A Linear Logic of Authorization and Knowledge[⋆]

Deepak Garg, Lujo Bauer, Kevin D. Bowers,
Frank Pfenning and Michael K. Reiter

Carnegie Mellon University

Abstract. We propose a logic for specifying security policies at a very high level of abstraction. The logic accommodates the subjective nature of affirmations for authorization and knowledge without compromising the objective nature of logical inference. In order to accurately model consumable authorizations and resources, we construct our logic as a modal enrichment of linear logic. We show that the logic satisfies cut elimination, which is a proof-theoretic expression of its soundness. We also demonstrate that the logic is amenable to meta-reasoning about specifications expressed in it through several examples.

1 Introduction

In this paper we develop a logic for specifying security properties of distributed systems at a very high level of abstraction. One of the difficulties in this domain is that security specifications, by nature, depend on individuals' intent as well as their state of knowledge. In addition to logical inference regarding the *truth* of propositions, we therefore also need to reason with *affirmations* of principals (to express intent), and *knowledge* of principals. In addition, we often need to capture changes of state, such as transfer of money or goods, which is most easily expressed in linear logic. We therefore arrive at a linear logic with additional modal operators for affirmation and knowledge, indexed by principals. We believe the combination of linearity with modalities such as affirmation and knowledge is an original contribution of this paper with some new insights, such as how to model possession of consumable resources as linear knowledge, or single-use authorizations as linear affirmations.

We show that our logic satisfies cut elimination, which is a proof-theoretic expression of its soundness. Moreover, the cut elimination theorem shows that the various components of the logic are orthogonal and, for example, there is a coherent subsystem containing only affirmations and not knowledge. We illustrate our logic through two examples. The first concerns a student registration system and demonstrates how to represent use-once authorizations, but avoids the use of knowledge. The second is a specification of monetary instruments which employs affirmations expressing authorization and knowledge to model possession

⋆ This work is supported by grants N0014-04-1-0724 of the Office of Naval Research, DAAD19-02-1-0389 of the Army Research Office and CNS-0433540 of the National Science Foundation.

of resources. In both examples we show how to exploit the formal foundation of our logic in order to reason about properties of specifications expressed in it. In the student registration example we show that various constraints, such as the maximal number of credits a student can sign up for, are respected. In the monetary examples, we verify balance conditions on the total amount of money in the bank or under the control of principals. These meta-theoretic analyses rely again on cut elimination and a somewhat deeper property, namely completeness of focusing.

While beyond the scope of the present paper, some prior work on proof-carrying authorization [7, 8] suggests that fragments of our logic would be a suitable basis for policy enforcement in a distributed architecture.

We now present the logic in two steps, first reviewing affirmation as developed in [14] in the new context of linearity. We then add possession and knowledge, followed by a brief sketch of the meta-theory of our logic and the examples. We conclude with additional related work and future plans.

2 A Constructive Linear Logic of Affirmation

When designing a new specification logic, we have to consider the range of properties we would like to express. First, we realize that we need standard logical connectives such as conjunction and implication. These are concerned with the truth of propositions, which is therefore our first judgment, $A\ \mathsf{true}$. We also need reasoning from hypotheses, so our basic judgment form is a hypothetical judgment. One might say that this constitutes the objective part of the logic since everyone agrees on the laws of logical reasoning and therefore on the meaning of the connectives.

Second, we need a way for principals to express their intent, such as who may access some information they have. We call this judgment *affirmation*, written as $K\ \mathsf{affirms}\ A$ (principal K affirms proposition A). The affirmation of a proposition does not imply its truth, otherwise everyone could give themselves access to any resource simply by affirming this. For example, a principal may affirm that she has access to a certain file on the disk. This is an expression of her intent; in truth she will not have access to the file unless the security policy allows it. On the other hand, if A is true then all principals are willing to affirm A since we assume principals are rational and can verify evidence for A.

In an implementation, we imagine affirmations can be established in two ways: cryptographically via certificates containing A signed by K, and logically via a deduction proving that A is true. The combination of signed certificates and logical proofs is the foundation of proof-carrying authorization [6, 7].

We would like to go a step further and allow affirmations that may be used only once. For example, a personal check for \$10 from K made out to L can be seen as an affirmation that K is prepared to pay L the sum of \$10. However, this affirmation can be used by L only once; L cannot be allowed to cash the check multiple times. In logical terms this means that the certificate is *linear*, and that our logic will be an enrichment of linear logic. Linear logic is characterized by a linear hypothetical judgment where each linear assumption must be used exactly once and therefore represents a consumable resource. Of course, we also still need assumptions whose use is unrestricted for reusable certificates. From the point of view of linear logic, these assumptions are of the form A valid, because their proof cannot depend on any linear resources.

Putting these observations together, we obtain the judgment forms

$$\begin{array}{l} \Gamma; \Delta \Longrightarrow A \;\mathsf{true} \\ \Gamma; \Delta \Longrightarrow K \;\mathsf{affirms}\; A \end{array}$$

where Δ consists of linear (use-once) hypotheses A true and Γ consists of unrestricted (reusable) hypotheses A valid. It turns out that we do not need to explicitly consider conclusions of the form A valid or hypotheses of the form K affirms A because they can always be eliminated.

The first set of rules just captures the nature of the hypothetical judgments. Because A true or A valid can always be inferred from their position in the sequent, we will generally abbreviate these judgments to just A. The short-hand γ stands for judgments we consider on the right, which are either A true or K affirms A.

$$\frac{}{\Gamma; A \Longrightarrow A}(\mathtt{init}) \qquad\qquad \frac{\Gamma, A; \Delta, A \Longrightarrow \gamma}{\Gamma, A; \Delta \Longrightarrow \gamma}(\mathtt{copy})$$

Next, the rules pertaining to the affirmation judgment. Because we do not consider hypotheses of the form K affirms A, there is only one rule which states that any principal K is prepared to affirm any true proposition.

$$\frac{\Gamma; \Delta \Longrightarrow A}{\Gamma; \Delta \Longrightarrow K \;\mathsf{affirms}\; A}(\mathsf{affirms})$$

Next, we internalize affirmation as a propositional modal operator so that we can combine affirmations with logical connectives such as implication. We write $\langle K \rangle A$ for the proposition that internalizes K affirms A. In a sequent calculus connectives are characterized by left and right rules.

$$\frac{\Gamma; \Delta \Longrightarrow K \;\mathsf{affirms}\; A}{\Gamma; \Delta \Longrightarrow \langle K \rangle A}(\langle\rangle R) \qquad\qquad \frac{\Gamma; \Delta, A \Longrightarrow K \;\mathsf{affirms}\; C}{\Gamma; \Delta, \langle K \rangle A \Longrightarrow K \;\mathsf{affirms}\; C}(\langle\rangle L)$$

While the right rule is straightforward, the left rule is key to understanding the modal nature of affirmation. Observe that in order to apply the left rule to $\langle K \rangle A$, the succedent of the sequent must be an affirmation by the same principal K. This means we can move from the truth of $\langle K \rangle A$ to the truth of A, but only if we are reasoning about the affirmations of K. This captures that K is rational.

3 Possession and Knowledge

The next step is to introduce knowledge into our logic. We are not aware of any attempts to combine epistemic logic with linear logic, so we believe this to be a contribution of this paper of independent interest.

One assumption commonly made about knowledge is that it is monotonic: we may learn more, for example, by inference, but we do not forget. Then what is "linear knowledge"? Returning to the usual interpretation of linear logic, we consider a linear assumption A true as a *resource* that may be consumed in a proof. Then linear knowledge is nothing but *possession of a resource* that may be consumed in a proof. We therefore write K has A for the new judgment of possession which is linear. It turns out we can always eliminate possession in the succedent of a sequent, so there is only one judgmental rule.

$$\frac{\Gamma; \Delta, A \Longrightarrow \gamma}{\Gamma; \Delta, K \;\mathsf{has}\; A \Longrightarrow \gamma}(\mathsf{has})$$

Informally, it states that if K possesses A then A may be used a resource. We have to take care, however, to make sure that other principals cannot steal the resource.

We internalize possession as a proposition, $[K]A$, expressing that it is true that K possesses A. The hypothetical nature of sequents means that a proposition $[K]A$ in the succedent expresses *potential possession*, where a proof corresponds to a plan to achieve this position. For example, the sequent

$$\cdot; K \;\mathsf{has}\; (B \multimap A), K \;\mathsf{has}\; B \Longrightarrow [K]A$$

could be read as: *If K has a means to transform a resource B into a resource A, and K also has resource B, then it is true that K can obtain resource A.* Of course, resources B and $B \multimap A$ would be consumed in the process of obtaining A, since those resources are linear.

The right rule expresses that, in order to show that K could obtain resource A, we have to show that resource A can be obtained using *only* the resources in Γ and Δ that K possesses *and no others*. This requires a restriction operator that removes from a context Δ all assumptions that are not of the form K has A, and similarly for the unrestricted context. In comparison, the left rule for $[K]A$ is straightforward, just transforming the proposition to the corresponding judgment.

$$\frac{\Gamma|_K; \Delta|_K \Longrightarrow A}{\Gamma; \Delta|_K \Longrightarrow [K]A}([]R) \qquad\qquad \frac{\Gamma; \Delta, K \;\mathsf{has}\; A \Longrightarrow \gamma}{\Gamma; \Delta, [K]A \Longrightarrow \gamma}([]L)$$

The restriction operator on linear assumptions[1] is defined formally as

$$\begin{array}{ll}
(\cdot)|_K & = \cdot \\
(\Delta, K \;\mathsf{has}\; A)|_K & = \Delta|_K, K \;\mathsf{has}\; A \\
(\Delta, L \;\mathsf{has}\; A)|_K & = \Delta|_K \quad \text{for } L \neq K \\
(\Delta, A \,\mathsf{true})|_K & = \Delta|_K
\end{array}$$

This means in the $[]R$ rule above, the linear context in the premise and the conclusion must contain only assumptions of the form K has B for the given K but possibly distinct B.

[1] The corresponding operator on unrestricted assumptions Γ is given below.

We also need more traditional knowledge of propositions, subject to unrestricted reuse. We write this judgment as K knows A. By its nature, it belongs in the context Γ. Again, knowledge is only required in assumptions so we have only one rule pertaining directly to the judgment. It allows us to infer the truth of A given K's knowledge of A. Of course, the converse must be prohibited.

$$\frac{\Gamma, K \;\textsf{knows}\; A; \Delta, A \Longrightarrow \gamma}{\Gamma, K \;\textsf{knows}\; A; \Delta \Longrightarrow \gamma}(\textsf{knows})$$

Internalizing knowledge in the same way as possession, keeping in mind its unrestricted nature, we obtain the following two rules.

$$\frac{\Gamma|_K; \cdot \Longrightarrow A}{\Gamma; \cdot \Longrightarrow [\![K]\!]A}([\![]\!]R) \qquad\qquad \frac{\Gamma, K \;\textsf{knows}\; A; \Delta \Longrightarrow \gamma}{\Gamma; \Delta, [\![K]\!]A \Longrightarrow \gamma}([\![]\!]L)$$

The first expresses that K can obtain knowledge of A if it follows by logical reasoning from the knowledge that K already has and no other assumptions. Formally, restriction is defined as follows.

$$\begin{array}{ll}
(\cdot)|_K & = \cdot \\
(\Gamma, K \;\textsf{knows}\; A)|_K & = \Gamma|_K, K \;\textsf{knows}\; A \\
(\Gamma, L \;\textsf{knows}\; A)|_K & = \Gamma|_K \quad \text{for } L \neq K \\
(\Gamma, A \,\textsf{valid})|_K & = \Gamma|_K
\end{array}$$

This concludes our introduction to the basic logic, omitting only the standard connectives, both linear and non-linear. Appendix A summarizes the sequent calculus, including the standard connectives. We include the first-order universal quantifier because we need it to encode the examples in section 5. All results of the next section extend to the first-order case easily.

4 Cut Elimination

In a sequent calculus the connectives are explained via their right and left rules. Since propositions are always decomposed in such rules when read from the conclusion to the premises, we are justified in saying that the meaning of propositions is determined by their proofs, but only if the underlying interpretation of the sequent as a hypothetical judgment is respected. This is the contents of two important theorems: the admissibility of cut and the identity principle. Admissibility of cut expresses that we can always eliminate an assumption $A \,\textsf{true}$ if we can supply a proof of A $\textsf{true}$. The identity principle states that we only need the (`init`) rule $\Gamma; A \Longrightarrow A$ for the case where A is atomic. To prove these we need to state them in a more general form to account for the other judgments in our logic. Other properties, such as weakening and contraction for the unrestricted context are immediate and we don't state them explicitly.

Theorem 1 (Admissibility of cut).

1. *If* $\Gamma; \Delta \Longrightarrow A$ *and* $\Gamma; \Delta', A \Longrightarrow \gamma$ *then* $\Gamma; \Delta', \Delta \Longrightarrow \gamma$.
2. *If* $\Gamma; \cdot \Longrightarrow A$ *and* $\Gamma, A; \Delta' \Longrightarrow \gamma$ *then* $\Gamma; \Delta' \Longrightarrow \gamma$.
3. *If* $\Gamma; \Delta \Longrightarrow K\ \mathsf{affirms}\ A$ *and* $\Gamma; \Delta', A \Longrightarrow K\ \mathsf{affirms}\ C$ *then* $\Gamma; \Delta, \Delta' \Longrightarrow K\ \mathsf{affirms}\ C$.
4. *If* $\Gamma|_K; \Delta|_K \Longrightarrow A$ *and* $\Gamma; \Delta', K\ \mathsf{has}\ A \Longrightarrow \gamma$ *then* $\Gamma; \Delta', \Delta|_K \Longrightarrow \gamma$.
5. *If* $\Gamma|_K; \cdot \Longrightarrow A$ *and* $\Gamma, K\ \mathsf{knows}\ A; \Delta' \Longrightarrow \gamma$ *then* $\Gamma; \Delta' \Longrightarrow \gamma$.

Proof. By nested induction, first on the structure of the cut formula A and then on the size of the two given derivations, as in [18, 11].

Theorem 2 (Identity). *In the sequent calculus where initial sequents are restricted to atomic propositions,* $\Gamma; A \Longrightarrow A$ *for any proposition* A.

Proof. By induction on the structure of A.

5 Examples and Reasoning About Policies

We present two examples of security policies expressed in our logic using linear authorization and knowledge. The first one uses linear authorizations to describe a university course registration system. In the second example, we use linear knowledge and authorizations to represent a system of monetary instruments like checks, promissory notes and bank accounts. We reason about interesting properties of these systems (like correctness with respect to a given specification) using the logic. Some of the methods used for reasoning can be generalized beyond these examples. We return to this point briefly at the end of the section.

5.1 Course Registration

This example describes a university registration system using linear authorizations. Some of the authorizations in this example may be replaced by linear possessions, but we do not do this to keep the example simple. We assume two main principals: a `calendar` which authorizes free time slots available for students, and a `registrar` who controls the entire registration process. The following table lists the predicates we use along with their intuitive meanings.

$\texttt{slot}(S, T)$	Student S is free during time slot T
$\texttt{credits}(S, R)$	Student S may register for R more credits in the semester
$\texttt{registered}(S, C, R, T)$	Student S is registered in course C for R credits in time slot T
$\texttt{seats}(C, N)$	There are N more seats available in course C
$\texttt{course}(C, R, T)$	Course C is worth R credits and runs in time slot T

We wish to enforce three conditions during registration:

1. No student registers for more than a stipulated number of credits.

2. A student does not register for two courses that use the same time slot.
3. A maximum registration limit for each course is respected.

In the logic, linear authorizations are represented as assumptions of the form $\langle K \rangle A$ in the linear context Δ. Authorizations meant for unrestricted use are represented as assumptions of the same form in the context Γ. In an implementation, these assumptions are substituted by certificates signed by the authorizing principals.

We assume that at the beginning of the semester a number of certificates are issued by the registrar and the calendar, i.e., we assume that a number of authorization assumptions are present in our context when we start reasoning. These are the following. For each student S there is one linear certificate of the form $\langle \mathtt{registrar} \rangle \mathtt{credits}(S, R)$ issued by the registrar. This certificate mentions the maximum number of credits R that the student is permitted to take during the semester. For each possible time slot T, each student S gets one certificate of the form $\langle \mathtt{calendar} \rangle \mathtt{slot}(S, T)$ from the calendar. This entitles the student to register for some course in time slot T.

For each course C, the registrar issues a linear certificate of the form $\langle \mathtt{registrar} \rangle \mathtt{seats}(C, N)$, that specifies the number of seats N in the course. The registrar also issues one *unrestricted* certificate for each course C. This certificate specifies the number of credits R the course is worth, and the time slot T in which it runs. It has the form $\langle \mathtt{registrar} \rangle \mathtt{course}(C, R, T)$.

Now we state the policy rule governing registration in courses. The rule is universally quantified over the terms S, N, R, R', C and T.

$$\begin{array}{l}
\mathtt{reg} : \langle \mathtt{registrar} \rangle \mathtt{course}(C, R, T) \supset \\
\quad \langle \mathtt{registrar} \rangle \mathtt{seats}(C, N) \multimap \\
\quad \langle \mathtt{registrar} \rangle \mathtt{credits}(S, R') \multimap \\
\quad (N \geq 1) \supset (R' \geq R) \supset \\
\quad \langle \mathtt{calendar} \rangle \mathtt{slot}(S, T) \multimap \\
\qquad (\langle \mathtt{registrar} \rangle \mathtt{registered}(S, C, R, T) \otimes \\
\qquad \langle \mathtt{registrar} \rangle \mathtt{credits}(S, R' - R) \otimes \\
\qquad \langle \mathtt{registrar} \rangle \mathtt{seats}(C, N - 1))
\end{array}$$

Intuitively, this rule says the following: if course C, worth R credits and running in time slot T, has at least one seat available, and student S can register for at least R more credits during the semester and is free during time slot T, then S may register for the course C. The rule consumes the credential $\langle \mathtt{registrar} \rangle \mathtt{credits}(S, R')$, replacing it with a similar credential that decrements R' by the number R of credits that the course is worth. This enforces condition (1) above. The rule also consumes S's time slot credential corresponding to the course's time slot T to prevent her from registering in another course that runs in the same slot. This enforces condition (2). Condition (3) is enforced because the rule replaces the credential $\langle \mathtt{registrar} \rangle \mathtt{seats}(C, N)$ with $\langle \mathtt{registrar} \rangle \mathtt{seats}(C, N - 1)$. This reduces the number of seats available in the course by one.

Observe that there is no condition in the rule that represents *intent* of student S to register for course C. This is because we are interested in expressing only the security aspects of the system in the logic. If this rule were to be implemented, say using a protocol, it would be necessary to ensure that the rule is used only when student S is actually willing to register for the course C.

Atomicity in policy implementation. In any realistic implementation of the above policy, it is essential that the policy rule `reg` be used *atomically*, i.e., in any application of the rule, all its pre-conditions be satisfied simultaneously.[2] This is significant due to linearity, because proving some pre-conditions of the rule may utilize linear resources, and a partial application of the rule may result in a situation where linear resources are incorrectly consumed.

A simple example illustrates this point. Suppose a student S wishes to register for course C worth R credits in time slot T and the following hold: (a) the course has a seat (there is a certificate $\langle\texttt{registrar}\rangle\texttt{seats}(C, N)$, where $N \geq 1$), (b) the student S has sufficient number of available credits (she has a certificate $\langle\texttt{registrar}\rangle\texttt{credits}(S, R')$ where $R' \geq R$), and (c) S does *not* have the required time slot certificate $\langle\texttt{calendar}\rangle\texttt{slot}(S, T)$. If we were to permit non-atomic use of the policy rule, we could consume the linear certificates mentioned in (a) and (b) to conclude the following:

$$\begin{array}{l}\langle\texttt{calendar}\rangle\texttt{slot}(S,T) \multimap \\ \quad (\langle\texttt{registrar}\rangle\texttt{registered}(S,C,R,T) \otimes \\ \quad \langle\texttt{registrar}\rangle\texttt{credits}(S,R'-R) \otimes \\ \quad \langle\texttt{registrar}\rangle\texttt{seats}(C,N-1))\end{array}$$

However, since the student does not have the required time slot certificate $\langle\texttt{calendar}\rangle\texttt{slot}(S, T)$, we cannot proceed any further. At this point, the system is stuck in an inconsistent state because two linear certificates mentioning the number of seats in course C and student S's available credits have been consumed. No student can register in course C now, and S cannot register for any other course. Thus it is essential that the policy rule above (and in general, any policy rule that uses linear resources), be enforced atomically in an implementation.

Atomicity in the logic. If we can enforce atomicity of policy rules at the level of the logic itself, we can reason more faithfully about the consequences of policies using the logic. In particular, we can prove useful invariance properties of the system as it evolves under a particular policy. Since atomicity is an artifact of the implementation, the method used to enforce it in the logic should not affect logical consequence or provability in the logic. One such method is focusing [5], which is a proof search technique that combines a number of inference steps atomically without affecting provability. A detailed description of focusing is beyond the scope of this paper. In the following we present it only to the extent that is appropriate for our purpose. We convert each policy rule into a derived inference rule, which we add to the logic. Atomicity is forced implicitly

[2] The pre-conditions of a rule $A_1 \multimap \ldots A_n \multimap B$ are $A_1, \ldots, A_n$.

by the inference rule. For example, the rule `reg` can be converted to the following inference rule.

$$\dfrac{\begin{array}{c} \Gamma; \cdot \Longrightarrow \langle\mathtt{registrar}\rangle\mathtt{course}(C, R, T) \\ \Gamma; \Delta_1 \Longrightarrow \langle\mathtt{registrar}\rangle\mathtt{seats}(C, N) \\ \Gamma; \Delta_2 \Longrightarrow \langle\mathtt{registrar}\rangle\mathtt{credits}(S, R') \\ \Gamma; \cdot \Longrightarrow N \geq 1 \\ \Gamma; \cdot \Longrightarrow R' \geq R \\ \Gamma; \Delta_3 \Longrightarrow \langle\mathtt{calendar}\rangle\mathtt{slot}(S, T) \\ \begin{array}{l} \Gamma;\ \Delta_4, \langle\mathtt{registrar}\rangle\mathtt{registered}(S, C, R, T), \\ \quad \langle\mathtt{registrar}\rangle\mathtt{credits}(S, R' - R), \langle\mathtt{registrar}\rangle\mathtt{seats}(C, N - 1) \end{array} \Longrightarrow \gamma \end{array}}{\Gamma; \Delta_1\Delta_2\Delta_3\Delta_4 \Longrightarrow \gamma}\ (\mathtt{reg})$$

Read bottom up, the rule says that we can conclude γ using the linear resources $\Delta_1\Delta_2\Delta_3\Delta_4$, if we can prove the preconditions of the rule `reg` using Δ_1, Δ_2 and Δ_3, and use Δ_4 and the three authorizations produced by `reg` to prove γ. This is exactly the behavior of the rule `reg` if it were implemented atomically.

System states and steps. We now formalize a notion of system state and state transition (step) for our example. Once we have these definitions we can prove that certain properties of system states are invariant under steps. These properties can be used to establish that the policy satisfies the three conditions mentioned at the beginning of the example. Informally, a state of the system is a pair of contexts $\Gamma; \Delta$ that contains only those authorizations that are relevant to our example.

Definition 2 (State) *A state of the system is a pair of contexts $\Gamma; \Delta$, satisfying the following conditions.*

1. *All assumptions in Γ have the form $\langle\mathtt{registrar}\rangle\mathtt{course}(C, R, T)$.*
2. *All assumptions in Δ have one of the forms $\langle\mathtt{registrar}\rangle\mathtt{credits}(S, R)$, $\langle\mathtt{calendar}\rangle\mathtt{slot}(S, T)$, $\langle\mathtt{registrar}\rangle\mathtt{registered}(S, C, R, T)$ or $\langle\mathtt{registrar}\rangle\mathtt{seats}(C, N)$.*
3. *For each student S, Δ has exactly one assumption of the form $\langle\mathtt{registrar}\rangle\mathtt{credits}(S, R)$.*
4. *For each student S and each time slot T, there is at most one assumption of* one *of the following forms in Δ: $\langle\mathtt{calendar}\rangle\mathtt{slot}(S, T)$ and $\langle\mathtt{registrar}\rangle\mathtt{registered}(S, C, R, T)$.*
5. *For each course C, there is exactly one assumption of the form $\langle\mathtt{registrar}\rangle\mathtt{course}(C, R, T)$ in Γ and one assumption of the form $\langle\mathtt{registrar}\rangle\mathtt{seats}(C, N)$ in Δ.*
6. *For each student S and each course C, there is at most one assumption of the form $\langle\mathtt{registrar}\rangle\mathtt{registered}(S, C, R, T)$ in Δ.*

Next, we define a state change, or step of the system. This notion is closely related to a similar idea from multi-set rewriting [10].

Definition 3 (Step) *We say that the pair of contexts $\Gamma; \Delta$ steps to the pair $\Gamma'; \Delta'$ (written $\Gamma; \Delta \longrightarrow \Gamma'; \Delta'$), if there is a derivation of $\Gamma; \Delta \Longrightarrow \gamma$ from $\Gamma; \Delta' \Longrightarrow \gamma$ that is* parametric *in the conclusion γ, i.e., there is a derivation of the following form that is correct for every γ.*

$$\begin{array}{c} \Gamma'; \Delta' \Longrightarrow \gamma \\ \vdots \\ \Gamma; \Delta \Longrightarrow \gamma \end{array}$$

By definition, $\longrightarrow$ is a transitive relation and $\longrightarrow^* = \longrightarrow$. The following lemma characterizes $\longrightarrow$ in terms of smaller inferences.

Lemma 1 (Characterization of steps). *Let $\Gamma; \Delta$ be a state, i.e., it satisfies all conditions in definition 2. Suppose $\Gamma; \Delta \longrightarrow \Gamma'; \Delta'$. Then $\Gamma = \Gamma'$ and the corresponding derivation from definition 3 must have the following form (for some $n \geq 0$).*

$$\cfrac{\ldots \qquad \cfrac{\ldots \qquad \Gamma; \Delta' \Longrightarrow \gamma}{\Gamma; \Delta_n \Longrightarrow \gamma}\,(\texttt{reg}) \atop \vdots \atop \Gamma; \Delta_1 \Longrightarrow \gamma}{\Gamma; \Delta \Longrightarrow \gamma}\,(\texttt{reg})$$

Further, the pairs $\Gamma; \Delta_i$ and $\Gamma; \Delta'$ are states of the system, i.e., they satisfy the conditions in definition 2.

Proof. The proof of this lemma follows by observing that if we reason backwards from the sequent $\Gamma; \Delta \Longrightarrow \gamma$, then the only rule that applies is `reg`. This is because γ is parametric (so no right rule applies). Further, we assumed that $\Gamma; \Delta$ is a state, and hence the only assumptions are of the form $\langle K \rangle A$, and so no left rule applies either because the form of γ is unknown. This argument can now be repeated. □

Lemma 1 provides us an induction principle for reasoning with steps. We can induct on the number of (`reg`) rules in the derivation mentioned in the lemma. Using this method we can prove the following correctness proposition.

Property 1 (Correctness of the policy) *Suppose $\Gamma; \Delta$ is a state and $\Gamma; \Delta \longrightarrow \Gamma; \Delta'$. Then the following hold.*

1. *For each student S, the sum of all credits R in authorizations of the form $\langle$*`registrar`*$\rangle$*`credits`*(S, R) and $\langle$*`registrar`*$\rangle$*`registered`*(S, C, R, T) in the context Δ is the same the corresponding sum in Δ'.*
2. *For every time slot T, and every student S, there is at most one authorization of the form $\langle$*`registrar`*$\rangle$*`registered`*(S, C, R, T) in Δ and Δ'.*
3. *For each course C, the sum of N in the (unique) certificate of the form $\langle$*`registrar`*$\rangle$*`seats`*(C, N) and the number of certificates of the form $\langle$*`registrar`*$\rangle$*`registered`*(S, C, R, T) in the context Δ is the same as the corresponding sum in Δ'.*

Proof. (1) and (3) follow by induction on the number of (reg) rules in the derivation corresponding to $\Gamma; \Delta \longrightarrow \Gamma; \Delta'$ from lemma 1. (2) follows from the fact that $\Gamma; \Delta'$ must be a state (lemma 1) and that each state satisfies clause 4 of definition 2. $\square$

Observe that if we start from a state $\Gamma; \Delta$ which has no assumptions of the form $\langle \texttt{registrar} \rangle \texttt{registered}(S, C, R, T)$, then the three statements of the above proposition imply the three correctness conditions mentioned at the beginning of the example for the state $\Gamma; \Delta'$. This formally proves that the policy implemented by the rule reg is correct with respect to those conditions.

5.2 Monetary Instruments

We describe a monetary system involving bank accounts, checks, promissory notes and tradeable items in our logic. In addition to linear authorizations, this example uses linear possessions to represent various monetary resources in the hands of principals. We use an approach similar to the previous example. First we describe the system in our logic using specific predicates and policy rules. Then we convert the policy rules to inference rules in order to force atomicity. Next, we define a notion of state and step for the system. Finally, we characterize steps in a manner analogous to lemma 1, and use this to prove two properties of the monetary system. The first property says that the total amount of money in the system remains unchanged as the system evolves. The second property says that the net assets of every principal remain constant.

We assume the existence of at least two principals, the bank and a credit company cc. We assume that every principal has an account in the bank. We represent account balances of principals as assumptions of linear possession: the assumption $\texttt{bank}\ \textsf{has}\ \texttt{balance}(K, N)$ represents the fact that K has N dollars in her bank account. In particular, the bank maintains its own account. This is represented as $\texttt{bank}\ \textsf{has}\ \texttt{balance}(\texttt{bank}, N)$.

A check for amount N is represented by the proposition $\texttt{check}(N)$.[3] A check is useful only when signed by a principal, who promises to pay the corresponding amount to its bearer, and possessed by some other principal who can use it. A check for N dollars signed by K, and possessed by K' is represented as $K'\ \textsf{has}\ \langle K \rangle \texttt{check}(N)$. Observe the difference in the use of affirmation and knowledge here: K's signature on the check is represented using an affirmation, which corresponds realistically to the fact that the check is an intent of payment made by K, whereas the fact that K' holds the check is represented using linear possession.

A promissory note of amount N signed by principal K and possessed by principal K' is represented by the assumption $K'\ \textsf{has}\ \langle K \rangle \texttt{iou}(N)$. The difference between a promissory note and check is that a check can be cashed at a bank whereas a promissory note cannot be. We assume that the credit company holds

[3] In order to keep the representation simple, we do not write the beneficiary of the check as an explicit argument in the predicate check.

$$\begin{array}{l}
\texttt{ax1} : [L]\langle K\rangle\texttt{check}(N) \multimap [\texttt{bank}]\texttt{balance}(K, N_1) \multimap \\
\quad [\texttt{bank}]\texttt{balance}(L, N_2) \multimap (N_1 \geq N) \supset \\
\quad\quad ([\texttt{bank}]\texttt{balance}(K, N_1 - N) \otimes [\texttt{bank}]\texttt{balance}(L, N_2 + N))
\end{array}$$

$$\begin{array}{l}
\texttt{ax2} : [\texttt{bank}]\texttt{balance}(K, N_1) \multimap [\texttt{bank}]\texttt{balance}(\texttt{bank}, N_2) \multimap (N_1 \geq N) \supset \\
\quad\quad ([\texttt{bank}]\texttt{balance}(K, N_1 - N) \otimes \\
\quad\quad\; [K]\langle\texttt{bank}\rangle\texttt{check}(N) \otimes \\
\quad\quad\; [\texttt{bank}]\texttt{balance}(\texttt{bank}, N_2 + N))
\end{array}$$

$$\texttt{ax3} : [\texttt{cc}]\langle K\rangle\texttt{iou}(N) \multimap ([\texttt{cc}]\langle K\rangle\texttt{iou}(N + N') \;\otimes\; [K]\langle\texttt{cc}\rangle\texttt{check}(N'))$$

$$\texttt{ax4} : [L]\langle M\rangle\texttt{check}(N) \multimap [K]\texttt{item}(N) \multimap ([L]\texttt{item}(N) \;\otimes\; [K]\langle M\rangle\texttt{check}(N))$$

$$\begin{array}{l}
\texttt{ax5} : [K]\langle L\rangle\texttt{check}(N) \multimap [\texttt{cc}]\langle K\rangle\texttt{iou}(N') \multimap (N' \geq N) \supset \\
\quad\quad ([\texttt{cc}]\langle K\rangle\texttt{iou}(N' - N) \;\otimes\; [\texttt{cc}]\langle L\rangle\texttt{check}(N))
\end{array}$$

$$\texttt{ax6} : [K]\langle K\rangle\texttt{check}(N)$$

Fig. 1. Rules for the monetary system in section 5.2

one promissory note of the form cc has $\langle K\rangle\texttt{iou}(N)$, for each principal K. The amount N may be zero if K does not owe the credit company anything. Finally, we have tradeable items in the system. An item of value N possessed by K is represented as K has $\texttt{item}(N)$.

Various possible transactions in the system are represented by the rules ax1-ax6 shown in figure 1. These rules are universally quantified over terms in uppercase. (ax1) says that if a principal L has a check for amount N signed by K, she may take it to the bank and get it cashed. In the process, the bank increments L's balance by N and decrements K's balance by the same amount. (ax2) permits a principal K having account balance at least N to obtain a banker's check for that amount. The bank transfers the amount N from K's account to its own and gives K a signed check for the same amount. (ax3) says that the credit company is willing to sign checks for principals by taking promissory notes from them.

If a principal K possesses an item worth N, she can sell it to L, provided L can produce a check for the same amount. This is represented by (ax4). The check moves from L to K during the transaction. (ax5) permits a principal K to pay the credit company using a check. (ax6) says that any principal K may sign a check for any amount N.

As with our last example, we convert each of these policy rules to an inference rule, which we add to our logic. All six inference rules for this example are shown in Appendix B. We proceed to define the notion of state and step for our system. In this example there are no unrestricted resources; therefore we omit the context Γ from our definitions.

Definition 4 (State) *A state of the system is a context Δ satisfying the following conditions.*

1. *Δ contains assumptions of the following forms only: (a)* $\texttt{bank}\ \mathsf{has}\ \texttt{balance}(K, N)$*, (b)* $\texttt{cc}\ \mathsf{has}\ \langle K\rangle\texttt{iou}(N)$*, (c)* $K\ \mathsf{has}\ \langle L\rangle\texttt{check}(N)$*, and (d)* $K\ \mathsf{has}\ \texttt{item}(N)$*.*
2. *For each principal K, there is exactly one assumption of each of the following forms in Δ:* $\texttt{bank}\ \mathsf{has}\ \texttt{balance}(K, N)$*, and* $\texttt{cc}\ \mathsf{has}\ \langle K\rangle\texttt{iou}(N)$*.*

The definition of a transition step is similar to that in the previous example.

Definition 5 (Step) *We say that the context Δ steps to Δ' (written $\Delta \longrightarrow \Delta'$) if there is a derivation of $\cdot;\Delta \Longrightarrow \gamma$ from the assumption $\cdot;\Delta' \Longrightarrow \gamma$ that is parametric in the conclusion γ.*

Now we state a characterization lemma for steps, similar to lemma 1.

Lemma 2 (Characterization of steps). *Suppose Δ is a state, i.e., it satisfies the conditions in definition 4. Let $\Delta \longrightarrow \Delta'$. Then the corresponding derivation from definition 5 has the following form, where each rule marked $(*)$ is one of the inference rules (*$\texttt{ax1}$*)-(*$\texttt{ax6}$*) (see appendix B).*

$$\dfrac{\dots \qquad \cdot;\Delta' \Longrightarrow \gamma}{\cdot;\Delta_n \Longrightarrow \gamma}\,(*)$$
$$\vdots$$
$$\dfrac{\dots \qquad\qquad \cdot;\Delta_1 \Longrightarrow \gamma}{\cdot;\Delta \Longrightarrow \gamma}\,(*)$$

Further, Δ_i and Δ' are states of the system, i.e., they satisfy the conditions in definition 4.

As before, this lemma gives us an induction principle for steps. Using that we can prove the property shown below. The first statement in the property says that the total amount of money in the system (as measured by the sum of the bank balances of all principals) remains constant. The second statement says that the net assets of every principal remain the same as the system evolves.

Property 2 (Consistency) *Let Δ be a state, and $\Delta \longrightarrow \Delta'$. Then the following hold.*

1. *The sum of all N such that* $\texttt{bank}\ \mathsf{has}\ \texttt{balance}(K, N)$ *exists in Δ, is the same as the corresponding sum for Δ'.*
2. *For each principal K, the sum of amounts N in all assumptions of the form* $\texttt{bank}\ \mathsf{has}\ \texttt{balance}(K, N)$*,* $K\ \mathsf{has}\ \langle L\rangle\texttt{check}(N)$ *and* $K\ \mathsf{has}\ \texttt{item}(N)$ minus *the sum of amounts N in assumptions of the form* $L\ \mathsf{has}\ \langle K\rangle\texttt{check}(N)$ *and* $\texttt{cc}\ \mathsf{has}\ \langle K\rangle\texttt{iou}(N)$ *is the same for both Δ and Δ'.*

Proof. By induction on the number of rules marked $(*)$ in the derivation corresponding to $\Delta \longrightarrow \Delta'$ from lemma 2. □

Generic description of atomicity and steps. In reasoning about the policies expressed in the two examples above, we used a number of similar concepts. In each case we had: (a) inference rules derived from policy rules to force atomicity, (b) notion of state, (c) notion of step, and (d) correctness conditions that are invariant across steps (properties 1 and 2). Of these, the definition of state and correctness are specific to a given policy and hard to generalize, but we can describe atomicity and step for all policies expressible in the logic in a generic manner by building a logic programming language based on our logic. We discuss this briefly.

Our enforcement of atomicity using inference rules is based on focusing. The notion of step relates quite naturally to the idea of forward chaining from proof search. Focusing and forward chaining can be combined systematically to build a logic programming language based on our logic. In such a language, proof search would proceed through interleaving phases of goal directed backward search and forward chaining. For policies expressed in the language, the notions of atomicity and step arise from the semantics of proof search. Technically, such a language requires a new lax modality to separate the two phases of proof search. Prior experience with the language LolliMon [17] suggests that this is feasible and can be implemented in practice.

6 Related Work

Our logic combines three major concepts: affirmation, knowledge, and linearity. As far as we are aware, this is the first time that a linear logic of affirmation and knowledge has been proposed and investigated. From the security perspective, affirmations are used for authorization, knowledge to specify the intended flow of information, linear affirmations for use-once credentials, and linear knowledge for possession of consumable resources. In prior work, two of the authors have developed the (non-linear) logic of affirmations for authorization [14], which is a small fragment of what is presented here. The use of linearity for single-use credentials together with an enforcement mechanism was first proposed by four of the authors [8], but the underlying logic was not fully developed and its properties not investigated. Furthermore, this logic was lacking a treatment of possession and knowledge.

The study of authorization by logical means was initiated by Abadi et al. [4]. Their logic was classical, presented in an axiomatic style and studied through a Kripke semantics. No proof-theory or meta-theoretic properties like cut elimination were described. Subsequently, a number of authorization logics have been studied and implemented [13, 9, 16, 15, 12, 19]. None of these logics is linear or provides proof-theoretic explanation of the logical operators. Further pointers on this line of work can be found in a survey by Abadi [2], who has also recently reformulated a (non-linear) constructive authorization logic [1] based on DCC [3] which is quite similar to a fragment of the logic given here.

The concepts of possession and knowledge are related to the K-operator from epistemic logics (see [20] for a survey of epistemic logics), which describes knowledge held by individuals and is similar to our operator $[\![K]\!]A$. As far as we are aware, the study of epistemic logics and its operators has been restricted almost exclusively to the classical setting. Simultaneous use of knowledge and linearity described in this paper to represent resources held by principals also appears to be new.

Proof-carrying authorization (PCA) [6, 7] is an architecture that can be used to specify and enforce security policies. We believe that the PCA architecture can be extended to include linearity in the style proposed here. The main new problem is enforcement of single use and atomicity of operations. Prior work in this direction indicates that contract signing protocols can be used for doing this effectively [8]. We do not believe it possible to implement the entire logic in a proof-carrying architecture because a proof of authorization may depend on private knowledge held by individuals, and verifying such a proof could result in a breach of security. Therefore, in order to effectively implement this logic using PCA, we have to restrict ourselves to certain fragments, for example, where authorizations made by any principal do not depend on knowledge held by others.

7 Conclusion

We have presented a new constructive linear logic that develops the logical concepts of affirmation and knowledge from judgmental principles. The logic yields a clean proof theory and an analysis of meanings of the connectives from proof rules. We have shown that the logic satisfies cut elimination and demonstrated that policies expressed in the logic are amenable to meta-theoretic analysis regarding their correctness. We believe that this logic is a good foundation for expressing policies involving linear authorizations and resources held by principals.

There are several avenues for future work besides those mentioned with the related work. One is to construct and implement a logic programming language based on this logic, as mentioned at the end of section 5. Another avenue for future work is to study non-interference properties for the logic, along the lines of [14]. These properties permit administrators to explore the consequences of their policies by showing that certain forms of assumptions cannot affect provability of certain other forms of conclusions. A third direction is to extend the logic with explicit constructs for distribution of knowledge on physical sites to reason about networked security systems at a slightly lower level of abstraction.

References

1. Martín Abadi. Personal communication.
2. Martín Abadi. Logic in access control. In *Proceedings of the 18th Annual Symposium on Logic in Computer Science (LICS'03)*, pages 228–233, Ottawa, Canada, June 2003. IEEE Computer Society Press.

3. Martín Abadi, Anindya Banerjee, Nevin Heintze, and Jon G. Riecke. A core calculus of dependency. In *Conference Record of the 26th Sympoisum on Principles Of Programming Languages (POPL'99)*, pages 147–160, San Antonio, Texas, January 1999. ACM Press.
4. Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, October 1993.
5. Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.
6. Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In G. Tsudik, editor, *Proceedings of the 6th Conference on Computer and Communications Security*, pages 52–62, Singapore, November 1999. ACM Press.
7. Lujo Bauer. *Access Control for the Web via Proof-Carrying Authorization*. PhD thesis, Princeton University, November 2003.
8. Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter. Consumable credentials in logic-based access control. Technical Report CMU-CYLAB-06-002, Carnegie Mellon University, February 2006.
9. Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur.*, 6(1):71–127, 2003.
10. Stefano Bistarelli, Iliano Cervesato, Gabriele Lenzini, and Fabio Martinelli. Relating Multiset Rewriting and Process Algebras for Security Protocol Analysis. *Journal of Computer Security*, 13:3–47, February 2005.
11. Bor-Yuh Evan Chang, Kaustuv Chaudhuri, and Frank Pfenning. A judgmental analysis of linear logic. Submitted. Extended version available as Technical Report CMU-CS-03-131R, December 2003.
12. Jason Crampton, George Loizou, and Greg O' Shea. A logic of access control. *The Computer Journal*, 44(1):137–149, 2001.
13. John DeTreville. Binder, a logic-based security language. In M.Abadi and S.Bellovin, editors, *Proceedings of the 2002 Symposium on Security and Privacy (S&P'02)*, pages 105–113, Berkeley, California, May 2002. IEEE Computer Society Press.
14. Deepak Garg and Frank Pfenning. Non-interference in constructive authorization logic. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW 19)*. IEEE Computer Society Press, 2006. To appear.
15. Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.
16. Ninghui Li and John C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *PADL '03: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 58–73. Springer-Verlag, 2003.
17. Pablo López, Frank Pfenning, Jeff Polakow, and Kevin Watkins. Monadic concurrent linear logic programming. In *Proceedings of the 7th International Symposium on Principles and Practice of Declarative Programming (PPDP'05)*, Lisbon, Portugal, 2005.
18. Frank Pfenning. Structural cut elimination I. Intuitionistic and classical logic. *Information and Computation*, 157(1/2):84–141, March 2000.
19. Harald Rueß and Natarajan Shankar. Introducing Cyberlogic. In *Proceedings of the 3rd Annual High Confidence Software and Systems Conference*, Baltimore, Maryland, April 2003.

20. W. van der Hoek and R. Verbrugge. Epistemic logic: A survey. *Game Theory and Applications*, 8:53–94, 2002.

A Summary of the Sequent Calculus

This appendix summarizes the sequent calculus for our logic.

A.1 Affirmation, Possession and Knowledge

The following are the sequent calculus rules relating to affirmation, possession and knowledge.

$$\frac{}{\Gamma; A \Longrightarrow A}(\texttt{init}) \qquad \frac{\Gamma, A; \Delta, A \Longrightarrow \gamma}{\Gamma, A; \Delta \Longrightarrow \gamma}(\texttt{copy})$$

$$\frac{\Gamma; \Delta, A \Longrightarrow \gamma}{\Gamma; \Delta, K \textsf{ has } A \Longrightarrow \gamma}(\textsf{has}) \qquad \frac{\Gamma, K \textsf{ knows } A; \Delta, A \Longrightarrow \gamma}{\Gamma, K \textsf{ knows } A; \Delta \Longrightarrow \gamma}(\textsf{knows})$$

$$\frac{\Gamma|_K; \Delta|_K \Longrightarrow A}{\Gamma; \Delta|_K \Longrightarrow [K]A}([]R) \qquad \frac{\Gamma; \Delta, K \textsf{ has } A \Longrightarrow \gamma}{\Gamma; \Delta, [K]A \Longrightarrow \gamma}([]L)$$

$$\frac{\Gamma|_K; \cdot \Longrightarrow A}{\Gamma; \cdot \Longrightarrow [\![K]\!]A}([\![]\!]R) \qquad \frac{\Gamma, K \textsf{ knows } A; \Delta \Longrightarrow \gamma}{\Gamma; \Delta, [\![K]\!]A \Longrightarrow \gamma}([\![]\!]L)$$

$$\frac{\Gamma; \Delta \Longrightarrow A}{\Gamma; \Delta \Longrightarrow K \textsf{ affirms } A}(\textsf{affirms})$$

$$\frac{\Gamma; \Delta \Longrightarrow K \textsf{ affirms } A}{\Gamma; \Delta \Longrightarrow \langle K \rangle A}(\langle\rangle R) \qquad \frac{\Gamma; \Delta, A \Longrightarrow K \textsf{ affirms } C}{\Gamma; \Delta, \langle K \rangle A \Longrightarrow K \textsf{ affirms } C}(\langle\rangle L)$$

A.2 Standard Connectives of Linear Logic

The sequent calculus rules for the following standard connectives of linear logic [11]: $A \otimes B$, $A \multimap B$, $A \supset B$, $!A$ and $\forall x.A$ are shown below. In the $\forall R^a$ rule, the parameter a must be fresh, i.e., it must not occur in Γ, Δ or $\forall x.A$.

$$\frac{\Gamma;\Delta_1 \Longrightarrow A \qquad \Gamma;\Delta_2 \Longrightarrow B}{\Gamma;\Delta_1\Delta_2 \Longrightarrow A\otimes B}(\otimes R) \qquad \frac{\Gamma;\Delta,A,B \Longrightarrow \gamma}{\Gamma;\Delta,A\otimes B \Longrightarrow \gamma}(\otimes L)$$

$$\frac{\Gamma;\Delta,A \Longrightarrow B}{\Gamma;\Delta \Longrightarrow A\multimap B}(\multimap R) \qquad \frac{\Gamma;\Delta_1 \Longrightarrow A \qquad \Gamma;\Delta_2,B \Longrightarrow \gamma}{\Gamma;\Delta_1\Delta_2,A\multimap B \Longrightarrow \gamma}(\multimap L)$$

$$\frac{\Gamma,A;\Delta \Longrightarrow B}{\Gamma;\Delta \Longrightarrow A\supset B}(\supset R) \qquad \frac{\Gamma;\cdot \Longrightarrow A \qquad \Gamma;\Delta,B \Longrightarrow \gamma}{\Gamma;\Delta,A\supset B \Longrightarrow \gamma}(\supset L)$$

$$\frac{\Gamma;\cdot \Longrightarrow A}{\Gamma;\cdot \Longrightarrow !A}(!R) \qquad \frac{\Gamma,A;\Delta \Longrightarrow \gamma}{\Gamma;\Delta,!A \Longrightarrow \gamma}(!L)$$

$$\frac{\Gamma;\Delta \Longrightarrow A[a/x]}{\Gamma;\Delta \Longrightarrow \forall x.A}(\forall R^a) \qquad \frac{\Gamma;\Delta,A[t/x] \Longrightarrow \gamma}{\Gamma;\Delta,\forall x.A \Longrightarrow \gamma}(\forall L)$$

B Inference rules for the example in section 5.2

$$\frac{\begin{array}{c}\Gamma;\Delta_1 \Longrightarrow [L]\langle K\rangle\texttt{check}(N)\\ \Gamma;\Delta_2 \Longrightarrow [\mathsf{bank}]\texttt{balance}(K,N_1)\\ \Gamma;\Delta_3 \Longrightarrow [\mathsf{bank}]\texttt{balance}(L,N_2)\\ \Gamma;\cdot \Longrightarrow (N_1 \geq N)\\ \Gamma;\Delta_4,\mathsf{bank\ has\ }\texttt{balance}(K,N_1-N),\mathsf{bank\ has\ }\texttt{balance}(L,N_2+N) \Longrightarrow \gamma\end{array}}{\Gamma;\Delta_1\Delta_2\Delta_3\Delta_4 \Longrightarrow \gamma}\ (\texttt{ax1})$$

$$\frac{\begin{array}{c}\Gamma;\Delta_1 \Longrightarrow [\mathsf{bank}]\texttt{balance}(K,N_1)\\ \Gamma;\Delta_2 \Longrightarrow [\mathsf{bank}]\texttt{balance}(\mathsf{bank},N_2)\\ \Gamma;\cdot \Longrightarrow (N_1 \geq N)\\ \begin{array}{l}\Gamma;\Delta_3,\mathsf{bank\ has\ }\texttt{balance}(K,N_1-N),K\mathsf{\ has\ }\langle\mathsf{bank}\rangle\texttt{check}(N),\\ \qquad\mathsf{bank\ has\ }\texttt{balance}(\mathsf{bank},N_2+N)\end{array} \Longrightarrow \gamma\end{array}}{\Gamma;\Delta_1\Delta_2\Delta_3 \Longrightarrow \gamma}\ (\texttt{ax2})$$

$$\frac{\begin{array}{c}\Gamma;\Delta_1 \Longrightarrow [\mathsf{cc}]\langle K\rangle\texttt{iou}(N)\\ \Gamma;\Delta_2,\mathsf{cc\ has\ }\langle K\rangle\texttt{iou}(N+N'),K\mathsf{\ has\ }\langle\mathsf{cc}\rangle\texttt{check}(N') \Longrightarrow \gamma\end{array}}{\Gamma;\Delta_1\Delta_2 \Longrightarrow \gamma}\ (\texttt{ax3})$$

$$\frac{\begin{array}{c}\Gamma;\Delta_1 \Longrightarrow [L]\langle M\rangle\texttt{check}(N)\\ \Gamma;\Delta_2 \Longrightarrow [K]\texttt{item}(N)\\ \Gamma;\Delta_3,L\mathsf{\ has\ }\texttt{item}(N),K\mathsf{\ has\ }\langle M\rangle\texttt{check}(N) \Longrightarrow \gamma\end{array}}{\Gamma;\Delta_1\Delta_2\Delta_3 \Longrightarrow \gamma}\ (\texttt{ax4})$$

$$\frac{\begin{array}{c}\mathit{\Gamma}; \mathit{\Delta}_1 \Longrightarrow [K]\langle L\rangle\mathtt{check}(N) \\ \mathit{\Gamma}; \mathit{\Delta}_2 \Longrightarrow [\mathtt{cc}]\langle K\rangle\mathtt{iou}(N') \\ \mathit{\Gamma}; \cdot \Longrightarrow N' \geq N \\ \mathit{\Gamma}; \mathit{\Delta}_3, \mathtt{cc}\ \mathsf{has}\ \langle K\rangle\mathtt{iou}(N'-N), \mathtt{cc}\ \mathsf{has}\ \langle L\rangle\mathtt{check}(N) \Longrightarrow \gamma\end{array}}{\mathit{\Gamma}; \mathit{\Delta}_1\mathit{\Delta}_2\mathit{\Delta}_3 \Longrightarrow \gamma}\ (\mathtt{ax5})$$

$$\frac{\mathit{\Gamma}; \mathit{\Delta}, K\ \mathsf{has}\ \langle K\rangle\mathtt{check}(N) \Longrightarrow \gamma}{\mathit{\Gamma}; \mathit{\Delta} \Longrightarrow \gamma}\ (\mathtt{ax6})$$