triangleq r(A, \xi) :- s(B, \xi), t(C, \xi) \end{aligned}$$

PROPOSITION 3.1 (see [19]). $r(A, B) \in \langle\!|\mathcal{P}|\!\rangle$ *if and only if* $B \in [\![\mathcal{P}]\!](A.r)$.

***Inference system.*** We now provide an operational semantics for $RT_0$ via a very intuitive inference system. The four kinds of credentials are handled by the four rules in Table 2, where judgement $\mathcal{P} \succ c$ should be read as: "using the credentials in $\mathcal{P}$, we can infer the credential $c$." The rules should be self-explanatory.

We can now prove that the inference system provides an alternative way of presenting the semantics of $RT_0$. (The proof is in [15]).

PROPOSITION 3.2 (Soundness and Completeness). $\mathcal{P} \succ A.r \leftarrow B$ *if and only if* $B \in [\![\mathcal{P}]\!](A.r)$.

EXAMPLE 3 (Example 1, formalised). We use the inference system to formally derive a credential authorising $B$ to inspect *Ent*. To save space, we shorten *Ent* as $E$, *auditor* as $a$, *authSoc* as $aS$, *member* as $m$, *legalSoc* as $l$, *fairSoc* as $f$ and *BSoc* as $BS$. Let $\mathcal{P} \triangleq \{(1), (2), (3), (4), (5), (6)\}$; then,

$$\dfrac{\dfrac{(1) \qquad \dfrac{(2) \qquad \dfrac{(3) \quad (4) \quad (5)}{UK.aS \leftarrow BS} \qquad (6)}{UK.a \leftarrow B}}{E.a \leftarrow B}$$

where the leaf-nodes hold by rule (RT$_1$), the top inference by rule (RT$_4$), the middle inference by rule (RT$_3$) and the bottom one by rule (RT$_2$). □

## 4. Context-dependent credentials

We now extend $RT_0$ by adding boolean guards and time validity; the syntax is in Table 3. A *guard* is either the always-satisfiable guard $\mathtt{tt}$, the atomic statements $B \in A.r$ and $B \notin A.r$ or a conjunction of guards. *Time validity* is defined as expected, with $\tau$ ranging over time constants; we assume the obvious arithmetic

Everything from Table 1, plus

GUARDS:
$g \quad ::= \quad \mathtt{tt} \mid B \in A.r \mid B \notin A.r \mid g_1 \wedge g_2$

TIME VALIDITY:
$\upsilon \quad ::= \quad [\tau_1, \tau_2] \mid [\tau_1, \tau_2) \mid (\tau_1, \tau_2] \mid (\tau_1, \tau_2) \mid (-\infty, \tau]$
$\mid \quad (-\infty, \tau) \mid [\tau, +\infty) \mid (\tau, +\infty) \mid (-\infty, +\infty)$
$\mid \quad \upsilon_1 \cup \upsilon_2 \mid \upsilon_1 \cap \upsilon_2 \mid \upsilon_1 \setminus \upsilon_2$

CONTEXT-DEPENDENT CREDENTIALS:
$\chi \quad ::= \quad$ **if** $g$ **then** $c$ **in** $\upsilon$

**Table 3.** Syntax of Context-dependent Credentials

on validity. In particular, notice that

$$(-\infty, +\infty) \setminus (-\infty, +\infty) = \emptyset$$

where $\emptyset$ can be represented in our syntax, e.g., as $(\tau_1, \tau_2)$ with $\tau_2 < \tau_1$.

*Context-dependent credentials* (CDCs, for short) take the form **if** $g$ **then** $c$ **in** $\upsilon$, meaning "the credential $c$ is available during $\upsilon$, provided the guard $g$ is satisfied by the execution context." Finite sets of CDCs are denoted by $\aleph$. To make notation lighter, we write "**if** $g$ **then** $c$" to denote the CDC "**if** $g$ **then** $c$ **in** $(-\infty, +\infty)$." Similarly, we write "$c$ **in** $\upsilon$" for "**if** $\mathtt{tt}$ **then** $c$ **in** $\upsilon$." Finally, we write "$c$" to denote "**if** $\mathtt{tt}$ **then** $c$ **in** $(-\infty, +\infty)$."

### 4.1 Logic-programming semantics

As with $RT_0$, we shall now give three semantics to sets of CDCs. We start with the logic-programming approach, since the subsequent constructions will be inspired by it. The semantics is calculated at a precise time instant, by only considering those CDCs that are valid at that moment.

DEFINITION 4.1. *The* logic-programming semantics of $\aleph$ at time $\tau$, *denoted as* $\langle\!|\aleph|\!\rangle_\tau$, *is the stable model semantics of* $\mathtt{GLP}_\tau(\aleph)$, *the general logic program defined as*

$$\mathtt{GLP}_\tau(\aleph) \triangleq \bigcup_{\substack{\textbf{if } g \textbf{ then } c \textbf{ in } \upsilon \in \aleph \\ \tau \in \upsilon}} \mathtt{glc}(\textbf{if } g \textbf{ then } c \textbf{ in } \upsilon)$$

*where* $\mathtt{glc}(\textbf{if } g \textbf{ then } c \textbf{ in } \upsilon) \triangleq \mathtt{lc}(c), \mathtt{cond}(g)$ *with*

$$\begin{aligned} \mathtt{cond}(\mathtt{tt}) &\triangleq \emptyset \\ \mathtt{cond}(B \in A.r) &\triangleq r(A, B) \\ \mathtt{cond}(B \notin A.r) &\triangleq \neg\, r(A, B) \\ \mathtt{cond}(g_1 \wedge g_2) &\triangleq \mathtt{cond}(g_1), \mathtt{cond}(g_2) \end{aligned}$$

*and* $\mathtt{lc}(c)$ *as in Definition 3.2.*

We chose the stable model semantics because, apart from being one of the most generous semantics for general logic programs (cf. Section 2), it can be seen as "the sets of belief that a rational agent might hold" [14]. Quoting from [14]:

> "if $K$ is the set of atoms I consider true, then any clause that has a subgoal $\neg\alpha$ with $\alpha \in K$ is, from my point of view, useless; furthermore, any subgoal $\neg\alpha$ with $\alpha \notin K$ is, from my point of view, trivial. Then, I can simplify the clauses in $\mathtt{GLP}_\tau(\aleph)$. If $K$ happens to be precisely the set of atoms that logically follow from the simplified set of clauses, then I am 'rational'."

The 'rational agent' explanation of the stable model semantics given above can be readily rephrased to intuitively justify the semantics for CDCs. Indeed, if $K$ is the entities-to-rôles assignment that should hold, then any credential whose guard contains $B \notin A.r$, with $r(A, B) \in K$, is useless as its guard is not satisfied; furthermore, any guard $B \notin A.r$ with $r(A, B) \notin K$ is trivially satisfied.

### 4.2 Set-theoretic semantics

An alternative way to define the semantics of CDCs is to directly state which entities belong to every rôle. This can be done by adapting Definition 3.1 to the current framework. Again, to avoid inconsistencies, we shall follow the approach put forward by [14] and exploit the stable model construction. Moreover, like before, the semantics will be parameterised with the evaluation time.

In the following definition, we shall exploit two auxiliary notations; let $R$ be a function mapping rôles to sets of entity names. Then, $\aleph|_R$ is the set of CDCs obtained from $\aleph$ by deleting $(a)$ each credential containing a negative guard $B \notin A.r$, with $B \in R(A.r)$, and $(b)$ all negative guards in the remaining credentials. Moreover, predicate $R \models g$ is defined as follows:

$$\begin{array}{ll} R \models \mathtt{tt} & \text{always} \\ R \models B \in A.r & \text{iff } B \in R(A.r) \\ R \models B \notin A.r & \text{iff } B \notin R(A.r) \\ R \models g_1 \wedge g_2 & \text{iff } R \models g_1 \text{ and } R \models g_2 \end{array}$$

DEFINITION 4.2. *The* set-theoretic semantics of $\aleph$ at time $\tau$, *denoted as* $[\![\aleph]\!]_\tau$, *is the function mapping rôles to sets of entity names* $R$ *such that:*

1. $R$ *is* stable for $\aleph$ at time $\tau$, *i.e. it coincides with the least fixpoint (w.r.t. point-wise set inclusion) of the following sequence of functions:*

$$\begin{aligned} R_0 &\quad \textit{is the function mapping every rôle to } \emptyset \\ R_{i+1} &\triangleq \bigoplus_{s.t.\ R_i \models g \ \wedge\ \tau \in \upsilon}^{\textbf{if } g \textbf{ then } c \textbf{ in } \upsilon \in \aleph|_R} f(R_i, c) \end{aligned}$$

   *where '* $\oplus$ *' and* $f$ *are defined in Definition 3.1.*
2. $R$, *if exists, is the only stable function for* $\aleph$ *at time* $\tau$.

The coincidence of these two semantics can be established via the following Lemma (whose proof is given in [15]).

LEMMA 4.1.

1. *Let* $R$ *be a stable function for* $\aleph$ *at time* $\tau$*; then, the set* $K \triangleq \{y(X, Z) : Z \in R(X.y)\}$ *is a stable model of* $\mathtt{GLP}_\tau(\aleph)$.
2. *Let* $K$ *be a stable model of* $\mathtt{GLP}_\tau(\aleph)$*; then, the function* $R$ *that maps every* $A.r$ *to* $\{B : r(A, B) \in K\}$ *is a stable function for* $\aleph$ *at time* $\tau$.

PROPOSITION 4.2 (Coincidence of the Semantics). *r(A,B)* $\in \langle\!|\aleph|\!\rangle_\tau$ *if and only if* $B \in [\![\aleph]\!]_\tau(A.r)$.

**Proof:** For the 'if' part (the 'only if' part is proved similarly), let $R = [\![\aleph]\!]_\tau$. We know, by Lemma 4.1(1), that the set $K \triangleq \{\, y(X, Z) : Z \in R(X.y) \,\}$ is a stable model of $\mathtt{GLP}_\tau(\aleph)$; we now prove that it is its only stable model. By contradiction, let $K'$ be a different stable model of $\mathtt{GLP}_\tau(\aleph)$; by Lemma 4.1(2), we know that the function $R'$ mapping $A.r$ to $\{B : r(A, B) \in K'\}$ is stable for $\aleph$ at time $\tau$. But this contradicts the fact that $R$ is the only stable function for $\aleph$ at $\tau$, that instead holds since $R = [\![\aleph]\!]_\tau$. Now, assume that $B \in [\![\aleph]\!]_\tau(A.r)$; by construction, $r(A, B) \in K = \langle\!|\aleph|\!\rangle_\tau$. ■

$(\mathrm{CDC}_1)$

$$\frac{\textbf{if } \bigwedge_i B_i \in A_i.r_i \ \wedge \bigwedge_j B'_j \notin A'_j.r'_j \textbf{ then } c \textbf{ in } v \in \aleph \qquad \forall i\,.\,\aleph \vdash_\tau A_i.r_i \leftarrow B_i \qquad \forall j\,.\,\aleph \nvdash_\tau A'_j.r'_j \leftarrow B'_j \qquad \tau \in v}{\aleph \vdash_\tau c}$$

$(\mathrm{CDC}_2)$

$$\frac{\aleph \vdash_\tau A.r \leftarrow B.s \qquad \aleph \vdash_\tau B.s \leftarrow C}{\aleph \vdash_\tau A.r \leftarrow C}$$

$(\mathrm{CDC}_3)$

$$\frac{\aleph \vdash_\tau A.r \leftarrow B.s.t \qquad \aleph \vdash_\tau B.s \leftarrow C \qquad \aleph \vdash_\tau C.t \leftarrow D}{\aleph \vdash_\tau A.r \leftarrow D}$$

$(\mathrm{CDC}_4)$

$$\frac{\aleph \vdash_\tau A.r \leftarrow B.s \sqcap C.t \qquad \aleph \vdash_\tau B.s \leftarrow D \qquad \aleph \vdash_\tau C.t \leftarrow D}{\aleph \vdash_\tau A.r \leftarrow D}$$

**Table 4.** An Inference System for Context-dependent Credentials

### 4.3 Inference system

We now adapt the inference system of Table 2 to take guards and validity into account. To simplify notation, in what follows we shall equate guards up-to commutativity, associativity and idempotency of $\wedge$, and up-to absorption of $\mathbf{tt}$-conjuncts. Thus, every CDC **if** $g$ **then** $c$ **in** $v$ will be considered in a 'normal form' like

$$\textbf{if } \bigwedge_{i\in I} B_i \in A_i.r_i \ \wedge \bigwedge_{j\in J} B'_j \notin A'_j.r'_j \textbf{ then } c \textbf{ in } v$$

for finite (and possibly empty) index sets $I$ and $J$, with $g$ logically equivalent to $\bigwedge_{i\in I} B_i \in A_i.r_i \ \wedge \bigwedge_{j\in J} B'_j \notin A'_j.r'_j$.

The inference system for guarded credentials is given in Table 4; it mainly extends the inference rules given in Table 2 by replacing $(\mathrm{RT}_1)$ with $(\mathrm{CDC}_1)$ and by considering only valid CDCs. Rule $(\mathrm{CDC}_1)$ requires that, to use a guarded credential, all its positive guards should be inferable and none of its negative guards can be inferred. However, the latter requirement is far from being innocuous: the presence of negative premises can undermine the well-foundedness of the inference system. Moreover, as in the logic-programming framework, sets of rules of the proposed form do not necessarily define a unique relation. To deal with these problems, we slightly adapt the well-known theory developed in [7] for transition systems with negative premises.

First, notice that inference systems are usually defined via *inference system specifications* (as in Tables 2 and 4); these are (small) collections of *rule schemata*, i.e. "meta-rules" which use names as placeholders; that is, they are implicitly universally quantified. If the rule schemata have no negative premises, they uniquely determine an *inference set*, i.e. the set of all judgements derivable from them via a finite depth inference. More precisely, given a 'positive' inference system specification $\mathcal{I}$, we first instantiate the placeholders in $\mathcal{I}$ with any possible name, thus obtaining a (possibly infinite) set of (closed) inference rules $\mathcal{I}'$; then, the inference set associated to $\mathcal{I}$ is $\mathtt{FinInf}(\mathcal{I}')$, the set formed by all the judgements derivable via a finite depth inference using rules from $\mathcal{I}'$.

With negative premises, things become more delicate, because of well-foundedness problems: the associated inference set can be defined in several ways and, according to the definition chosen, an inference system specification can determine zero, one or several inference sets. We now adapt the approach of [7] to define an inference set for the inference system specification in Table 4.

Given a set of guarded credentials $\aleph$ and a time $\tau$, we aim at defining a set of $RT_0$-credentials $([\![\aleph]\!])_\tau$ such that, if $c \in ([\![\aleph]\!])_\tau$, then $c$ is inferable from $\aleph$ at time $\tau$. Hence, we guess a set of $RT_0$-credentials $\mathcal{P}$ (the candidate inference set) and use the 'implicit' definition put forward by stable models to define the inference set associated to the inference system. Like in the stable model approach, $\mathcal{P}$ plays the rôle of an 'oracle,' stating what is inferable from $\aleph$ at $\tau$ and what is not.

In what follows, we let $\mathrm{POS}(g)$ and $\mathrm{NEG}(g)$ contain, respectively, the positive and negative guards occurring in $g$, with $\mathbf{tt}$ belonging to $\mathrm{POS}(g)$. Moreover, we use the shortcut $\aleph \vdash_\tau g$, defined as follows:

$$\begin{array}{rcl} \aleph \vdash_\tau \mathbf{tt} & \triangleq & \emptyset \\ \aleph \vdash_\tau B \in A.r & \triangleq & \aleph \vdash_\tau A.r \leftarrow B \\ \aleph \vdash_\tau B \notin A.r & \triangleq & \aleph \nvdash_\tau A.r \leftarrow B \\ \aleph \vdash_\tau g_1 \wedge g_2 & \triangleq & \aleph \vdash_\tau g_1 \quad \wedge \quad \aleph \vdash_\tau g_2 \end{array}$$

Finally, we shall say that an inference rule *holds in* $\vdash_\tau$ *for* $\aleph$ if it is obtained by replacing the placeholders of a rule schema in $\vdash_\tau$ with any rôle and entity name occurring in $\aleph$.

DEFINITION 4.3. *Given a set of CDCs $\aleph$, a time $\tau$ and the inference system specification $\vdash_\tau$ of Table 4,* the inference set associated with $\vdash_\tau$ under $\aleph$, *written $([\![\aleph]\!])_\tau$, is the set of $RT_0$-credentials $\mathcal{P}$ such that*

1. *$\mathcal{P}$ is* stable for $\aleph$ and $\vdash_\tau$, *i.e.*

$$\mathcal{P} = \mathtt{FinInf}(\mathit{Strip}(\vdash_\tau, \aleph, \mathcal{P}))$$

*where $\mathit{Strip}(\vdash_\tau, \aleph, \mathcal{P})$ is the set formed by all the instances of $(\mathrm{CDC}_2)$, $(\mathrm{CDC}_3)$ and $(\mathrm{CDC}_4)$ that hold in $\vdash_\tau$ for $\aleph$ and all the rules*

$$\frac{\textbf{if } g \textbf{ then } c \textbf{ in } v \in \aleph \qquad \tau \in v \qquad \aleph \vdash_\tau \mathrm{POS}(g)}{\aleph \vdash_\tau c}$$

*such that*

$$\frac{\textbf{if } g \textbf{ then } c \textbf{ in } v \in \aleph \qquad \tau \in v \qquad \aleph \vdash_\tau g}{\aleph \vdash_\tau c}$$

*holds in $\vdash_\tau$ for $\aleph$ and $A.r \leftarrow B \notin \mathcal{P}$, for all $B \notin A.r$ belonging to $\mathrm{NEG}(g)$.*

2. *$\mathcal{P}$, if exists, is the only stable set for $\aleph$ and $\vdash_\tau$.*

Soundness and completeness of the inference system w.r.t. the two semantics presented in the previous subsections can be established as a consequence of the following Lemma (whose proof is in [15]).

LEMMA 4.3.

1. *Let $\mathcal{P}$ be stable for $\aleph$ and $\vdash_\tau$; then, the set $K \triangleq \{y(X,Z) : X.y \leftarrow Z \in \mathcal{P}\}$ is a stable model of $\mathtt{GLP}_\tau(\aleph)$.*
2. *Let $K$ be a stable model of $\mathtt{GLP}_\tau(\aleph)$; then, the set $\mathcal{P} \triangleq \mathtt{FinInf}(\mathit{Strip}(\vdash_\tau, \aleph, \{A.r \leftarrow B : r(A,B) \in K\}))$ is stable for $\aleph$ and $\vdash_\tau$.*

PROPOSITION 4.4 (Soundness and Completeness). *$r(A,B) \in \langle\!|\aleph|\!\rangle_\tau$ if and only if $A.r \leftarrow B \in ([\![\aleph]\!])_\tau$.*

**Proof:** Like the proof of Proposition 4.2, but relying on Lemma 4.3. ■

EXAMPLE 4 (Example 1, continued). We now use the inference system to formally justify (10) and (12). Let $v = [\tau, +\infty)$ be the time validity of (4). Now, let $\aleph \triangleq \{(2), (3), (7), (8), (9), (11)\}$ and use the abbreviations introduced in Example 3, plus $e$ for *employees*; then, pick any $\tau \in v \cap v_1 \cap v_2$ and, by rule ($\mathrm{CDC}_1$), infer

$$\dfrac{(11) \in \aleph \qquad \dfrac{\aleph \vdash_\tau (2) \qquad \dfrac{\aleph \vdash_\tau (3) \quad \aleph \vdash_\tau (7) \quad \aleph \vdash_\tau (8)}{\aleph \vdash_\tau UK.aS \leftarrow BS} \qquad \aleph \vdash_\tau (9)}{\aleph \vdash_\tau UK.a \leftarrow B} \qquad \aleph \nvdash_\tau E.e \leftarrow B}{\aleph \vdash_\tau E.a \leftarrow B}$$

Clearly $E.e \leftarrow B$ cannot be inferred from $\aleph$ (at any time) and, like before, the leaf-nodes hold by rule ($\mathrm{CDC}_1$), the top inference by rule ($\mathrm{CDC}_4$), the middle inference by rule ($\mathrm{CDC}_3$) and the bottom one by rule ($\mathrm{CDC}_2$).

Now, consider $\aleph' \triangleq \aleph \cup \{(12)\}$; in this case, the inference shown above no longer holds because now, trivially, $\aleph' \vdash_\tau E.e \leftarrow B$. If we assume (12) be valid only in $v'$, then the inference shown above holds only in $(v \cap v_1 \cap v_2) \setminus v'$. □

To conclude, let us point out two aspects of the last example. Firstly, (11) is still an approximation of realistic auditing behaviour, since it is also reasonable to let $B$ become an *Ent*'s auditor not only if he is not a current employee of *Ent*, but also if he has not been an employee recently. This feature can be modelled with CDCs as follows: add the credential

$$Ent.recentEmpl \leftarrow B \textbf{ in } [\tau', \tau' + \tau'']$$

where $\tau'$ is the instant in which $B$ stops to be an employee of *Ent* and $\tau''$ is the offset after which it is assumed that $B$ can inspect *Ent*. Moreover, replace (11) with

$$\textbf{if } B \in UK.auditor \wedge B \notin Ent.employees \ \wedge \ B \notin Ent.recentEmpl \textbf{ then } Ent.auditor \leftarrow B$$

Secondly, there is some room for improvements is the definition of (11); compared with (1), its weakness is that we must specialise the credential to every possible $B$. A better solution would be something like

$$Ent.auditor \leftarrow UK.auditor \sqcap \neg Ent.employee$$

We do not believe that such a feature would radically change the theory presented in this paper; nonetheless, we leave its investigation for future research.

## 5. Deriving Contexts for Execution

Context-dependent credentials, as the name suggests, depend on the context where the credential is exhibited, namely the other credentials available and the exact time of evaluation. So it is of interest, given a set of CDCs, to determine some constraints on the execution context that enable a desired inference, whenever possible. This turns out to be fundamental in large-scale distributed systems where users have partial views of their execution context.

As we have already discussed in Section 1, a context is a pair $\langle \aleph; \tau \rangle$ that defines the set of CDCs made available and the time of the evaluation. The problem we now want to solve is the following:

> given a set of CDCs $\aleph$ (representing the credentials available to a user) and a goal $c$, which are the assumptions that should be made on the execution context in order to derive the goal?

As in the previous sections, we shall follow two lines of work: firstly, we adapt the inference system of Section 4.3 to also derive constraints on the environment, both on the execution time and on the CDCs it must provide or not provide; secondly, we enhance the logic-programming semantics of Section 4.1 with features taken from the field of abductive logic programming.

($\mathrm{TV}_1$)
$$\dfrac{\textbf{if } \bigwedge_i B_i \in A_i.r_i \ \wedge \ \bigwedge_j B'_j \notin A'_j.r'_j \textbf{ then } c \textbf{ in } v \in \aleph \qquad \forall i \,.\, \aleph \Vdash_{v_i} A_i.r_i \leftarrow B_i \qquad \forall j \,.\, \aleph \Vdash_{v_j} A'_j.r'_j \leftarrow B'_j}{\aleph \Vdash_{(v \cap \bigcap_i v_i) \setminus \bigcup_j v_j} c}$$

($\mathrm{TV}_2$)
$$\dfrac{\aleph \Vdash_{v_1} A.r \leftarrow B.s \qquad \aleph \Vdash_{v_2} B.s \leftarrow C}{\aleph \Vdash_{v_1 \cap v_2} A.r \leftarrow C}$$

($\mathrm{TV}_3$)
$$\dfrac{\aleph \Vdash_{v_1} A.r \leftarrow B.s.t \qquad \aleph \Vdash_{v_2} B.s \leftarrow C \qquad \aleph \Vdash_{v_3} D.t \leftarrow D}{\aleph \Vdash_{v_1 \cap v_2 \cap v_3} A.r \leftarrow D}$$

($\mathrm{TV}_4$)
$$\dfrac{\aleph \Vdash_{v_1} A.r \leftarrow B.s \sqcap C.t \qquad \aleph \Vdash_{v_2} B.s \leftarrow D \qquad \aleph \Vdash_{v_3} C.t \leftarrow D}{\aleph \Vdash_{v_1 \cap v_2 \cap v_3} A.r \leftarrow D}$$

($\mathrm{TV}_5$)
$$\dfrac{\aleph \Vdash_{v_1} c \qquad \aleph \Vdash_{v_2} c}{\aleph \Vdash_{v_1 \cup v_2} c}$$

**Table 5.** Inferring Time Validity

### 5.1 Adapting the Inference System

For the sake of presentation, we shall give the inference system in two steps: we first give an inference system that calculates the maximal time validity in which a certain $RT_0$-credential can be inferred from a given set of CDCs; then, we give an inference system that calculates a minimal set of CDCs that must be added to the given set of CDCs to infer a certain $RT_0$-credential. Clearly, the two systems can be combined, but this would complicate the presentation. Moreover, a nice feature of these inference systems is that they have no negative premises; this makes it trivial to define the inference set associated with them.

***Inferring Time Validity.*** The revised inference system enhances judgements of the form $\aleph \vdash_\tau c$ to yield the new judgements

$$\aleph \Vdash_v c$$

Intuitively, $\aleph \Vdash_v c$ means that, at any time $\tau \in v$ in which $\aleph$ has a semantics, it is possible to derive the credential $c$ from $\aleph$.

The inference rules are in Table 5. The key rule is ($\mathrm{TV}_1$): it states that a CDC can be exploited whenever it is valid, whenever its positive premises are satisfied, but not when any of its negative premises may hold. Clearly, the base case of the inference system is ($\mathrm{TV}_1$) involving a CDC whose guard is $\mathtt{tt}$. Rules ($\mathrm{TV}_2$), ($\mathrm{TV}_3$) and ($\mathrm{TV}_4$) simply state that an inference rule can be applied only

if all its premises hold. Finally, rule ($\text{TV}_5$) states that, if a $RT_0$ credential $c$ can be inferred both with validity $v_1$ and with validity $v_2$, then $c$ can be inferred with validity $v_1 \cup v_2$.

Notice that $\Vdash_v$ generalises $\vdash_\tau$: indeed, the former coincides with the latter whenever $v = [\tau, \tau]$. Moreover, since in general there are several possible ways to infer a certain $RT_0$ credential $c$ from $\aleph$, rule ($\text{TV}_5$) can be used several times to enlarge $c$'s validity. We now want to isolate all those inferences in which ($\text{TV}_5$) has been used to enlarge as much as possible the validity.

DEFINITION 5.1. *An inference terminating in* $\aleph \Vdash_v c$ *is called* maximal *if and only if*

1. *there exists no* $v' \supset v$ *such that* $\aleph \Vdash_{v'} c$, *and*
2. *every its sub-inference terminating in* $\aleph \Vdash_{v'} c'$, *for* $c' \neq c$, *is maximal.*

The first requirement ensures that ($\text{TV}_5$) has been used as much as possible to infer the validity of $c$. The second requirement of the Definition ensures that this property is propagated through all the inference tree. We are interested in maximal inferences since they guarantee that the $v_i$ and $v_j$ in the premise of ($\text{TV}_1$) are the maximal time validity for $A_i.r_i \leftarrow B_i$ and $A'_j.r'_j \leftarrow B'_j$, respectively; thus, for these inferences we can prove soundness and completeness of $\Vdash_v$, by means of Proposition 5.2 whose proof relies on the following Lemma.

LEMMA 5.1. $\aleph \vdash_\tau c$ *implies that there exists a* $v$ *containing* $\tau$ *such that* $\aleph \Vdash_v c$.

**Proof:** It suffices to mimic the derivation for $\aleph \vdash_\tau c$ by replacing every application of rule ($\text{CDC}_i$) with an application of rule ($\text{TV}_i$); $v$ will be the intersection of the validity of all the CDCs used in the inference and will be at least $[\tau, \tau]$. ■

PROPOSITION 5.2. *Let* $\aleph \Vdash_v c$ *be a maximal inference and* $(\![\aleph]\!)_\tau$ *be defined. Then,* $\aleph \vdash_\tau c$ *if and only if* $\tau \in v$.

**Proof:** By induction on the depth of $\aleph \Vdash_v c$. For the base case, $\aleph$ must contain a CDC **if** $\mathtt{tt}$ **then** $c$ **in** $v$. If $\tau \in v$, we trivially conclude thanks to ($\text{CDC}_1$). Vice versa, assume by contradiction that there is a $\tau' \notin v$ such that $\aleph \vdash_{\tau'} c$; but then the inference leading to $\aleph \Vdash_v c$ would not be maximal, because of Lemma 5.1.

For the inductive step, we reason by case analysis on the last rule used; the most difficult cases are when using ($\text{TV}_1$) and ($\text{TV}_5$). In the first case, we have that $\aleph$ contains a CDC **if** $\bigwedge_i B_i \in A_i.r_i \wedge \bigwedge_j B'_j \notin A'_j.r'_j$ **then** $c$ **in** $v'$, with $v = (v' \cap \bigcap_i v_i) \setminus \bigcup_j v_j$. By induction, $\aleph \vdash_\tau A_i.r_i \leftarrow B_i$ if and only if $\tau \in v_i$ and $\aleph \vdash_\tau A'_j.r'_j \leftarrow B'_j$ if and only if $\tau \in v_j$; hence, $\aleph \nvdash_\tau A'_j.r'_j \leftarrow B'_j$ if and only if $\tau \notin v_j$. We then work like in the base case. If $\aleph \Vdash_v c$ terminates with an application of ($\text{TV}_5$), then $v = v_1 \cup v_2$. This case is less straightforward, because judgements $\aleph \Vdash_{v_1} c$ and $\aleph \Vdash_{v_2} c$ are, in general, not maximal.[2] Let $\aleph \vdash_\tau c$; by Lemma 5.1, there exists a $v'$ containing $\tau$ such that $\aleph \Vdash_{v'} c$. Now, it must be that $v' \subseteq v$, otherwise $\aleph \Vdash_v c$ would not be maximal; we trivially conclude. Vive versa, let $\tau \in v$ and let $\aleph \Vdash_{v'} c$ be the deepest sub-inference of $\aleph \Vdash_v c$ whose premises do not entail $c$ (hence, $\aleph \Vdash_{v'} c$ has been obtained via an application of ($\text{TV}_i$), for $i \neq 5$) and such that $\tau \in v'$. By definition of the rules in Table 5, each of these premises has a validity containing $\tau$; since these premises have been obtained via maximal inferences, by induction we can replace $\Vdash_{\ldots}$ with $\vdash_\tau$. Now, apply ($\text{CDC}_i$) and easily conclude. ■

[2] They can be maximal whenever $v_1 \subseteq v_2$ or vice versa, i.e. when the inference terminates with a 'trivial' application of ($\text{TV}_5$), that is always possible. In this case, the proof relies on a straightforward induction.

EXAMPLE 5 (Example 2, formalised). Consider the set of CDCs $\aleph = \{(13), (14)\}$; when does it make the credential *Alice.readMail* $\leftarrow$ *Ent.secr* available (if ever)? As we said informally before, such a credential can be used at any time not included in $v$. We now formally derive this statement. By using ($\text{TV}_1$), this can be done easily:

$$\frac{(13) \in \aleph \qquad \aleph \Vdash_v \textit{Ent.active} \leftarrow \textit{Alice}}{\aleph \Vdash_{(-\infty,+\infty)\setminus v} \textit{Alice.readMail} \leftarrow \textit{Ent.secr}}$$

Indeed, (13) is a time-independent credential, i.e. it holds in all $(-\infty, +\infty)$.

Now, consider the credentials

$$\textit{Ent.active} \leftarrow \textit{Ent.inMission} \tag{19}$$

$$\textit{Ent.inMission} \leftarrow \textit{Alice}\ \mathbf{in}\ v' \tag{20}$$

and the set $\aleph' = \aleph \cup \{(19), (20)\}$. Intuitively, (19) states that employees of *Ent* that are out on a mission are still active, whereas (20) states that *Alice* is in mission during $v'$. Now, the inference previously shown becomes (with the obvious shorthands for entity and rôle names, plus $\Vdash$ standing for $\Vdash_{(-\infty,+\infty)}$)

$$\frac{\dfrac{\dfrac{\aleph' \Vdash E.a \leftarrow E.i \quad \aleph' \Vdash_{v'} E.i \leftarrow A}{\aleph' \Vdash_v E.a \leftarrow A} \qquad \aleph' \Vdash_{v'} E.a \leftarrow A}{(13) \in \aleph' \qquad \aleph' \Vdash_{v \cup v'} E.a \leftarrow A}}{\aleph \Vdash_{(-\infty,+\infty)\setminus(v \cup v')} A.r \leftarrow E.s}$$

and states that *Ent.secr* can read *Alice*'s mail only outside $v \cup v'$. □

***Inferring Environmental CDCs.*** The second inference system enhances judgements of the form $\aleph \vdash_\tau c$ to yield judgements like

$$\aleph \Vdash^\phi_\tau c$$

Intuitively, $\aleph \Vdash^\phi_\tau c$ means that it is possible to derive the credential $c$ from $\aleph$ at time $\tau$ in any execution context providing enough CDCs to satisfy $\phi$. Here, we let $\phi$ range over propositional formulae over the atoms $B \in A.r$, i.e.

$$\phi ::= \mathtt{tt} \mid B \in A.r \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$$

We shall equate propositional formulae up-to commutativity, associativity and idempotency of both $\wedge$ and $\vee$, and up-to absorption of identity elements.

The aim of propositional formulae is to characterise sets of CDCs. Formally, the satisfiability relation is defined as follows:

| | |
|---|---|
| $\aleph \vDash_\tau \mathtt{tt}$ | iff $[\![\aleph]\!]_\tau$ exists |
| $\aleph \vDash_\tau B \in A.r$ | iff $B \in [\![\aleph]\!]_\tau(A.r)$ |
| $\aleph \vDash_\tau \neg\phi$ | iff $\aleph \nvDash_\tau \phi$ |
| $\aleph \vDash_\tau \phi_1 \wedge \phi_2$ | iff $\aleph \vDash_\tau \phi_1$ and $\aleph \vDash_\tau \phi_2$ |
| $\aleph \vDash_\tau \phi_1 \vee \phi_2$ | iff $\aleph \vDash_\tau \phi_1$ or $\aleph \vDash_\tau \phi_2$ |

The inference rules are in Table 6. The key rule is ($\text{ENV}_1$): it states that a CDC can be exploited whenever it is valid and in any context enabling the inference of its positive premises but not enabling the inference of its negative premises. However, notice that we can always trivially infer $A.r \leftarrow B$ if the context provides it at time $\tau$; this justifies the new rule ($\text{ENV}_0$).

Also in this case, notice that $\Vdash^\phi_v$ generalises $\vdash_\tau$: the former coincides with the latter whenever $\phi$ is $\mathtt{tt}$, up-to absorption of boolean constants. Moreover, a rule similar to ($\text{TV}_5$)

$$\frac{\aleph \Vdash^{\phi_1}_\tau c \qquad \aleph \Vdash^{\phi_2}_\tau c}{\aleph \Vdash^{\phi_1 \vee \phi_2}_\tau c}$$

($\text{ENV}_0$)

$$\frac{}{\aleph \Vdash_\tau^{B \in A.r} A.r \leftarrow B}$$

($\text{ENV}_1$)

$$\frac{\tau \in \upsilon \qquad \forall\, i\,.\, \aleph \Vdash_\tau^{\phi_i} A_i.r_i \leftarrow B_i}{\aleph \Vdash_\tau^{\bigwedge_i \phi_i \,\wedge\, \bigwedge_j \neg(B'_j \in A'_j.r'_j)} c}$$

**if** $\bigwedge_i B_i \in A_i.r_i \,\wedge\, \bigwedge_j B'_j \notin A'_j.r'_j$ **then** $c$ **in** $\upsilon \in \aleph$

($\text{ENV}_2$)

$$\frac{\aleph \Vdash_\tau^{\phi_1} A.r \leftarrow B.s \qquad \aleph \Vdash_\tau^{\phi_2} B.s \leftarrow C}{\aleph \Vdash_\tau^{\phi_1 \wedge \phi_2} A.r \leftarrow C}$$

($\text{ENV}_3$)

$$\frac{\aleph \Vdash_\tau^{\phi_1} A.r \leftarrow B.s.t \qquad \aleph \Vdash_\tau^{\phi_2} B.s \leftarrow C \qquad \aleph \Vdash_\tau^{\phi_3} C.t \leftarrow D}{\aleph \Vdash_\tau^{\phi_1 \wedge \phi_2 \wedge \phi_3} A.r \leftarrow D}$$

($\text{ENV}_4$)

$$\frac{\aleph \Vdash_\tau^{\phi_1} A.r \leftarrow B.s \sqcap C.t \qquad \aleph \Vdash_\tau^{\phi_2} B.s \leftarrow D \qquad \aleph \Vdash_\tau^{\phi_3} C.t \leftarrow D}{\aleph \Vdash_\tau^{\phi_1 \wedge \phi_2 \wedge \phi_3} A.r \leftarrow D}$$

**Table 6.** Inferring Environmental Credentials

could also be introduced in this inference system. However, it would only make $\aleph \Vdash_\tau^{\phi} c$ more descriptive, in the sense that the formula $\phi$ would describe different possible contexts enabling the inference of $c$ from $\aleph$ at time $\tau$. Indeed, differently from $\Vdash_\upsilon$ (where rule ($\text{TV}_5$) was crucial to prove Proposition 5.2), we can prove soundness and completeness of $\Vdash_\tau^{\phi}$ without any requirement on the strategy used in the inference of $\aleph \Vdash_\tau^{\phi} c$.

PROPOSITION 5.3 (Soundness). *If* $\aleph \Vdash_\tau^{\phi} c$*, then* $\aleph \cup \aleph' \vdash_\tau c$*, for any set of CDCs* $\aleph'$ *such that* $\aleph \cup \aleph' \models_\tau \phi$*.*

**Proof:** First, notice that $\aleph \cup \aleph' \models_\tau \phi$ implies that $\aleph \cup \aleph'$ has a (logic-programming/set-theoretic/inference-based) semantics; thus, $\vdash_\tau$ is meaningful. The proof is by induction on the depth of $\aleph \Vdash_\tau^{\phi} c$. We have two possible base cases:

- ($\text{ENV}_0$): in this case, $c = A.r \leftarrow B$. By definition of $\models_\tau$, $\aleph \cup \aleph' \models_\tau B \in A.r$ implies that $B \in [\![\aleph \cup \aleph']\!]_\tau(A.r)$; by Propositions 4.2 and 4.4, this entails $\aleph \cup \aleph' \vdash_\tau A.r \leftarrow B$.
- ($\text{ENV}_1$): by hypothesis, $\aleph$ must contain a CDC **if** $\mathtt{tt}$ **then** $c$ **in** $\upsilon$, for $\upsilon \ni \tau$ and $\phi = \mathtt{tt}$. Then, trivially, $\aleph \cup \aleph' \vdash_\tau c$.

For the inductive step, we reason by case analysis on the last rule used; we only consider the case in which the inference terminates with ($\text{ENV}_1$), since the other ones rely on a simple induction. In this case, we have that $\aleph$ contains a CDC **if** $\bigwedge_i B_i \in A_i.r_i \wedge \bigwedge_j B'_j \notin A'_j.r'_j$ **then** $c$ **in** $\upsilon$ and that $\aleph \Vdash_\tau^{\phi_i} A_i.r_i \leftarrow B_i$, for every $i$; moreover, $\phi = \bigwedge_i \phi_i \wedge \bigwedge_j \neg(B'_j \notin A'_j.r'_j)$. By induction, $\aleph \cup \aleph' \models_\tau \phi_i$ (that holds since $\aleph \cup \aleph' \models_\tau \phi$) implies that $\aleph \cup \aleph' \vdash_\tau A_i.r_i \leftarrow B_i$. If we prove that $\aleph \cup \aleph' \nvdash_\tau A_j.r_j \leftarrow B_j$, for every $j$, we can conclude by ($\text{CDC}_1$). By contradiction, assume that there exists a $j$ such that $\aleph \cup \aleph' \vdash_\tau A_j.r_j \leftarrow B_j$; then, by Propositions 4.2 and 4.4, $\aleph \cup \aleph' \models_\tau B'_j \in A'_j.r'_j$, in contradiction with $\aleph \cup \aleph' \models_\tau \phi$. ■

PROPOSITION 5.4 (Completeness). *If* $([\![\aleph \cup \aleph']\!]_\tau$ *exists and* $\aleph \cup \aleph' \vdash_\tau c$*, then there exists a* $\phi$ *such that* $\aleph \cup \aleph' \models_\tau \phi$ *and* $\aleph \Vdash_\tau^{\phi} c$*.*

**Proof:** If $c = A.r \leftarrow B$, the proof is simple: by Propositions 4.2 and 4.4, $[\![\aleph \cup \aleph']\!]_\tau$ exists and associates $B$ to $A.r$; hence, $\aleph \cup \aleph' \models_\tau B \in A.r$ and, by rule ($\text{ENV}_0$), $\aleph \Vdash_\tau^{B \in A.r} A.r \leftarrow B$. Let $c \neq A.r \leftarrow B$; in this case, it must be that $\aleph \cup \aleph'$ contains $\chi \triangleq$ **if** $\bigwedge_i B_i \in A_i.r_i \wedge \bigwedge_j B'_j \notin A'_j.r'_j$ **then** $c$ **in** $\upsilon$ such that $\tau \in \upsilon$, $\aleph \cup \aleph' \vdash_\tau A_i.r_i \leftarrow B_i$, for every $i$, and $\aleph \cup \aleph' \nvdash_\tau A'_j.r'_j \leftarrow B'_j$, for every $j$. We now prove that we can always find a $\chi$ such that $\aleph \cup \aleph' \models_\tau \bigwedge_j B'_j \notin A'_j.r'_j$. By contradiction, suppose that every $\chi$ entailing $c$ have at least one $j$ such that $\aleph \cup \aleph' \models_\tau B'_j \in A'_j.r'_j$. This fact, together with Proposition 5.3 and ($\text{ENV}_0$), would imply that $\aleph \cup \aleph' \vdash_\tau A'_j.r'_j \leftarrow B'_j$; thus, we would have no means to infer $\aleph \cup \aleph' \vdash_\tau c$. So, choose one of the $\chi$'s whose negative guards are not satisfied by $\aleph \cup \aleph'$ (as we have just proved, at least one exists). As proved before, for every $i$ there exist a $\phi_i$ such that $\aleph \cup \aleph' \models_\tau \phi_i$ and $\aleph \Vdash_\tau^{\phi_i} A_i.r_i \leftarrow B_i$; trivially, $\aleph \cup \aleph' \models_\tau \bigwedge_i \phi_i \wedge \bigwedge_j \neg(B'_j \in A'_j.r'_j)$ and, by ($\text{ENV}_1$), $\aleph \Vdash_\tau^{\bigwedge_i \phi_i \wedge \bigwedge_j \neg(B'_j \in A'_j.r'_j)} c$, as desired. ■

EXAMPLE 6 (Example 2, formalised). Consider again the set of CDCs $\aleph = \{(13), (14)\}$; which constraints should be put on the environment to let *Bob* read *Alice*'s mail (if possible)? As we have already informally said, the context must provide enough information to infer that *Bob* is a secretary of *Ent* while *Alice* is not active. We now formally derive this statement. Let $\tau \notin \upsilon$; by using ($\text{ENV}_1$), ($\text{ENV}_0$) and ($\text{ENV}_2$), we have

$$\frac{\aleph \Vdash_\tau^{A \notin E.a} A.r \leftarrow E.s \qquad \aleph \Vdash_\tau^{B \in E.s} E.s \leftarrow B}{\aleph \Vdash_\tau^{A \notin E.a \,\wedge\, B \in E.s} A.r \leftarrow B}$$

where $A \notin E.a$ is the obvious shortcut for $\neg(A \in E.a)$.

On the contrary, pick up $\tau \in \upsilon$. Now, since $\aleph \models_\tau A \in E.a$, there exists no $\aleph'$ such that $\aleph \cup \aleph' \models_\tau A \notin E.a$; hence, in no context, the *Ent*'s secretaries can read *Alice*'s mails when she is active, as desired. □

### 5.2 Abductive Logic Programming

The inference system given in Section 4.3 provides a way to decide, given a set of CDCs, which knowledge can be derived from it; as we have proved, such an inference system closely corresponds to the stable model semantics of a general logic program originating from the set of CDCs, see Proposition 4.4. The inference systems presented in Section 5.1 extend the system of Section 4.3 by providing more information, i.e. they describe the requirements an execution context should satisfy in order to derive a given goal. It is then natural to look for a logic-programming counterpart of such inference systems.

There is an intimate correspondence between the inference systems of Section 5.1 and a variant of logic programming, called *abductive logic programming* (ALP) [16]. Apart from theoretical interest, such a correspondence is also of practical use, since well-established proof procedures [17, 13, 10] and tools [1] for ALP provide us with a more tractable and automatable way of working with CDCs.

Informally, ALP seeks to derive, given a general logic program $\mathtt{P}$ and a goal $r(A, B)$, a set of unit clauses $\mathtt{P}'$ such that:

- from $\mathtt{P} \cup \mathtt{P}'$, we can derive $r(A, B)$, and
- $\mathtt{P} \cup \mathtt{P}'$ satisfies some form of 'consistency.'

More precisely, we have the following definition.

DEFINITION 5.2 (Abducible explanation). *Given a general logic program* $\mathtt{P}$ *and a goal* $r(A,B)$*, an* abducible explanation of $r(A,B)$ in $\mathtt{P}$ *is a set of unit clauses* $\mathtt{P}'$ *such that the stable model semantics of* $\mathtt{P} \cup \mathtt{P}'$ *contains* $r(A,B)$.

Intuitively, $\mathtt{P} \cup \mathtt{P}'$ is 'consistent' if it has a stable model semantics (i.e., it has a semantics according to Definition 2.1); moreover, it can be used to derive $r(A,B)$ only if its stable model contains such an atom.

***Relationship between the semantics.*** We now prove a correspondence result between the semantics defined by the inference system in Table 6 and the abducible explanations of Definition 5.2. To this end, we first need to convert a set of unit clauses into a set of CDCs. This is carried out by function UC-CDC$(\cdot)$, whose formal definition is:

$$\begin{array}{rcl} \text{UC-CDC}(\emptyset) & \triangleq & \emptyset \\ \text{UC-CDC}(r(A,B) :- ) & \triangleq & A.r \leftarrow B \\ \text{UC-CDC}(\mathtt{P}_1 \cup \mathtt{P}_2) & \triangleq & \text{UC-CDC}(\mathtt{P}_1) \cup \text{UC-CDC}(\mathtt{P}_2) \end{array}$$

Moreover, we say that $\phi$ *respects* $\aleph$ *at time* $\tau$ if there is a set of CDCs $\aleph'$ of the form $A.r \leftarrow B$ (with a $\mathtt{tt}$-guard and a $(-\infty,+\infty)$ time validity) such that $\aleph' \models_\tau \phi$ and $[\![\aleph \cup \aleph']\!]_\tau$ exists.

PROPOSITION 5.5. *If* $\mathtt{P}$ *is an abducible explanation of* $r(A,B)$ *in* $\mathtt{GLP}_\tau(\aleph)$*, then* $\aleph \Vdash^\phi_\tau A.r \leftarrow B$*, for some* $\phi$ *such that* $\aleph \cup \text{UC-CDC}(\mathtt{P}) \models_\tau \phi$.

PROPOSITION 5.6. *If* $\aleph \Vdash^\phi_\tau A.r \leftarrow B$ *and* $\phi$ *respects* $\aleph$ *at time* $\tau$*, then there exists an abducible explanation* $\mathtt{P}$ *of* $r(A,B)$ *in* $\mathtt{GLP}_\tau(\aleph)$ *such that* $\aleph \cup \text{UC-CDC}(\mathtt{P}) \models_\tau \phi$.

***Exploiting Tools for ALP.*** We now show how to exploit the CIFF tool [10, 1] for ALP to automate the inference systems of Tables 5 and 6. To this end, we first need to introduce time validity of CDCs in their translation to logic clauses; this can be achieved by exploiting *constraints*, that have been introduced in ALP in [18] and that can be handled by CIFF. Now, general logic clauses can have in their premises more general predicates, like tests for equality or ordering relations among values and variables of arbitrary data types, such as integers or reals. Moreover, to reduce the space search, CIFF also uses some *integrity constraints* (not to be confused with constraints over data types) of the form $\lambda_1, \cdots, \lambda_k \Rightarrow \alpha$. Intuitively, integrity constraints are used to select, among all the abducible explanations, only those which satisfy all such implications. In our framework, we shall exploit very simple integrity constraints, but more sophisticated scenarios could also be considered.

To include timing information in the translation of sets of CDCs, from now on we shall always work with ternary relations $r(A,B,\zeta)$. Intuitively, $r(A,B,\zeta)$ extends $r(A,B)$ in that the constraints involving variable $\zeta$ in the definition of the predicate $r$ define the validity of the atom. For example, the CDC **if** $\mathtt{tt}$ **then** $A.r \leftarrow B$ **in** $[\tau,+\infty)$ is translated into the clause $r(A,B,\zeta) :- \zeta \geq \tau$, meaning that $B$ belongs to $A.r$ at any time $\zeta$ not less than $\tau$. We denote with $\langle\!|\aleph|\!\rangle$ the translation derived from that presented in Definition 4.1 with this extra feature, that allows us to ignore the evaluation time in defining the logic program associated to $\aleph$.

Two further technical devices are needed for CIFF to work properly.

1. Firstly, the clauses must be given as so-called *iff-definitions*. Intuitively, given a general logic program $\mathtt{P}$, the iff-definition associated with it is obtained by grouping all the clauses for the same atom $r(\cdot,\cdot,\cdot)$ in $\mathtt{P}$

$$r(\cdot,\cdot,\cdot) :- D_1 \quad , \quad \cdots \quad , \quad r(\cdot,\cdot,\cdot) :- D_k$$

into the iff-definition

$$r(\cdot,\cdot,\cdot) \ :-: \ D_1 \vee \cdots \vee D_k$$

where $D_i$ denotes a conjunction of literals and constraints. The resulting set of iff-definitions is denoted as *IFF*$(\mathtt{P})$. Clearly, to do so, we first need to remove any constant from the left-hand side of a general logic clause; this can be easily done by using a (new) variable in the definition and adding an equality constraint stating that the new variable must be equal to the old constant. For example,

$$r(A,B,\zeta) :- \zeta \geq \tau$$

becomes

$$r(\xi_1,\xi_2,\zeta) :- \xi_1 = A, \xi_2 = B, \zeta \geq \tau.$$

This task can be easily carried out automatically.

2. The second technical point required by the CIFF proof procedure is to clearly distinguish predicates that are abducible from those defined by the given logic program. This is needed to ensure that all the results returned by the procedure are minimal. To this aim, we exploit the integrity constraints and add to them (that are initially empty) the implication

$$\delta_r(\xi_1,\xi_2,\zeta) \Rightarrow r(\xi_1,\xi_2,\zeta)$$

where $\delta_r$ is a new relation symbol. We denote by $IC(\mathtt{P})$ the resulting set of integrity constraints and let $\delta_r$ be an abducible predicate.

As a consequence, we need to translate the given goal $A.r \leftarrow B$ accordingly, i.e. as $\delta_r(A,B,\zeta)$ where here $\zeta$ is meant to be *existentially* quantified. We denote with $G(A.r \leftarrow B)$ such a translation. Intuitively, an abducible explanation for $G(A.r \leftarrow B)$ induces a time validity (derived from the constraints over $\zeta$) and some knowledge that must be provided by the environment (derived from the abducible predicates) under which the goal can be inferred.

The input of CIFF is derived from a given set of CDCs and a goal $A.r \leftarrow B$ and takes the form

$$(\mathit{IFF}(\langle\!|\aleph|\!\rangle); \mathit{IC}(\langle\!|\aleph|\!\rangle); G(A.r \leftarrow B))$$

as described above. The CIFF proof procedure essentially manipulates sets of formulas that are either atoms or implications. It calculates a tree of such sets (thus called nodes) with the property that each node is obtained from its father by applying one rule to the formulas occurring in the father. There are several rules; for a precise discussion of them, see [10]. The procedure terminates when all nodes are *final*, i.e. they contain sets of formulas to which no more rule can be applied. A final node is *successful* if it does not contain contradictions; otherwise, it is called *failed*.

The root of the tree is formed by the goal and by all the integrity constraints. The output of CIFF is a set of pairs of the form $(\mathtt{P};\Gamma)$, one for each successful node of the computed tree; $\mathtt{P}$ contains the abducible atoms in the node and $\Gamma$ its constraints. It has been proved in [10] that, for every returned $(\mathtt{P};\Gamma)$, there exists an abducible explanation of the goal $\delta_r(A,B,\zeta)$ in $\langle\!|\aleph|\!\rangle$; such an explanation is $\mathtt{P}'\sigma$, where $\mathtt{P}'$ is obtained from $\mathtt{P}$ by removing $\delta_r(A,B,\zeta)$ and $\sigma$ is any substitution of values for variables satisfying $\Gamma$. Vice versa, in [10] it is also proved that, if all the derivations in CIFF are finite and failing, then no abducible explanation for the given input exists.

EXAMPLE 7 (Example 2, by means of CIFF). To conclude, let us show in some detail how Example 2 is handled by CIFF; this should also informally illustrate the rules underlying CIFF. The original setting provides the set of CDCs $\{(13),(14)\}$ and the goal (15),

with $v = [0, 10]$ in (14), for example. This is translated to the following input for CIFF:[3]

$$(\mathtt{P}; \mathtt{IC}; \delta_r(A, B, \zeta))$$

where P is the following general logic program in iff-form, associated to $\{(13), (14)\}$

$$\mathtt{P} \triangleq \left\{ \begin{array}{l} r(\xi_1, \xi_2, \zeta) :-: \ \xi_1 = A, s(E, \xi_2, \zeta), \neg a(E, A, \zeta) \\ a(\xi_3, \xi_4, \zeta) :-: \ \xi_3 = E, \xi_4 = A, \zeta \geq 0, \zeta \leq 10 \end{array} \right\}$$

and IC is the following set of integrity constraints.

$$\mathtt{IC} \triangleq \left\{ \begin{array}{l} \delta_r(\xi_1, \xi_2, \zeta) \ \Rightarrow \ r(\xi_1, \xi_2, \zeta) \\ \delta_a(\xi_3, \xi_4, \zeta) \ \Rightarrow \ a(\xi_1, \xi_2, \zeta) \end{array} \right\}$$

CIFF now builds the following tree. The root node is

$$N_0 : \quad \delta_r(A, B, \zeta) \ , \ \mathtt{IC}.$$

It then uses $\delta_r(A, B, \zeta)$ and the first integrity constraint from IC (by means of unification and *modus ponens*) to derive

$$N_1 : \quad \delta_r(A, B, \zeta) \ , \ r(A, B, \zeta) \ , \ \mathtt{IC}.$$

It then replaces $r(A, B, \zeta)$ with the body of its iff-definition in P and derives

$$N_2 : \quad \delta_r(A, B, \zeta), A = A, s(E, B, \zeta), \neg a(E, A, \zeta), \mathtt{IC}$$

that can be simplified to

$$N_3 : \quad \delta_r(A, B, \zeta), s(E, B, \zeta), \neg a(E, A, \zeta), \mathtt{IC}.$$

Then, it replaces $a(E, A, \zeta)$ with the body of its iff-definition in P and derives

$$\begin{array}{ll} N_4 : & \delta_r(A, B, \zeta), s(E, B, \zeta), \mathtt{IC}, \\ & \neg(E = E, A = A, \zeta \geq 0, \zeta \leq 10). \end{array}$$

This node has four children; two of them are failure nodes

$$\begin{array}{ll} N_5^1 : & \delta_r(A, B, \zeta), s(E, B, \zeta), E \neq E, \mathtt{IC} \\ N_5^2 : & \delta_r(A, B, \zeta), s(E, B, \zeta), A \neq A, \mathtt{IC} \end{array}$$

while the other two ones are successful

$$\begin{array}{ll} N_5^3 : & \delta_r(A, B, \zeta), s(E, B, \zeta), \zeta < 0, \mathtt{IC} \\ N_5^4 : & \delta_r(A, B, \zeta), s(E, B, \zeta), \zeta > 10, \mathtt{IC}. \end{array}$$

From $N_5^3$ and $N_5^4$, we infer that an abducible explanation for *A.read* $\leftarrow$ $B$ is the context $\langle \{\textit{Ent.secr} \leftarrow B\}; \tau \rangle$, for $\tau \in (-\infty, 0) \cup (10, +\infty)$; this exactly coincides with (16). □

## 6. Conclusions and Related Work

The main contribution of our work is the extensions we have proposed for introducing dynamic considerations into $RT_0$. The availability of CDCs is intermittent, either because their time validity can (temporarily or permanently) expire or because formerly available CDCs, required to satisfy their guards, can become (temporarily or permanently) not available. This feature reflects timed privileges [28, 23, 26] and consequently makes mechanisms to explicitly introduce/remove credentials redundant.

An important source of expressiveness of an access control model is the temporal dimension that permissions have in many real-world situations: permissions are often limited in time or may hold only for specific periods of time. Moreover, permissions can also be issued/revoked according to the context where they are calculated. Context-dependent credentials, presented in this paper, are a simple but powerful way to model both these features.

[3] To save space, we have shortened *Alice* as $A$, *Bob* as $B$, *Ent* as $E$, *active* as $a$, *readMail* as $r$ and *secr* as $s$.

It has to be said that the temporal dimension is present in the $RT$ family from its birth (see the language $RT_1$ in [20]). However, every form of negation has been always intentionally omitted, to keep the semantical development of the language simple. As shown in the examples throughout this paper, we believe that some policy specifications intrinsically rely on negative requirements; thus, we believe that the use of negation will have to be confronted if the goal is a highly flexible model capable of supporting the specification of complex protection requirements.

Unfortunately, the presence of negation creates problems when defining the semantics of a language. Usually, there are legal terms that either have no semantics or whose semantics is not uniquely definable. The stable model approach assigns a semantics to all those terms whose derivability does not depend on themselves as assumptions. As an example, in our framework every set of CDCs containing **if** $B \notin A.r$ **then** $A.r \leftarrow B$ **in** $v$ will have no (stable model) semantics. However, this is acceptable, since the previous CDC must not be read as "if $B$ is *still* not a member of $A.r$, then include $B$ in $A.r$" (that could be meaningful, in some cases) but it must be read as "$B$ is a member of $A.r$ whenever $B$ is not a member of $A.r$", that sounds contradictory. Similarly, sets of CDCs containing **if** $B \notin A.r$ **then** $C.s \leftarrow D$ **in** $v$ and **if** $D \notin C.s$ **then** $A.r \leftarrow B$ **in** $v$ have similar problems. However, we believe that policy specifications relying on these kinds of CDCs are 'ill-formed' and should not be considered in the implementation of actual security systems.

Finally, we want to stress the usefulness of the inference systems in Section 5.1: it is really desirable to have automatic tools that assist in the definition of security specifications. The inference systems we have presented in this paper are simple, but theoretically well-founded, aids to the definition of the proper validity of certificates, or of the contextual information required for the proper functioning of a set of certificates. This turns out to be fundamental mainly in large-scale distributed systems where users only have partial views of their execution context.

***Related work.*** A different approach that makes the policies defined by $RT_0$-credentials dynamic is given in [29]; this contains a security-typed imperative language whose main feature is the possibility of programming policy modifications. The main focus of the paper is on controlling information flow, but security levels are expressed by means of rôles, whose membership is made dynamic by the possibility of modifying rôle-definitions during execution. This approach embeds policy modifications for $RT_0$ into a full-fledged programming language and exploits the resulting framework for purposes different from ours; for this reason, they take an orthogonal approach and exploit type systems to rule out unwanted information flow.

*Default logic* [25] is one of the pioneering work in the field of non-monotonic reasoning (an example of which is also logic programming with negation). In default logic, there are ordinary inference rules and *default rules*, that are triples of the form *"(pre-requisites, justifications, conclusion)"* stating that the conclusion can be derived from the pre-requisites, provided that there is no evidence that the justifications might be false. To increase expressiveness, both pre-requisites and justifications can contain negative judgements; hence, also in default logic there are problems in giving semantics to a set of default rules: there can be zero, one or more than one possible semantics for a given set of rules. Again, like in logic programming, several choices could be taken to solve this problem, one of which strongly resembles the stable model approach. Default rules are quite similar to the rule ($\mathrm{CDC}_1$), where the positive premises correspond to pre-requisites and negative premises to justifications.

A somewhat related work is [5] where an access control model with periodic temporal intervals associated to authorisations is

given. An authorisation is automatically granted in the specified intervals and revoked when such intervals expire. Deductive temporal rules with periodicity are provided to derive new authorisations based on the presence or absence of other authorisations in specific periods of time. Like in our approach, possible inconsistencies deriving from negative requirements are handled by the stable models approach; however, [5] does not consider the powerful form of delegation put forward by linked rôles, such as $A.r.s$ in $RT_0$.

$RT_0$ is the most basic language of the $RT$ family. In [20], all the members of the family are presented: $RT_1$ adds to $RT_0$ parameterised rôles, which can express attribute fields; $RT_2$ adds to $RT_1$ logical objects, which can group logically related objects together so that permissions about them can be assigned together; $RT^T$ provides manifold rôles and rôle-product operators, which can express threshold and separation-of-duty policies; $RT^D$ provides delegation of rôle activations, which can express selective use of capacities and delegation of these capacities. The semantics of all these languages is given via a translation from credentials to negation-free logic programs; thus, we do not foresee any problem in applying our enhancements of $RT_0$ to the other members of the $RT$ family.

To conclude, notice that in this paper we have only considered what in [21] is called 'membership queries', that is, our guards only test whether a given entity belongs to a given rôle or not. More sophisticated queries are considered in that paper and could be integrated in our framework; for example, two other reasonable guards could be $A.r \subseteq \{B_1, \ldots, B_k\}$ and $A.r \subseteq B.s$, or their negations. We leave such an integration for future work.

***Acknowledgements*** We would like to thank Catuscia Palamidessi for several pointers and suggestions in the field of logic programming. The anonymous reviewers helped in improving the quality of the paper.

## References

[1] The CIFF web-page. `http://www.doc.ic.ac.uk/~ue/ciff/`.

[2] K. R. Apt. Logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 493–574. Elsevier and MIT Press, 1990.

[3] K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, 1988.

[4] K. R. Apt and R. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 20:9–71, 1994.

[5] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM Trans. on Database Systems*, 23(3):231–285, 1998.

[6] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralised trust management. In *IEEE Symposium on Security and Privacy*, 1996.

[7] R. N. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, 1996.

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

[9] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. IETF RFC 2693, 1999.

[10] U. Endriss, P. Mancarella, F. Sadri, G. Terreni, and F. Toni. The CIFF proof procedure for abductive logic programming with constraints. In *Proc. of 9th Europ. Conf. on Logics in Artificial Intelligence (JELIA'04)*, volume 3229 of *LNCS*, pages 31–43. Springer, 2004.

[11] F. Fages. A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics. *New Generation Computing*, 9(4):425–444, 1991.

[12] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[13] T. H. Fung and R. A. Kowalski. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming*, 33(2):151–165, 1997.

[14] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of 5th Intern. Conf. on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[15] D. Gorla, M. Hennessy, and V. Sassone. Inferring dynamic credentials for role-based trust management. Technical Report 04/2006, Dip. di Informatica, Univ. di Roma "La Sapienza".

[16] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.

[17] A. C. Kakas and P. Mancarella. Generalized stable models: a semantics for abduction. In *Proc. of 9th European Conference on Artificial Intelligence*, pages 385–391, 1990.

[18] A. C. Kakas, A. Michael, and C. Mourlas. Aclp: Abductive constraint logic programming. *Journal of Logic Programming*, 44(1-3):129–177, 2000.

[19] N. Li and J. C. Mitchell. Understanding SPKI/SDSI using first-order logic. In *Proc. of CSFW*, pages 89–103. IEEE Computer Society, 2003.

[20] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[21] N. Li, W. H. Winsborough, and J. C. Mitchell. Beyond proof-of-compliance: Safety and availability analysis in trust management. In *IEEE Symposium on Security and Privacy*, pages 123–139. IEEE Computer Society, 2003. Full version to appear in *JACM*.

[22] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, 2003. An extended abstract appeared in the *Proc. of CCS'01*, ACM.

[23] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proc. of 7th USENIX Security Symposium*, pages 217–228, 1998.

[24] T. C. Przymusinski. On the declarative semantics of logic programs with negation. In *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, 1988.

[25] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[26] R. L. Rivest. Can we eliminate certificate revocation lists? In *Proc. 2nd Int. Conf. on Financial Cryptography*, volume 1465 of LNCS, pages 178–183. Springer, 1998.

[27] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[28] S. G. Stubblebine. Recent-secure authentication: enforcing revocation in distributed systems. In *IEEE Symposium on Security and Privacy*, pages 224–235. IEEE Computer Society, 1995.

[29] N. Swamy, M. Hicks, S. Tse, and S. Zdancewic. Managing policy updates in security-typed languages. In *Proc. of the 19th CSFW*. IEEE Computer Society, 2006.

[30] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[31] A. Van Gelder, K. Ross, and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.