A Framework for Linear Authorization Logics

Vivek Nigam[1]

Abstract

Linear authorization logics (LAL) are logics based on linear logic that can be used for modeling effect-based authentication policies. LAL has been used in the context of the Proof-Carrying Authorization framework, where formal proofs must be constructed in order for a principal to gain access to some resource elsewhere. This paper investigates the complexity of the provability problem, that is, determining whether a linear authorization logic formula is provable or not. We show that the multiplicative propositional fragment of LAL is already undecidable in the presence of two principals. On the other hand, we also identify a first-order fragment of LAL for which provability is PSPACE-complete. Finally, we argue by example that the latter fragment is natural and can be used in practice.

1. Introduction

There are many situations where using and issuing authorizations may have effects. For example, a professor that is away might want to provide an authorization to one of his students to enter his office *at most once* in order to pick a book. Once this student has consumed this authorization by entering the office, the student can no longer enter it unless he obtains another authorization.

Such a scenario has been implemented [5] following the Proof-Carrying Authorization framework (PCA) [4], where access control policies are specified as logical theories and whenever a principal (or agent) requests permission to access some resource, she provides a formal proof demonstrating that such an access follows from the policies. While the use of logic to specify access control policies dates back to some decades ago [1], the main difference between PCA and previous approaches is the existence of *proof objects*. The use of proof objects reduces the required trust base of the principals in a system, as a principal just needs to *check* whether the attached proof object is correct.

Access control logics for distributed systems are called *authorization logics* [2]. Traditionally classical logics have been used to specify policies. However, in order to specify *effect-based* policies, such as the one illustrated above, one moves to linear logic [16]. As linear logic formulas can be interpreted as resources, linear logic theories can model state-based systems and therefore are suitable for specifying policies that

Email address: vivek.nigam@gmail.com (Vivek Nigam)
[1]Federal University of Paraíba, João Pessoa, Brazil

involve consumable credentials, such as money or the right to access a room at most once. *Linear authorization logics* (LAL) [14] are authorization logics based on linear logic extended with modality operators [2], *e.g.*, $says$ or has.

A central requirement in PCA is the *construction* of proof objects from policies specified using (linear) authorization logics. Although it is easy to check whether a proof object is correct, finding a correct proof object involves proof search which may be hard. In PCA, it is the burden of the requesting principal, which is normally assumed to be more powerful, to construct such objects from the policies available. It is therefore important to determine how hard is the task of constructing proofs, that is, to determine the complexity of the *provability problem* for LAL.

The contribution of this paper is twofold: (1) we propose a logical framework for LAL and (2) we investigate the complexity of the provability problem for different fragments of LAL.

For our first contribution, we propose using the sequent calculus proof system SELL, introduced in [28], as a logical framework where one can specify different linear authorization logics. First, we show how to encode existing authorization logics [14]. Then we show how SELL allows one to specify a wider range of policies that did not seem possible before. For instance, we modularly increase the expressiveness of our encoding by showing that one can also express in SELL policies of the form: "A principal may use a lower-ranked set of policy rules, but not a higher-ranked set of policy rules."

Our second main contribution is of investigating the complexity of the provability problem for LAL. We show that the provability problem is *undecidable* already for the propositional multiplicative fragment with no function symbols and only two principals that have only consumable credentials. The proof follows by encoding a two-counter Minsky machine [25], which is known to be Turing complete. This means that constructing proof objects for simple policies may already not be computable. Interestingly, the upper bound for the provability problem for the same fragment (MELL) of linear logic [16] is not known. As exponentials can be seen as modalities, this result means that adding an extra modality to MELL possibly leads to undecidability. This is in accordance with previous results on the complexity of SELL [7].

Our second complexity result in more interesting from both the application and technical point of views. In particular, we propose a *first-order* fragment of LAL for which the provability problem is PSPACE-complete with respect to the size of the given formula. In particular, we restrict policies to be only *balanced bipoles* with no function symbols and where principals have only consumable credentials, *i.e.*, principals have credentials that can be used exactly once.

Bipoles is a class of logical formulas that often appear in proof theory literature [23]. From a proof search perspective, one can make precise connections (sound and complete correspondence) between the reachability problem of multiset rewriting systems (MSR) and the provability problem of linear logic bipoles [6, 28]. However, the same correspondence does not work as smoothly when using LAL due to the presence of modalities, *e.g.*, $says$. But as we show in this paper, it works when using the expressiveness gained by using SELL. In particular, we use the ability to specify in SELL when formulas should be proved *without* using any policy rules. That is, such a formula should be necessarily derived using only the set of already derived formulas.

This condition can be intuitively interpreted as checking whether a formula follows from the state of the system (or table of a principal).

On the other hand, a sequence of papers [21, 19, 18, 17] have investigated the complexity of the reachability problem for systems whose actions are *balanced*. An action is classified as balanced if its pre and post-conditions have the *same number* of atomic formulas. It has been shown that the reachability problem for MSR with balanced actions is PSPACE-complete. Given the correspondence between the reachability and provability problem of bipoles formulas, we show that the provability problem for balanced bipoles is also PSPACE-complete.

This paper is structured as follows:

- Section 2 reviews the proof system SELL, showing how one can encode existing linear authorization logics and how to modularly extend such encoding in order to express a wider range of policies. Finally, we also review the focused proof system for SELL, which is the machinery used to formally prove the correspondence between logic provability and MSR reachability.
- Section 3 contains the undecidability proof for the propositional multiplicative fragment of the linear authorization logic proposed in [14].
- Section 4 describes the connections between bipoles and MSR, formalizing a novel correspondence between MSR reachability and logic provability of a first-order fragment of linear authorization logics, namely, when policies are bipoles.
- Section 5 contains the PSPACE-completeness proof for the provability problem when policies are balanced bipoles.
- Section 6 contains a student registration example based on a similar example from [14], but that is specified using balanced bipoles.

Finally, in Section 7 we conclude and comment on related work.

This is an expanded and improved version of the conference paper [27]. In particular, the encoding in [27] of Minsky machines used additive units ($\top$), thus not being purely multiplicative. Here, we modify that encoding and show that the purely multiplicative fragment of LAL (without $\top$) is undecidable.

2. A Framework for Linear Authorization Logics

We propose using linear logic with subexponentials (SELL) as a framework for specifying LAL. The system for classical linear logic with subexponentials was proposed in [8] and further investigated in [28]. However, as argued in [15], the use of intuitionistic logic seems more adequate to PCA applications as it allows only constructive proofs. We now review the proof system for intuitionistic linear logic with subexponentials.

Besides sharing all connectives with linear logic, SELL may include as many exponential-like connectives, called *subexponentials*, as one needs. Subexponentials, written $!^l$ and $?^l$, are labeled with an index, l. The subexponentials indexes available in a system are formally specified by the tuple $\langle I, \preceq, \mathcal{U}\rangle$, where I is the set of labels for subexponentials, $\preceq$ is a preorder relation among the elements of I, and $\mathcal{U} \subseteq I$ specifies which subexponentials allow weakening and contraction. The pre-order $\preceq$, on the other hand, specifies the provability relation among subexponentials and is upwardly closed with respect to the set $\mathcal{U}$, *i.e.*, if $x \preceq y$ and $x \in \mathcal{U}$, then $y \in \mathcal{U}$.

$$\frac{}{A \longrightarrow A}\ I \qquad \frac{\Gamma_1 \longrightarrow F \quad \Gamma_2, F \longrightarrow G}{\Gamma_1, \Gamma_2 \longrightarrow G}\ \text{Cut}$$

$$\frac{\Gamma, F, H \longrightarrow G}{\Gamma, F \otimes H \longrightarrow G}\ \otimes_l \quad \frac{\Gamma_1 \longrightarrow F \quad \Gamma_2 \longrightarrow H}{\Gamma_1, \Gamma_2 \longrightarrow F \otimes H}\ \otimes_r \quad \frac{\Gamma_1 \longrightarrow F \quad \Gamma_2, H \longrightarrow G}{\Gamma_1, \Gamma_2, F \multimap H \longrightarrow G}\ \multimap_l \quad \frac{\Gamma, F \longrightarrow H}{\Gamma \longrightarrow F \multimap H}\ \multimap_r$$

$$\frac{\Gamma, F[e/x] \longrightarrow G}{\Gamma, \exists x.F \longrightarrow G}\ \exists_l \quad \frac{\Gamma \longrightarrow G[t/x]}{\Gamma \longrightarrow \exists x.G}\ \exists_r \quad \frac{\Gamma, F[t/x] \longrightarrow G}{\Gamma, \forall x.F \longrightarrow G}\ \forall_l \quad \frac{\Gamma \longrightarrow G[e/x]}{\Gamma \longrightarrow \forall x.G}\ \forall_r$$

Figure 1: Multiplicative, first-order fragment of intuitionistic linear logic. As usual in the $\exists_l$ and $\forall_l$, e is fresh, *i.e.*, it does not appear in Γ nor G.

Given a signature Σ, the proof system SELL_Σ is constructed as follows: The system contains all the introduction rules for $\&, \oplus, \otimes, \multimap, \exists, \forall$ and the units, $1, \top$ and 0 as well as the exchange rules exactly as in linear logic [16]. The rules for the first-order multiplicative fragment are depicted in Figure 1. For every index $a \in \mathcal{I}$, we add the rules:

$$\frac{\Gamma, F \longrightarrow G}{\Gamma, !^a F \longrightarrow G}\ !^a{}_L \qquad \frac{!^{x_1} F_1, \ldots !^{x_n} F_n \longrightarrow G}{!^{x_1} F_1, \ldots !^{x_n} F_n \longrightarrow !^a G}\ !^a{}_R$$

$$\frac{!^{x_1} F_1, \ldots !^{x_n} F_n, F \longrightarrow ?^{x_{n+1}} G}{!^{x_1} F_1, \ldots !^{x_n} F_n, ?^a F \longrightarrow ?^{x_{n+1}} G}\ ?^a{}_L \qquad \frac{\Gamma \longrightarrow G}{\Gamma \longrightarrow ?^a G}\ ?^a{}_R$$

where the rules $!^a{}_R$ and $?^a{}_L$ have the side condition that $a \preceq x_i$ for all i. That is, one can only introduce a $!^a$ on the right (or a $?^a$ on the left) if all other formulas in the sequent are marked with indexes that are greater or equal than a.

Finally, for all indexes $a \in \mathcal{U}$, we add the following structural rules:

$$\frac{\Gamma, !^a F, !^a F \longrightarrow G}{\Gamma, !^a F \longrightarrow G}\ C\ , \quad \frac{\Gamma \longrightarrow G}{\Gamma, !^a F \longrightarrow G}\ W \text{ and } \frac{\Gamma \longrightarrow \cdot}{\Gamma \longrightarrow ?^a G}\ W$$

That is, we are also free to specify which indexes are unrestricted, namely those appearing in the set $\mathcal{U}$, and which are linear or consumable, namely the remaining indexes.

Danos *et al.* showed in [8] that the classical version of SELL admits cut-elimination. It is also possible to show that the intuitionistic version shown above admits cut-elimination for any signature Σ.

Theorem 2.1. *For any signature Σ, the cut-rule is admissible in SELL_Σ.*

In the remainder of the paper, we elide the subscript Σ from SELL_Σ, whenever it is clear from the context.

2.1. Specifying Linear Authorization Logics

This section enters into the details of how one can encode LAL in SELL. Besides containing all the connectives of linear logic, except the exponentials, ! and ?, LAL contains three sorts of families of modalities, namely says, has, and knows, indexed by principal names [14], *e.g.*, $K\, says\, C$, $K\, has\, C$, and $K\, knows\, C$, where K is a principal name and C is a formula. The $says$ modality expresses the intent of a principal, while the has modality expresses that a principal possesses some consumable resource, which can only be used once, *e.g.*, money, and the $knows$ modality expresses the knowledge of a principal, which can be used as many times as needed, *i.e.*, it is an unrestricted resource that can be weakened and contracted.

Intuitively, one can conclude that a principal possesses some resource if one can derive it only from her possessions and from her knowledge base. On the other hand, one can conclude that a principal knows some knowledge if it can be derived only from her knowledge base. Formally, the introduction rules for possession and knowledge modalities are as follows [14]:

$$\frac{\Gamma, F \longrightarrow G}{\Gamma, K\, has\, F \longrightarrow G}\ has_L \qquad \frac{\Psi, \Delta \longrightarrow G}{\Psi, \Delta \longrightarrow K\, has\, G}\ has_R$$

$$\frac{\Gamma, F \longrightarrow G}{\Gamma, K\, knows\, F \longrightarrow G}\ knows_L \qquad \frac{\Psi \longrightarrow G}{\Psi \longrightarrow K\, knows\, G}\ knows_R$$

where Ψ contains only formulas of the form $K\, knows\, C$, while Δ contains only formulas of the form $K\, has\, C$. Moreover, $K\, knows\, F$ can be weakened and contracted on the left.

$$\frac{\Gamma, K\, knows\, F, K\, knows\, F \longrightarrow G}{\Gamma, K\, knows\, F \longrightarrow G}\ C \qquad \frac{\Gamma \longrightarrow G}{\Gamma, K\, knows\, F \longrightarrow G}\ W$$

On the other hand, $says$ are families of lax modalities [12], whose introduction rules are as follows:

$$\frac{\Gamma, F \longrightarrow K\, says\, G}{\Gamma, K\, says\, F \longrightarrow K\, says\, G}\ says_L \qquad \frac{\Gamma \longrightarrow G}{\Gamma \longrightarrow K\, says\, G}\ says_R$$

The left inference rule specifies that to prove $K\, says\, G$ one may use the affirmations of the principal K, while the right rule specifies that principals are rational and always affirm formulas that are provable.

Finally, it is assumed that all principals know a common set of global policies Θ. In [14], it was assumed that these rules are in the knowledge base of all principals, *i.e.*, the formula $K\, knows\, F$ for all formulas $F \in \Theta$ and principal names K appears to the left-hand-side of sequents. Notice that they can be used as many times needed as knowledge is unrestricted.

We start by encoding these modalities in SELL and later in Section 2.2 we propose extensions that allow one to express a wider range of policies.

Assume given a finite set of principal names $\mathcal{K}$. The set of subexponential indexes is given below:

$$I_{\mathcal{K}} = \{h_K, k_K, sL_K, sR_K \mid K \in \mathcal{K}\} \cup \{gl, lin\}.$$

Intuitively, h_K is used for specifying has modalities, k_K is used for specifying $knows$ modalities, sL_K and sR_K are used for specifying $says$ modalities, lin for linear formulas appearing on the left-hand-side of sequents, and gl for the policy rules shared among the principals. Moreover, only the k_K indexes and the index gl are unrestricted, that is, $k_K, gl \in \mathcal{U}$, for all $K \in \mathcal{K}$, while the remaining subexponentials are linear.

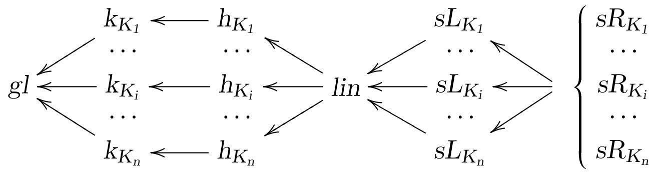

Figure 2: Graphical representation of the partial order $\preceq$ among subexponential indexes. Here if $a \longrightarrow b$ means that $a \preceq b$. For instance, $h_{K_j} \preceq k_{K_j}$ for all principal names K_j. The bracket denotes that index sR_K is less than sL_K for all principals $K \in \mathcal{K}$, *e.g.*, $sR_{K_1} \preceq sL_{K_n}$. The subexponential signature specifying this system is denoted by $\Sigma_{\mathcal{K}}$, where $\mathcal{K} = \{K_1, \ldots, K_n\}$.

Finally, these indexes are organized in the partial order $\preceq$ as depicted in Figure 2. The subexponential signature specifying this system is denoted by $\Sigma_{\mathcal{K}}$. We will normally use the Greek letter Θ to denote the set of formulas specifying the global policies that are known to all principals.

We encode $says$, has, and $knows$ modalities using the four types of subexponential indexes above and two encodings $[\![\cdot]\!]_L$ and $[\![\cdot]\!]_R$, for, respectively, negative and positive occurrences of formulas, (or to the left and right-hand-side of the sequent):

$$\begin{array}{lcllcl}
[\![K\, has\, C]\!]_L & = & !^{h_K}[\![C]\!]_L & [\![K\, has\, C]\!]_R & = & !^{h_K}[\![C]\!]_R \\
[\![K\, knows\, C]\!]_L & = & !^{k_K}[\![C]\!]_L & [\![K\, knows\, C]\!]_R & = & !^{k_K}[\![C]\!]_R \\
[\![K\, says\, C]\!]_L & = & !^{sL_K}?^{sR_K}[\![C]\!]_L & [\![K\, says\, C]\!]_R & = & ?^{sR_K}[\![C]\!]_R
\end{array}$$

Notice the asymmetry of the encoding of $says$ modalities. Its left encoding uses $!^{sL_K}?^{sR_K}$, while the right encoding uses $?^{sR_K}$. As we show below, these encodings capture the requirement for the introduction of a lax modality on the left. For the remaining formulas whose main connective is not a modality, the left-encoding adds an additional $!^{lin}$, while the right-encoding does not do that. For example the encoding of formulas whose main connective is a $\multimap$ is shown below:

$$\begin{array}{lcl}
[\![F \multimap G]\!]_L & = & !^{lin}([\![F]\!]_R \multimap [\![G]\!]_L) \\
[\![F \multimap G]\!]_R & = & [\![F]\!]_L \multimap [\![G]\!]_R
\end{array}$$

We show in detail some of the the introduction rules of $\mathrm{SELL}_{\Sigma_{\mathcal{K}}}$. In the derivations below, we write $!^a\{F_1, \ldots, F_n\}$ to represent the formulas $!^aF_1, \ldots, !^aF_n$.

Due to the condition on the right introduction of bangs, the right introduction rules for $!^{k_K}$ and $!^{h_K}$ have necessarily the following forms:

$$\frac{!^{gl}\{\Theta\}, !^{k_K}\{\Gamma\} \longrightarrow F}{!^{gl}\{\Theta\}, !^{k_K}\{\Gamma\} \longrightarrow !^{k_K}F} \qquad \frac{!^{gl}\{\Theta\}, !^{k_K}\{\Gamma\}, !^{h_K}\{\Delta\} \longrightarrow F}{!^{gl}\{\Theta\}, !^{k_K}\{\Gamma\}, !^{h_K}\{\Delta\} \longrightarrow !^{h_K}F}$$

As one can easily verify by using the encoding given above and by instatiating Θ as $\emptyset$, the rule to the left corresponds to the right introduction rule for $knows$ modalities, as it specifies that one can derive a $knows$ formula for a principal K on the right if this formula is derivable using only the knowledge of K. On the other hand, the rule to the right corresponds to the right introduction rule for has modalities, as it specifies that one can introduce a has formula for the principal K on the right if this formula is derivable only from K's possessions and K's knowledge.

Furthermore, the rules above also illustrate the possibility of distinguishing by using the subexponential gl the set of global policies from the private knowledge base of principals. Since they can be contracted and weakened they can be safely be used in LAL proofs. In [14] such global policies were specified by assuming that all principals know these global policies. Both approaches are equivalent as the knowledge of principals is also unrestricted. We use here, however, the former approach, as it explicitly distinguishes the collective global policies which are known to all principals from the private knowledge of principals.

In order to specify the lax restriction for $says$ modalities, we use the indexes sL_K and sR_K. Due to the restriction on the left introduction of question-marks, the left introduction rule for $?^{sR_K}$ has the following shape:

$$\frac{\Gamma, F \longrightarrow ?^{sR_K} G}{\Gamma, ?^{sR_K} F \longrightarrow ?^{sR_K} G}$$

where all formulas in Γ are marked with bangs whose indexes belong to the set

$$\{k_{K_i}, h_{K_i}, sL_{K_i} \mid K_i \in \mathcal{K}\} \cup \{lin, gl\}.$$

That is, one is only allowed to introduce a $?^{sR_K}$ on the left if the formula to the right hand side of the sequent is marked with $?^{sR_K}$. Furthermore, notice that Γ can contain affirmations of other principals and even formulas that are not part of the knowledge nor possession nor affirmation of any principal. This is the reason why in the encoding above we translate $says$ modalities on the left by adding $!^{sL_{K_i}} ?^{sR_{K_i}}$ and formulas whose main connective is not a modality with $!^{lin}$.

We can prove that the encoding above is sound and complete. One needs to take extra care with the $!^{lin}$ used in the encoding. However, since they appear only on the left-hand side of sequents, they do not cause any problems.

Theorem 2.2. *A sequent $\Gamma \longrightarrow F$ is provable in the proof system for linear authorization logic shown above if and only if $[\![\Gamma]\!]_L \longrightarrow [\![F]\!]_R$ is provable in SELL.*

Proof In our proof, we rely on the fact that all occurrences of ${!^{lin}}_L$ rules, that is, derelictions of $!^{lin}$, permute upwards with respect to all other rules. We show some of the cases below:

$$\dfrac{\dfrac{\Gamma_1, F \longrightarrow A \quad \Gamma_2 \longrightarrow B}{\Gamma_1, \Gamma_2, F \longrightarrow A \otimes B}\ \otimes}{\Gamma_1, \Gamma_2, !^{lin} F \longrightarrow A \otimes B}\ {!^{lin}}_L \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\Gamma_1, F \longrightarrow A}{\Gamma_1, !^{lin} F \longrightarrow A}\ {!^{lin}}_L \quad \Gamma_2 \longrightarrow B}{\Gamma_1, \Gamma_2, !^{lin} F \longrightarrow A \otimes B}\ \otimes_R$$

$$\dfrac{\dfrac{\Gamma, F \longrightarrow G_i}{\Gamma, F \longrightarrow G_1 \oplus G_2}\ \oplus_{r_i}}{\Gamma, !^{lin}F \longrightarrow G_1 \oplus G_2}\ !^{lin}{}_L \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma, F \longrightarrow G_i}{\Gamma_1, !^{lin}F \longrightarrow G_i}\ !^{lin}{}_L}{\Gamma, !^{lin}F \longrightarrow G_1 \oplus G_2}\ \oplus_{r_i}$$

$$\dfrac{\dfrac{\Gamma, F \longrightarrow G_1 \quad \Gamma, F \longrightarrow G_2}{\Gamma, F \longrightarrow G_1 \,\&\, G_2}\ \&_R}{\Gamma, !^{lin}F \longrightarrow G_1 \,\&\, G_2}\ !^{lin}{}_L \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma, F \longrightarrow G_1}{\Gamma, !^{lin}F \longrightarrow G_1}\ !^{lin}{}_L \quad \dfrac{\Gamma, F \longrightarrow G_2}{\Gamma, !^{lin}F \longrightarrow G_2}\ !^{lin}{}_L}{\Gamma, !^{lin}F \longrightarrow G_1 \,\&\, G_2}\ \&_R$$

$$\dfrac{\dfrac{\Gamma_1, F \longrightarrow A \quad \Gamma_2, B \longrightarrow G}{\Gamma_1, \Gamma_2, A \multimap B, F \longrightarrow G}\ \multimap_L}{\Gamma_1, \Gamma_2, A \multimap B, !^{lin}F \longrightarrow G}\ !^{lin}{}_L \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma_1, F \longrightarrow A}{\Gamma_1, !^{lin}F \longrightarrow A}\ !^{lin}{}_L \quad \Gamma_2, B \longrightarrow G}{\Gamma_1, \Gamma_2, A \multimap B, !^{lin}F \longrightarrow G}\ \multimap_L$$

$$\dfrac{\dfrac{\Gamma, F \longrightarrow G[c/x]}{\Gamma, F \longrightarrow \forall x.G}\ \forall_R}{\Gamma !^{lin}F \longrightarrow \forall x.G}\ !^{lin}{}_L \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma, F \longrightarrow G[c/x]}{\Gamma, !^{lin}F \longrightarrow G[c/x]}\ !^{lin}{}_L}{\Gamma, !^{lin}F \longrightarrow \forall x.G}\ \forall_R$$

Notice that the cases for $!^{k_K}{}_R$, $!^{h_K}{}_R$, and $?^{sR_K}{}_L$ do not appear because of their side-conditions and because $!^{lin}$ is only used when the encoded formula is linear, that is, its main connective is not a modality. Consider for instance the following derivation where we introduce a $!^{lin}$ on the left:

$$\dfrac{\Gamma, F \longrightarrow !^{k_K}G}{\Gamma, !^{lin}F \longrightarrow !^{k_K}G}\ !^{lin}{}_L$$

From our encoding, F's main connective cannot be of the form $!^{k_{K_i}}$ nor $!^{h_{K_i}}$. Hence, it is not possible to introduce the bang to the right.

By eagerly applying the transformations above, we can transform an arbitrary proof Ξ of a sequent $[\![\Gamma]\!]_L \longrightarrow [\![G]\!]_R$ to a proof where for every occurrence of a $!^{lin}{}_L$ rule in the resulting proof of the form below

$$\dfrac{\Gamma, F \longrightarrow G}{\Gamma, !^{lin}F \longrightarrow G}\ !^{lin}{}_L$$

such that the premise $\Gamma, F \longrightarrow G$ is introduced by a rule introducing the formula F.

Now, we can show a correspondence between the derivations in the proof system for LAL and the derivations with the property above SELL. We show some of the cases. The most interesting cases are the left-introduction rules. The right-introduction rules were already shown before.

$$\dfrac{\Gamma, A, B \longrightarrow G}{\Gamma, A \otimes B \longrightarrow G}\ \otimes_L \quad \leftrightsquigarrow \quad \dfrac{\dfrac{[\![\Gamma]\!]_L, [\![A]\!]_L, [\![B]\!]_L \longrightarrow [\![G]\!]_R}{[\![\Gamma]\!]_L, [\![A]\!]_L \otimes [\![B]\!]_L \longrightarrow [\![G]\!]_R}\ \otimes_L}{[\![\Gamma]\!]_L, !^{lin}([\![A]\!]_L \otimes [\![B]\!]_L) \longrightarrow [\![G]\!]_R}\ !^{lin}$$

$$\dfrac{\Gamma_1 \longrightarrow A \quad \Gamma_2, B \longrightarrow G}{\Gamma_1, \Gamma_2, A \multimap B \longrightarrow G} \multimap_L \quad \leftrightsquigarrow \quad \dfrac{\dfrac{[\![\Gamma_1]\!]_L \longrightarrow [\![A]\!]_R \quad [\![\Gamma_2]\!]_L, [\![B]\!]_L \longrightarrow [\![G]\!]_R}{[\![\Gamma_1]\!]_L, [\![\Gamma_2]\!]_L, [\![A]\!]_R \multimap [\![B]\!]_L \longrightarrow [\![G]\!]_R} \multimap_L}{[\![\Gamma_1]\!]_L, [\![\Gamma_2]\!]_L, !^{lin}([\![A]\!]_R \multimap [\![B]\!]_L) \longrightarrow [\![G]\!]_R} !^{lin}$$

$$\dfrac{\Gamma, F_i \longrightarrow G}{\Gamma, F_1 \,\&\, F_2 \longrightarrow G} \&_{L_i} \quad \leftrightsquigarrow \quad \dfrac{\dfrac{[\![\Gamma]\!]_L, [\![F_i]\!]_L \longrightarrow [\![G]\!]_R}{[\![\Gamma]\!]_L, [\![F_1]\!]_L \,\&\, [\![F_2]\!]_L \longrightarrow [\![G]\!]_R} \&_{L_i}}{[\![\Gamma]\!]_L, !^{lin}([\![F_1]\!]_L \,\&\, [\![F_2]\!]_L) \longrightarrow [\![G]\!]_R} !^{lin}$$

The remaining cases are similar. □

2.2. *Additional Constructs using SELL*

We can use subexponentials to partition policy rules into hierarchies and control their use. Intuitively, higher ranked policies can only be used by principals with higher credentials, such as system administrators, while lower-ranked policies can also be used by other principals with lower credentials. We show how to specify when such policies can and cannot be used in a proof in a simple and *declarative fashion* by using SELL's subexponentials. For simplicity, assume that, besides the set of global policies, there are only two different sets of policy rules a lower-ranked, Γ_L, and a higher-ranked, Γ_H. The general case where there are a greater number of types of policy rules can be specified in a similar fashion.

Formally, we extend the system described in Section 2.1 with five more indexes:

$$I_{\mathcal{K}}^{LH} = I_{\mathcal{K}} \cup \{l, h, e_l, e_h, e_{lh}\}.$$

Intuitively, l and h are used to mark formulas specifying the lower and higher-ranked policies as follows $!^{l}\{\Gamma_L\}$ and $!^{h}\{\Gamma_H\}$; the index e_l is used to *disallow* the use of lower-ranked policies; the index e_h is used to *disallow* the use of higher-ranked policies; and the index e_{lh} is used to *disallow* the use of both higher and lower-ranked policies. Since policies can be used in an unrestricted fashion, we assume that l and h are unrestricted indexes, *i.e.*, $l, h \in \mathcal{U}$. The previous partial order relation among the indexes is extended as depicted in Figure 3. The subexponential signature specifying this system is denoted by $\Sigma_{\mathcal{K}}^{LH}$.

The derivation below illustrates, formally, the use of e_l to disallow the use of lower ranked policies in a derivation.

$$\dfrac{\dfrac{\Gamma \longrightarrow F}{\Gamma \longrightarrow !^{e_l} F} {!^{e_l}}_R}{\Gamma, !^{l}\{\Gamma_L\} \longrightarrow !^{e_l} F} \, n \times W$$

Notice that according to the preorder depicted in Figure 3, to introduce $!^{e_l}$ on the right one needs to weaken all the formulas marked with $!^{l}$, that is, weaken the lower-ranked policies. Hence, the formula F should be provable without using lower ranked policies. The same reasoning applies to e_h, and e_{lh}, but for, respectively, higher-ranked policies and both higher and lower-ranked policies. The subexponential e_{lh} will also play an important role for our PSPACE-completeness result described in Section 5.

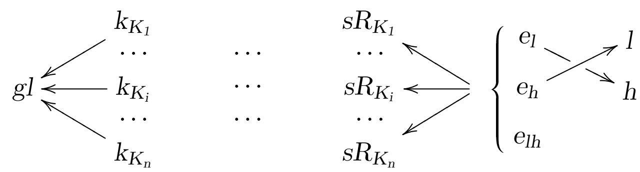

Figure 3: Graphical representation of the partial order $\preceq$ among subexponential indexes. Here if $a \longrightarrow b$ means that $a \preceq b$. The bracket denotes that the three indexes e_l, e_h, and e_{lh} are less than sR_K for all principals $K \in \mathcal{K}$, *e.g.*, $e_{lh} \preceq sR_{K_1}$. Notice that the indexes l and h are not related to the indexes sR_K, sL_K, h_K nor k_K. The elided part corresponds to the same sub-graph as in Figure 2. The subexponential signature specifying this system is denoted by $\Sigma_{\mathcal{K}}^{LH}$, where $\mathcal{K} = \{K_1, \ldots, K_n\}$.

- $(\mathcal{K}_1 \otimes \mathcal{K}_2)[i] = \begin{cases} \mathcal{K}_1[i] \cup \mathcal{K}_2[i] & \text{if } i \notin \mathcal{U} \\ \mathcal{K}_1[i] & \text{if } i \in \mathcal{U} \end{cases}$
- $\mathcal{K}[\mathcal{S}] = \bigcup\{\mathcal{K}[i] \mid i \in \mathcal{S}\}$
- $(\mathcal{K} +_l F)[i] = \begin{cases} \mathcal{K}[i] \cup \{F\} & \text{if } i = l \\ \mathcal{K}[i] & \text{otherwise} \end{cases}$
- $\mathcal{K} \leq_i [l] = \begin{cases} \mathcal{K}[l] & \text{if } i \preceq l \\ \emptyset & \text{if } i \npreceq l \end{cases}$
- $(\mathcal{K}_1 \star \mathcal{K}_2) \mid_{\mathcal{S}}$ is true if and only if $(\mathcal{K}_1[j] \star \mathcal{K}_2[j])$ for all $j \in \mathcal{S}$.

Figure 4: Specification of operations on contexts. Here, $i \in I$, $\mathcal{S} \subseteq I$, and the binary connective $\star \in \{=, \subset, \subseteq\}$.

For a small example using the constructs above, consider the following theory:

$$\text{admin } \mathit{knows}\,(\text{superuser}(\mathit{bob})) \otimes \mathit{bob\ says}\,(\mathit{alice\ has}\ P) \multimap \mathit{alice\ has}\ P$$
$$\text{admin } \mathit{knows}\,(\text{user}(\mathit{bob})) \otimes\, !^{e_h}\mathit{bob\ says}\,(\mathit{alice\ has}\ P) \multimap \mathit{alice\ has}\ P$$

The first clause specifies that if the administrator knows that the principal *bob* is a super-user and if *bob* is able to derive from both lower and higher-ranked policies that *alice* has access to P, then *alice* has access to P. On the other hand, the second clause specifies that if administrator knows that *bob* is a normal user, then *bob* may only use the lower ranked policies Γ_L to show that *alice* has access to some resource P. In both cases, however, one can use the global policies Θ.

2.3. Focusing with Subexponentials

We review the focused proof system for SELL. Focusing was first introduced by Andreoli [3] for linear logic in the context of logic programming to reduce the non-determinism during proof search. Focused proofs can be interpreted as the *normal form proofs for proof search* and are of great interest for systems such as PCA, where agents need to search for proofs. In this paper, focusing is also used in Section 4 to demonstrate the correspondence between provability using bipoles and MSR reachability.

Nigam in [26] proposed a focused proof system for classical linear logic with subexponentials. Here, we review its intuitionistic version. A similar system also appears in [7]. Before we introduce the system, we need some more terminology. We classify as *negative* all formulas whose main connective is $\&, \multimap, \forall, ?^l$ and the unit $\bot$, and classify the remaining non-atomic formulas as *positive*.[2]

As in the focused system for classical linear logic with subexponentials [28], we make use of indexed contexts $\mathcal{K}$ that maps a subexponential index to multiset of formulas, *e.g.*, if l is a subexponential index, then $\mathcal{K}[l]$ is a multiset of formulas, where intuitively they are all marked with $!^l$. That is, $\mathcal{K}[l] = \{F_1, \ldots, F_n\}$ should be interpreted as the multiset of formulas $!^l F_1, \ldots, !^l F_n$ prefixed with a $!^l$. We also make use of the operations on contexts depicted in Figure 4. Most of the operations are straightforward. For instance, $\mathcal{K}_1 \otimes \mathcal{K}_2[i]$ is used to specify the tensor right introduction rule ($\otimes_r$) and linear implication left rule ($\multimap_l$). $\mathcal{K}_1 \otimes \mathcal{K}_2[i]$ is defined as follows: when i is a bounded subexponential, $\mathcal{K}_1 \otimes \mathcal{K}_2[i]$ is obtained by multiset union of $\mathcal{K}_1[i]$ and $\mathcal{K}_2[i]$, and when i is an unbounded subexponential, then it is $\mathcal{K}_1[i]$.[3]

The rules for the multiplicative fragment together with $\top$ of the focused proof system SELLF are depicted in Figure 5. The rules for the remaining connective can be easily added, see [28], but for our purposes this fragment will be enough. In particular, SELLF contains four types of sequents.

1. $[\mathcal{K} : \Gamma], \Delta \longrightarrow \mathcal{R}$ is an unfocused sequent, where $\mathcal{R}$ is either a bracketed context $[F]$ or an unbracketed context. Here Γ contains only atomic or negative formulas, while $\mathcal{K}$ is the indexed context containing formulas whose main connective is a $!^l$ for some subexponential index l.
2. $[\mathcal{K} : \Gamma] \longrightarrow [F]$ is a sequent representing the end of the negative (or asynchronous) phase.
3. $[\mathcal{K} : \Gamma] -_F\!\!\rightarrow$ is a sequent focused on the right.
4. $[\mathcal{K} : \Gamma] \xrightarrow{F} G$ is a sequent focused on the left.

As one can see from inspecting the proof system in Figure 5, proofs are composed of two alternating phases, a negative phase, containing sequents of the first form above and where all the negative non-atomic formulas to the right and all the positive non-atomic formulas to the left are introduced. Atomic or positive formulas to the right and atomic or negative formulas to the left are bracketed by the $[]_l$ and $[]_r$ rules, while formulas whose main connective is a $!^l$ are added to the indexed context $\mathcal{K}$ by rule $!^l{}_L$. The second type of sequent above marks the end of the negative phase. A positive phase starts by using the decide rules to focus either on a formula on the right or on the left, resulting on the third and fourth sequents above. Then one introduces all the positive formulas to the right and the negative formulas to the left, until one is focused either on a negative formula on the right or a positive formula on the left. This point marks the end of the positive phase by using the R_l and R_r rules and starting another

[2] Andreoli's original focusing theorem for linear logic [3] assumed a global polarity assignment for atomic formulas. Here, our proof system is constructed assuming a positive polarization for atomic formulas. One could construct more elaborated focused proof systems for SELL, which allow a flexible polarity assignment. However, this is out of the scope of this paper.

[3] As specified by the side-condition of the $\otimes_r$ and $\multimap_l$ rule in Figure 5, there is an invariant that $\mathcal{K}_1[i] = \mathcal{K}_2[i]$ when i is unbounded. Therefore, for unbounded indexes, we use $\mathcal{K}_1$, but one could alternatively have used $\mathcal{K}_2$ or even their intersection.

Negative Phase

$$\frac{}{[\mathcal{K}:\Gamma],\Delta \longrightarrow \top}\ \top_r \qquad \frac{[\mathcal{K}:\Gamma],\Delta,F,G \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,F\otimes G \longrightarrow \mathcal{R}}\ \otimes_l \qquad \frac{[\mathcal{K}:\Gamma],\Delta,F \longrightarrow G}{[\mathcal{K}:\Gamma],\Delta \longrightarrow F \multimap G}\ \multimap_r$$

$$\frac{[\mathcal{K}:\Gamma],\Delta \longrightarrow G[c/x]}{[\mathcal{K}:\Gamma],\Delta \longrightarrow \forall x.G}\ \forall_r \qquad \frac{[\mathcal{K}:\Gamma],\Delta,G[c/x] \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,\exists x.G \longrightarrow \mathcal{R}}\ \exists_l \qquad \frac{[\mathcal{K}+_l F:\Gamma],\Delta \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,!^l F \longrightarrow \mathcal{R}}\ !^l{}_l$$

Positive Phase

$$\frac{[\mathcal{K}_1:\Gamma_1]-_F\!\!\rightarrow \quad [\mathcal{K}_2:\Gamma_2]-_G\!\!\rightarrow}{[\mathcal{K}_1\otimes\mathcal{K}_2:\Gamma_1,\Gamma_2]-_{F\otimes G}\!\!\rightarrow}\ \otimes_r,\ \text{where } (\mathcal{K}_1 = \mathcal{K}_2)|_{\mathcal{U}}$$

$$\frac{[\mathcal{K}_1:\Gamma_1]-_F\!\!\rightarrow \quad [\mathcal{K}_2:\Gamma_2]\xrightarrow{H}[G]}{[\mathcal{K}_1\otimes\mathcal{K}_2:\Gamma_1,\Gamma_2]\xrightarrow{F\multimap H}[G]}\ \multimap_l\ ,\ \text{where } (\mathcal{K}_1 = \mathcal{K}_2)|_{\mathcal{U}}$$

$$\frac{[\mathcal{K}:\Gamma]-_{G[t/x]}\!\!\rightarrow}{[\mathcal{K}:\Gamma]-_{\exists x.G}\!\!\rightarrow}\ \exists_r \qquad \frac{[\mathcal{K}:\Gamma]\xrightarrow{F[t/x]}[G]}{[\mathcal{K}:\Gamma]\xrightarrow{\forall x.F}[G]}\ \forall_l \qquad \frac{[\mathcal{K}\leq_l:\cdot],\Delta \longrightarrow F}{[\mathcal{K}:\cdot],\Delta-_{!^l F}\!\!\rightarrow}\ !^l{}_r\ \star$$

$$\frac{[\mathcal{K}\leq_l:\cdot],F \longrightarrow [\cdot]}{[\mathcal{K}:\cdot]\xrightarrow{?^l F}[?^k G]}\ ?^l{}_l\ \star \text{ and } k\in\mathcal{U}\wedge l\not\leq k \qquad \frac{[\mathcal{K}\leq_l:\cdot],F \longrightarrow [?^k G]}{[\mathcal{K}:\cdot]\xrightarrow{?^l F}[?^k G]}\ ?^l{}_l\ \star \text{ and } l\leq k$$

$$\frac{}{[\mathcal{K}:\Gamma]-_A\!\!\rightarrow}\ I_r \text{ given } A\in(\Gamma\cup\mathcal{K}[\mathcal{I}]) \text{ and } (\Gamma\cup\mathcal{K}[\mathcal{I}\setminus\mathcal{W}])\subseteq\{A\}$$

Structural Rules

$$\frac{[\mathcal{K}:\Gamma,N_a],\Delta \longrightarrow \mathcal{R}}{[\mathcal{K}:\Gamma],\Delta,N_a \longrightarrow \mathcal{R}}\ []_l \qquad \frac{[\mathcal{K}:\Gamma],\Delta \longrightarrow [P_a]}{[\mathcal{K}:\Gamma],\Delta \longrightarrow P_a}\ []_r \qquad \frac{[\mathcal{K}:\Gamma],P_a \longrightarrow [F]}{[\mathcal{K}:\Gamma]\xrightarrow{P_a}[F]}\ R_l \qquad \frac{[\mathcal{K}:\Gamma] \longrightarrow N}{[\mathcal{K}:\Gamma]-_N\!\!\rightarrow}\ R_r$$

$$\frac{[\mathcal{K}:\Gamma]\xrightarrow{F}[G]}{[\mathcal{K}+_l F:\Gamma] \longrightarrow [G]}\ D_l, \text{provided, } l\notin\mathcal{U} \qquad \frac{[\mathcal{K}+_l F:\Gamma]\xrightarrow{F}[G]}{[\mathcal{K}+_l F:\Gamma] \longrightarrow [G]}\ D_l, \text{provided, } l\in\mathcal{U}$$

$$\frac{[\mathcal{K}:\Gamma]\xrightarrow{F}[G]}{[\mathcal{K}:\Gamma,F] \longrightarrow [G]}\ D_l \qquad \frac{[\mathcal{K}:\Gamma]-_G\!\!\rightarrow}{[\mathcal{K}:\Gamma] \longrightarrow [G]}\ D_r \qquad \frac{[\mathcal{K}:\Gamma]-_G\!\!\rightarrow}{[\mathcal{K}:\Gamma] \longrightarrow [?^l G]}\ D_r$$

Figure 5: Focused Proof System for Intuitionistic Linear Logic with Subexponentials. Here, $\mathcal{R}$ stands for either a bracketed context, $[F]$, or an unbracketed context. A is an atomic formula; P_a is a positive or atomic formula; N is a negative formula; and N_a is a negative or atomic formula. In the $?^l$ and $!^l$ rules, $\star$ stands for "given $\mathcal{K}[\{x \mid l\not\leq x\wedge x\notin\mathcal{U}\}] = \emptyset$]."

negative phase.

One can prove the following soundness and completeness theorem following the same lines as the proof in Nigam's thesis [26] for the focused proof system for classical linear logic with subexponentials, using the technique introduced by Miller and Saurin in [24].

Theorem 2.3. *The sequent* $\longrightarrow G$ *is provable in SELL if and only if the sequent* $[\mathcal{K} : \cdot], \cdot \longrightarrow G$ *is provable in SELLF, where* $\mathcal{K}[l] = \emptyset$ *for all indexes* l*.*

Remark: Notice that focusing is lost whenever an exponential is introduced. As our encoding of LAL described in Section 2.1 contains many bangs and question-marks, its focusing behavior is quite shallow. A particular problem with that encoding is the $!^{lin}$ used for the encoding of linear formulas. Under the focusing disciplines this amounts to the loss of focusing whenever the encoding of a linear LAL formula is focused on. The following encoding fixes this problem:

$$
\begin{array}{lcl@{\qquad}lcl}
[\![K\ \mathit{has}\ C]\!]_L^{P/N} & = & !^{h_K}[\![C]\!]_L^N & [\![K\ \mathit{has}\ C]\!]_R & = & !^{h_K}[\![C]\!]_R \\
[\![K\ \mathit{knows}\ C]\!]_L^{P/N} & = & !^{k_K}[\![C]\!]_L^N & [\![K\ \mathit{knows}\ C]\!]_R & = & !^{k_K}[\![C]\!]_R \\
[\![K\ \mathit{says}\ C]\!]_L^{P/N} & = & !^{sL_K}?^{sR_K}[\![C]\!]_L^N & [\![K\ \mathit{says}\ C]\!]_R & = & ?^{sR_K}[\![C]\!]_R \\
[\![F \otimes G]\!]_L^P & = & [\![F]\!]_L^P \otimes [\![G]\!]_L^P & [\![F \otimes G]\!]_R & = & [\![F]\!]_R \otimes [\![G]\!]_R \\
[\![F \multimap G]\!]_L^N & = & [\![F]\!]_R \multimap [\![G]\!]_L^N & [\![F \multimap G]\!]_R & = & [\![F]\!]_L^P \multimap [\![G]\!]_R \\
[\![\exists x.F]\!]_L^P & = & \exists x.[\![F]\!]_L^P & [\![\exists x.F]\!]_R & = & \exists x.[\![F]\!]_R \\
[\![\forall x.F]\!]_L^N & = & \forall x.[\![F]\!]_L^N & [\![\forall x.F]\!]_R & = & \forall x.[\![F]\!]_R \\
[\![\top]\!]_L^{P/N} & = & \top & [\![\top]\!]_R & = & \top \\
[\![A]\!]_L^{P/N} & = & !^{lin}A & [\![A]\!]_R & = & A \\
[\![N]\!]_L^P & = & !^{lin}[\![N]\!]_L^N & [\![P]\!]_L^N & = & [\![P]\!]_L^P
\end{array}
$$

Here, A is an atomic formula; N is a negative formula; and P is a positive formula. We use three encodings $[\![\cdot]\!]_L^P$, $[\![\cdot]\!]_L^N$, and $[\![\cdot]\!]_R$. Intuitively, the first two are used, respectively, when encoding negative occurrences of positive polarity and negative polarity formulas, while the third one is used to encode positive occurrences. Notice that differently from the encoding used in Section 2.1, the encoding above only introduces $!^{lin}$ on atomic formulas and when a positive encoding $[\![\cdot]\!]_L^P$ meets a negative formula. That is, the $!^{lin}$ appear only in the intersection of focusing phases. We do not pursue further the encoding above in this paper, but we point out that by using the encoding above, one obtains similar focusing behaviors as proofs obtained from the focused proof system for LAL proposed in [10].

3. Undecidability

We show that the provability problem for propositional multiplicative fragment of LAL, as described in Section 2.1, which is equivalent to the logic described in [14], is *undecidable*. In particular, we encode a two-counter machine [25], which is known to be Turing complete, as a linear authorization theory. Notice that in our encoding we do *not* use the extra expressiveness described in Section 2.2.

This result is important in the context of PCA, as it shows that PCA using simple linear authorization policies may be not feasible. Moreover, this undecidability result is also interesting from a proof complexity point of view. It is has been shown that the provability problem for propositional multiplicative additive linear logic with exponentials (MAELL) is undecidable [22]. The same problem, however, for propositional multiplicative linear logic with exponentials (MELL) is still open. In fact, it is believed to be decidable [9]. The difference between MELL and the MELL fragment of LAL is the presence of different modalities, such as says, has, and knows. As we show in our encoding, these modalities play a crucial role for the sound and complete encoding of two-counter Minsky machines, namely for specifying the 0-test instructions. Although we are still not able to make any claims about the upper-bound of MELL, it is still interesting that the use of extra modalities leads already to undecidability. This is also in accordance with the results in [7], where Chaudhuri shows that the MELL fragment of SELL is also undecidable.

Two-Counter Minsky Machines Let M be a standard two-counter machine containing two registers r_1 and r_2 with natural numbers. Assume that M contains two types of instructions one for a-states and another for b-states. The instructions are depicted in Figure 6. Instructions of M specify its state transition rules. We assume that no instructions are labeled with the same state. The initial state is a_1 and the final state is a_0. Furthermore, a_0 is a halting state so it is distinct from the label of any of M's instructions.

M's configuration is a triple of the form $\langle m, n_1, n_2\rangle$, where m is a state, while n_1 and n_2 are the values of the registers r_1 and r_2. A *computation* performed by M is a sequence of M's configurations such that each step is obtained by applying one of M's instructions: $\langle a_1, n, 0\rangle \xrightarrow{a_1} \cdots \langle a_i, n_i, m_i\rangle \xrightarrow{a_i} \langle b_k, n_k, m_k\rangle \xrightarrow{b_k} \cdots$. A terminating computation is one that ends with a configuration of the form $\langle a_0, 0, 0\rangle$ where the values of the registers are both 0.[4]

Encoding Two-Counter Minsky Machines We assume the existence of only two principals A and B. Intuitively, A will be responsible for incrementing and decrementing the register r_1, while B will be responsible for the register r_2.

A machine configuration is encoded as a sequent as follows: The value of the register r_1 is the number of occurrences of $A\,\mathit{has}\,r_1$ formulas in the sequent, while the value of the register r_2 is the number of occurrences of $B\,\mathit{has}\,r_2$ formulas in the sequent. The state of the configuration is encoded as the formula appearing to the right-hand-side of the sequent. If this formula is $A\,\mathit{says}\,a_k$, then the configuration's state is a_k and similarly, if this formula is $B\,\mathit{says}\,b_j$, then the configuration's state is b_j. For example, the following sequent is the translation of the machine M's configuration $\langle a_4, 2, 1\rangle$

$$!^{gl}\{\Theta_M\}, A\,\mathit{has}\,r_1, A\,\mathit{has}\,r_1, B\,\mathit{has}\,r_2 \longrightarrow A\,\mathit{says}\,a_4.$$

[4] Notice that the condition that the values of the registers to be zero are not strictly necessary. We could have assumed that the registers r_1 and r_2 store any values. However, assuming their emptiness will simplify the encoding of Minsky Machines in LAL.

(Add r_1) a_k:	$r_1 = r_1 + 1$; goto b_j
(Add r_2) b_k:	$r_2 = r_2 + 1$; goto a_j
(Sub r_1) a_k:	$r_1 = r_1 - 1$; goto b_j
(Sub r_2) b_k:	$r_2 = r_2 - 1$; goto a_j
(0-test r_1) a_k:	if $r_1 = 0$ then goto b_{j_1} else goto b_{j_2}
(0-test r_2) b_k:	if $r_2 = 0$ then goto a_{j_1} else goto a_{j_2}
(Jump$_1$) a_k:	goto b_j
(Jump$_1$) b_k:	goto a_j

Figure 6: Instructions of a two-counter Minsky machine.

ADD$_1$:	$(A\ has\ r_1 \multimap B\ says\ b_j) \multimap A\ says\ a_k$
ADD$_2$:	$(B\ has\ r_2 \multimap A\ says\ a_j) \multimap B\ says\ b_k$
SUB$_1$:	$(A\ has\ r_1 \otimes B\ says\ b_j) \multimap A\ says\ a_k$
SUB$_2$:	$(B\ has\ r_2 \otimes A\ says\ a_j) \multimap B\ says\ b_k$
0-IF$_1$:	$B\ has\ (B\ says\ b_{j_1}) \multimap A\ says\ a_k$
0-IF$_2$:	$A\ has\ (A\ says\ a_{j_1}) \multimap B\ says\ b_k$
0-ELSE$_1$:	$(A\ has\ r_1 \multimap B\ says\ b_{j_2}) \otimes A\ has\ r_1 \multimap A\ says\ a_k$
0-ELSE$_2$:	$(B\ has\ r_2 \multimap A\ says\ a_{j_2}) \otimes B\ has\ r_2 \multimap B\ says\ b_k$
JUMP$_1$	$B\ says\ b_j \multimap A\ says\ a_k$
JUMP$_2$	$A\ says\ a_j \multimap B\ says\ b_k$
FINAL	$1 \multimap A\ says\ a_0$

Figure 7: Translation of the instructions of a two-counter Minsky machine M as a set of linear authorization logic formulas Θ_M.

Instructions, on the other hand, are translated as the set of global policy rules, Θ_M, depicted in Figure 7.[5] In the derivations below, we will normally elide the $!^{gl}\{\Theta_M\}$ from the sequents, in order to improve presentation. We also assume that they are contracted and weakened whenever needed.

ADD$_i$ is the translation of the instruction Add r_i. Once the clause ADD$_1$, for example, is used by back-chaining on it, one obtains a derivation with the following shape containing one open premise:

$$\dfrac{\dfrac{}{A\ says\ a_k \longrightarrow A\ says\ a_k}\ I \quad \dfrac{\Gamma, A\ has\ r_1 \longrightarrow B\ says\ b_j}{\Gamma \longrightarrow A\ has\ r_1 \multimap B\ says\ b_j}\ \multimap_R}{\Gamma \longrightarrow A\ says\ a_k}\ \text{ADD}_1$$

Seeing this derivation from bottom-up, one can verify that it specifies M's Add r_1 instructions. In particular, its end sequent corresponds to a configuration $\langle a_k, m, n\rangle$, while the derivation's open premise corresponds to the configuration $\langle b_j, m+1, n\rangle$. The

[5] For better presentation we use the notation with $says$, has, and $knows$ modalities. However, formally these should be interpreted using the left encoding described in Section 2.1. For example, the clause ADD$_1$ is in fact $!^{lin}[(!^{h_A}!^{lin} r_1 \multimap ?^{sR_B} b_j) \multimap !^{sL_A} ?^{sR_A} !^{lin} a_k]$.

clause SUB_i and JUMP_i follow the same idea, only that SUB_1 consumes a has formula, specifying M's Sub instructions, while JUMP_1 just changes the formula appearing on the right-hand-side, specifying M's Jump instructions.

The most interesting clauses are the 0-IF_i clauses. In these clauses, we use the modalities explicitly to specify the if case of M's 0-test instructions. In particular, once one back-chains on the clause 0-IF_1, due to the restriction on has modalities, the formula $B\,\mathit{has}\,(B\,\mathit{says}\,b_{j_2})$ can only be introduced if there are no $A\,\mathit{has}\,r_1$ formulas in the context. The derivation obtained has therefore the following shape:

$$\dfrac{\dfrac{}{A\,\mathit{says}\,a_k \longrightarrow A\,\mathit{says}\,a_k}\,I \qquad \dfrac{\Gamma \longrightarrow B\,\mathit{says}\,b_{j_1}}{\Gamma \longrightarrow B\,\mathit{has}\,(B\,\mathit{says}\,b_{j_1})}\,\mathit{has}_R}{\Gamma \longrightarrow A\,\mathit{says}\,a_k}\,\text{0-IF}_1$$

with proviso that Γ has no occurrences of $A\,\mathit{has}\,r_1$. Intuitively, this proviso corresponds to the check that $r_1 = 0$. On the other hand, the operational semantics of the else part of the 0-test is captured by using the 0-ELSE_i clauses. In particular, once one back-chains on the clause 0-ELSE_1, one obtains a derivation with the following shape, where A_k is the formula $A\,\mathit{says}\,a_k$ and R_1 is the formula $A\,\mathit{has}\,r_1$:

$$\dfrac{\dfrac{}{A_k \longrightarrow A_k}\,I \qquad \dfrac{\dfrac{\Gamma, R_1 \longrightarrow B\,\mathit{says}\,b_j}{\Gamma \longrightarrow R_1 \multimap B\,\mathit{says}\,b_j}\,\multimap_R \qquad \dfrac{}{R_1 \longrightarrow R_1}\,I}{\Gamma, R_1 \longrightarrow (R_1 \multimap B\,\mathit{says}\,b_j) \otimes R_1}\,\otimes_R}{\Gamma, R_1 \longrightarrow A_k}\,\text{0-ELSE}_1$$

Notice that the number of $A\,\mathit{says}\,r_1$ in the open premise is the same as in the end-sequent. However, one can only use this clause if there is at least one $A\,\mathit{says}\,r_1$ in the context of the end-sequent, otherwise the right-most branch is not provable.

Finally, the clause FINAL is used to check for a terminating computation, when the state is a_0 and the values in the registers are both zero. This is illustrated by the following derivation obtained by back-chaining on FINAL

$$\dfrac{\dfrac{}{A\,\mathit{says}\,a_0 \longrightarrow A\,\mathit{says}\,a_0}\,I \qquad \dfrac{}{\cdot \longrightarrow 1}\,1_R}{\cdot \longrightarrow A\,\mathit{says}\,a_0}\,\text{FINAL}$$

Notice that one can only introduce 1 on the right when the left-hand-side is empty, that is, when the conclusion sequent corresponds to a state where both registers are 0.

From the discussion above, it should be clear that our encoding is complete. Soundness is more complicated. In particular, we need invariants on how says formulas may be moved when the context is split. The following two lemmas are enough. The first one states that if two says formulas appear on the left-hand-side of a sequent, then the sequent is not provable, while the second lemma states that if a says formula appears to the left-hand-side of a sequent that is provable, then there is a computation of M that does not contain any instance of the if case of the 0-test.

Lemma 3.1. *Let M be an arbitrary two-counter machine and Γ be an arbitrary multiset of formulas of the form $A\,\mathit{has}\,r_1$ and $B\,\mathit{has}\,r_2$. Let Θ_M be the theory encoding M's instructions. Then for any states q_j, q_i and q_k of M and for any principals*

$C, D, E \in \{A, B\}$ the sequent $!^{gl}\{\Theta_M\}, C\, \mathit{says}\, q_i, D\, \mathit{says}\, q_j, \Gamma \longrightarrow E\, \mathit{says}\, q_k$ is not provable.

Proof We proceed by contradiction. Assume that the sequent above is provable and consider its lowest height proof. We cannot apply the initial rule since there are at least two linear formulas, which cannot be weakened, to the left of the sequent, namely, $C\, \mathit{says}\, q_i$ and $D\, \mathit{says}\, q_j$. Hence the only alternative is to use one of the formulas in the theory Θ_M. We can also not use the clause FINAL, since to introduce the formula 1 the context must contain be empty, which is not the case due to the extra *says* formula. Moreover, one can easily check that at least one premise obtained by using any other clause in Θ_M also has at least two linear formulas of the form *says* formulas in the left-hand-side of the sequent. This contradicts the assumption of that the proof has the lowest height. □

Lemma 3.2. *Let M be an arbitrary two-counter machine M and Γ be a multiset of formulas containing only $A\, \mathrm{has}\, r_1$ and $B\, \mathrm{has}\, r_2$ formulas with multiplicity of m and n, respectively. Let Θ_M be the theory encoding M's instructions. For any $C, D \in \{A, B\}$ and any states q_j and q_k of M if the sequent $!^{gl}\{\Theta_M\}, D\, \mathit{says}\, q_j, \Gamma \longrightarrow C\, \mathit{says}\, q_k$ is provable, then there is an execution of M from the configuration $\langle q_k, m, n\rangle$ to the configuration $\langle q_j, 0, 0\rangle$, such that the execution does not contain any transition using the if case of a zero test instruction.*

Proof The proof is by induction on the height of the proof of $!^{gl}\{\Theta_M\}, D\, \mathit{says}\, q_j, \Gamma \longrightarrow C\, \mathit{says}\, q_k$. The base case is when the proof ends with an initial rule, in which case $\Gamma = \emptyset$ and $q_k = q_j$. That is, this proof corresponds to the zero length execution.

For the inductive case, one has to consider all possible ways to prove the sequent above. We show only the case for the clause ADD_1. The remaining cases follow the same reasoning:

$$\dfrac{A\, \mathit{says}\, a_k, \Gamma' \longrightarrow C\, \mathit{says}\, q_k \qquad \dfrac{D\, \mathit{says}\, q_j, \Gamma'', A\, \mathrm{has}\, r_1 \longrightarrow B\, \mathit{says}\, b_j}{D\, \mathit{says}\, q_j, \Gamma'' \longrightarrow A\, \mathrm{has}\, r_1 \multimap B\, \mathit{says}\, b_j}}{D\, \mathit{says}\, q_j, \Gamma \longrightarrow C\, \mathit{says}\, q_k}$$

where $\Gamma = \Gamma' \cup \Gamma''$. Notice that from Lemma 3.1, the formula $D\, \mathit{says}\, q_j$ has to be moved to the right branch, otherwise the resulting left premise would contain both $A\, \mathit{says}\, a_k$ and $D\, \mathit{says}\, q_j$ to the left and not be provable. From the inductive hypothesis on the left and right branches, we have that there is an execution from $\langle q_k, m', n'\rangle$ to $\langle a_k, 0, 0\rangle$ and moreover from $\langle b_j, m'' + 1, n''\rangle$ to $\langle q_j, 0, 0\rangle$, where $m = m' + m''$ and $n = n' + n''$. Since there is no if case of a zero test in any one of these two executions, we can join them as follows:

$$\langle q_k, m' + m'', n' + n''\rangle \longrightarrow \ldots \longrightarrow \langle a_k, m'', n''\rangle \xrightarrow{a_k} \langle b_j, m'' + 1, n''\rangle \longrightarrow \cdots \longrightarrow \langle q_j, 0, 0\rangle.$$

We now show that there is no transition corresponding to the if case of a zero test instruction. As described above, these instructions are specified by the clauses 0-IF_1

and 0-IF$_2$. Given Lemma 3.1, the only possible way to use, for instance, the clause 0-IF$_1$ would be as follows:

$$\frac{A\,says\,a_k, \Gamma' \longrightarrow C\,says\,q_k \quad \Gamma'', D\,says\,q_j \longrightarrow B\,has\,(B\,says\,b_{j_1})}{D\,says\,q_j, \Gamma \longrightarrow C\,says\,q_k}$$

where $\Gamma = \Gamma' \cup \Gamma''$ and where the formula $D\,says\,q_j$ moves to the right-branch. However, one cannot introduce $B\,has\,(B\,says\,b_{j_1})$ due to the presence of $D\,says\,q_j$ and therefore the right-premise of this derivation is not provable. □

With the lemmas above, we can easily show the soundness direction of the following soundness and completeness theorem:

Theorem 3.3. *Given a two-counter Minsky machine, M, and its translation Θ_M, then there is a terminating computation from $\langle a_1, n, 0\rangle$ if and only if the sequent encoding $\langle a_1, n, 0\rangle$ and the M's instructions, as described above, is provable in $SELL_{\Sigma_\mathcal{K}}$, where $\mathcal{K} = \{A, B\}$.*

Proof The completeness direction of our encoding of Minsky machines follows from the text above. We now complete the soundness direction using the Lemmas 3.1 and 3.2. The proof follows by induction on the height of proofs.

We show some of the inductive cases.

Let $\Gamma \longrightarrow Q\,says\,q$ be an arbitrary sequent encoding a configuration of M, where Γ is a multiset of $A\,has\,r_1$ and $B\,has\,r_2$ and $Q \in \{A, B\}$ and q is one of M's state, such that q is a a-state if Q is the principal A, and is a b-state if Q is the principal B.

Case ADD$_1$: Assume that a clause ADD$_1$ is the last one used in a proof of The derivation would then have the following shape, where $\Gamma = \Gamma_1 \cup \Gamma_2$:

$$\frac{\Gamma_1, A\,says\,a_k \longrightarrow Q\,says\,q \quad \Gamma_2 \longrightarrow A\,has\,r_1 \multimap B\,says\,b_j}{\Gamma_1, \Gamma_2 \longrightarrow Q\,says\,q}\ \text{ADD}_1$$

From the invertibility of $\multimap_R$ rule, we can assume that the right-premise is introduced by a $\multimap_R$ rule, obtaining a proof for the sequent $\Gamma_2, A\,has\,r_1 \longrightarrow B\,says\,b_j$. From the inductive hypothesis applied to this premise, we have that there is a terminating computation $\langle b_j, n_2 + 1, m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_0, n, m\rangle$, where n_2 and m_2 are, respectively, the multiplicity of $A\,has\,r_1$ and $B\,has\,r_2$ in Γ_2.

From Lemma 3.2 applied on the left-open premise, there is a computation from the configuration $\langle q, n_1, m_1\rangle \longrightarrow \cdots \longrightarrow \langle a_k, 0, 0\rangle$, where n_1 and m_1 are, respectively, the multiplicity of $A\,has\,r_1$ and $B\,has\,r_2$ in Γ_1 with no occurrence of a if case of the 0-test instructions. Hence, by adding n_2 and m_2 to the registers r_1 and r_2, respectively, of all the configurations of this computation, there is a computation from $\langle q, n_1 + n_2, m_1 + m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_k, n_2, m_2\rangle$.

We can now plug this computation with the computation above by using the (Add r_1) a_k instruction:

$$\langle q, n_1 + n_2, m_1 + m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_k, n_2, m_2\rangle \xrightarrow{a_k} \langle b_j, n_2 + 1, m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_0, n, m\rangle.$$

All other inductive cases are very similar, except the case for the 0-IF$_1$ and 0-IF$_2$. We show only the former case as the latter is symmetric.

$$\frac{\Gamma_1, A\,says\,a_k \longrightarrow Q\,says\,q \quad \Gamma_2 \longrightarrow B\,has\,(B\,says\,b_{j_1})}{\Gamma_1, \Gamma_2 \longrightarrow Q\,says\,q}\ \text{0-IF}_1$$

By permutation arguments, we can assume that the right-premise is introduced by a has_R rule. (If this is not the case, we can construct another proof obtained by permuting the application of 0-IF$_1$ rule upwards until we obtain such a proof.) Hence Γ_2 contains only $B\,\mathit{has}\,r_2$ formulas and we have a proof of the sequent $\Gamma_2 \longrightarrow B\,\mathit{says}\,b_{j_1}$. Applying the inductive hypothesis on this sequent, we have a terminating computation $\langle b_{j_1}, 0, m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_0, n, m\rangle$, where m_2 is the multiplicity of $B\,\mathit{has}\,r_2$ in Γ_2.

From Lemma 3.2 applied on the left-open premise, there is a computation from the configuration $\langle q, n_1, m_1\rangle \longrightarrow \cdots \longrightarrow \langle a_k, 0, 0\rangle$, where n_1 and m_1 are, respectively, the multiplicity of $A\,\mathit{has}\,r_1$ and $B\,\mathit{has}\,r_2$ in Γ_1 with no occurrence of a if case of the 0-test instructions. Hence, by adding m_2 to the registers r_2 of all the configurations of this computation, there is a computation from $\langle q, n_1, m_1 + m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_k, 0, m_2\rangle$.

We can now plug this computation with the computation above using the if case of the instruction (0-test r_1) a_k:

$$\langle q, n_1, m_1 + m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_k, 0, m_2\rangle \xrightarrow{a_k} \langle b_j, 0, m_2\rangle \longrightarrow \cdots \longrightarrow \langle a_0, n, m\rangle.$$

□

From the encoding above, we can infer that the undecidability of propositional multiplicative fragment of linear authorization logics.

Corollary 3.4. *The provability problem for the propositional multiplicative fragment of LAL is undecidable.*

4. Proof Search and MSR

This section paves the way for specifying a fragment of first-order linear authorization logics whose provability problem is PSPACE-complete on the size of the given formula. For this, we use the system introduced in Section 2.2, which allows one to express when a formula is provable without using policy rules. This type of operation allows us to formalize a correspondence between the provability problem and the reachability problem for multiset rewrite systems (MSR) by using the machinery described in Section 2.3.

Informally, the state of the system consists of a multiset of facts, specifying the affirmations, possessions, and knowledge of principals, and a state changes by means of rewrite rules that may remove facts from the state, while inserting other facts. However, as in MSR, we would like to determine whether a rule is applicable by using *easy* operations, *e.g.*, checking for membership. In order to capture this intuition, we use the expressiveness gained in Section 2.2, namely the ability of specifying when a formula can *only* be derivable without using policy rules.

Firstly, assume that the set of global policies Θ is empty. Moreover, since for simplicity we do not make a distinction between lower-ranked (Γ_L) and higher-ranked policies (Γ_H) in the remainder of this paper, let us assume that all policies are higher-ranked policies (see Section 2.2). Consider the following grammar with different types of formulas.

$$\begin{array}{llll} T & ::= \; K\,\mathit{says}\,A \mid K\,\mathit{has}\,A \mid K\,\mathit{says}\,T \mid K\,\mathit{has}\,T & & \\ Pr & ::= \; !^{e_{lh}} T \mid Pr \otimes Pr & Ps & ::= \; T \mid Ps \otimes Ps \\ Ps_n & ::= \; Ps \mid \exists x.Ps & P & ::= \; Pr \multimap Ps_n \mid \forall x.P \\ G & ::= \; !^{e_{lh}} T \otimes \top & & \end{array}$$

Here, A is an atomic formula. T-formulas are consumable possessions and affirmations of principals. Intuitively, a state of the system consists of a multiset of T-formulas. Notice that T-formulas do not contain knows formulas. As we comment later in this section, adding knows formulas easily leads to the undecidability of the logic.

Policy rules are specified as P-formulas, which are constructed using Pr-formulas (for pre-condition) and Ps_n (for post-condition with nonce creation). According to the grammar above, policy rules have the following shape:

$$\underbrace{\forall \vec{y}.[}_{\text{FV}} \; \underbrace{!^{e_{lh}} T_1 \otimes \cdots \otimes !^{e_{lh}} T_m}_{\text{Pre-condition}} \underbrace{\multimap \exists \vec{x}.}_{\text{Nonces}} \underbrace{[T_1' \otimes \cdots \otimes T_k']]}_{\text{Post-condition}} \tag{1}$$

Such a formula can be interpreted as a multiset rewrite rule. The existential variables, $\vec{x}$, appearing in the post-condition specify the creation of fresh values, also known as nonces in protocol security literature [11], while all free variables (FV) in the pre and post-condition appear in the universally quantified variables $\vec{y}$. Following terminology in proof theory [23], we call this fragment *bipoles*.[6]

The novelty with respect to usual encodings of MSR in linear logic [6, 28] is on the occurrences of $!^{e_{lh}}$ appearing before T-formulas in the pre-condition of P-formulas. As discussed in Section 2.2, this connective specifies that one should be able to prove the formulas T_is in the pre-condition without using any policy rules, *i.e.*, the T_is must be derivable only from the T-formulas in the state. The following derivation illustrates the shape of a derivation obtained when using in a proof an instance of a bipole as shown in Equation 1, where fresh values are created accordingly:

$$\frac{T_1'' \longrightarrow T_1 \quad \cdots \quad T_m'' \longrightarrow T_m \qquad !^{h}\{\Gamma_H\}, \mathcal{T}, T_1', \ldots, T_k' \longrightarrow G}{!^{h}\{\Gamma_H\}, \mathcal{T}, T_1'', T_2'', \ldots, T_m'' \longrightarrow G} \tag{2}$$

The derivation above can be seen as an inference rule, where from bottom-up this derivation behaves like a rewrite rule replacing the T-formulas $T_1'', \ldots, T_m''$ by the T-formulas $T_1', \ldots, T_k'$ appearing at the post-condition of the P-formula used. More importantly, however, all open premises except the right-most have to be proved *without* using any policy rules. This means that the derivations introducing these open premises are simple. In fact, the height of their derivations is bounded by the number of occurrences of modalities in the corresponding open premise (see Lemma 5.3). The paper [14] also points out the importance of such type of derivations in order to prove properties of policies.

[6] In fact, the class of bipoles is bit more general than the P-formulas above. However, for the lack of a better name and since P-formulas contain most bipoles, we use the same name.

G-formulas also deserve some explanation. They are of the form $!^{e_{lh}} T_G \otimes \top$, specifying the goal that one wants to prove (the T-formula T_G) and appearing at the right-hand-side of sequents. As in the pre-condition of P-formulas, the formula $!^{e_{lh}} T_G$ can be intuitively interpreted as checking whether the formula T_G is provable from the state of the system without using policy rules. On the other hand, the formula $\top$ specifies that if T_G is provable, then one is not interested on the remaining formulas ($\mathcal{T}$). Formally, G-formulas are introduced by derivations of the following form:

$$\dfrac{T'' \longrightarrow T_G \qquad \dfrac{}{!^{h}\{\Gamma_H\}, \mathcal{T} \longrightarrow \top}\ \top_R}{!^{h}\{\Gamma_H\}, \mathcal{T}, T'' \longrightarrow\ !^{e_{lh}} T_G \otimes \top} \tag{3}$$

That is, there is necessarily a T-formula T'' from which one can derive T_G and the right-branch is closed by the introduction of $\top$.

The use of $\top$ is a way of abstracting infinite computations. As argued in [6, 10], distributed systems are endless processes where principals exchange credentials and affirmations forever. Since proofs are finite, we need an abstraction. This is exactly the role that $\top$ is playing. There might be an infinite derivation introducing the right-branch of the derivation above, but by using $\top$, we specify that we are not really interested on it. We are only interested on determining whether the formula T_G can be derived and not on how the remaining credentials are used afterwards.

We can formally show that a sequent is provable if and only if it is provable using derivations of the shapes shown in Derivations 2 and 3. This soundness and completeness result is formally shown by using the soundness and completeness of the *focused discipline* for SELL [28] and the following auxiliary lemma, which is proved by using the fact that T-formulas are linear, that is, they cannot be contracted nor weakened.

Lemma 4.1. *Let $\Delta \cup \{T\}$ be a multiset of T-formulas. If the sequent $\Delta \longrightarrow T$ is provable in SELL, then Δ has exactly one T-formula,* i.e.*, the sequent has the form $T' \longrightarrow T$.*

Theorem 4.2. *Let $\mathcal{T}$ be a multiset of T-formulas, Γ_H be a multiset of P-formulas, and G be a G-formula. Let $\mathcal{R}$ be the set of inference rules obtained from the derivations corresponding to the P-formula in Γ_H (as shown in Derivation 2) and the derivation obtained from the G-formula G (as shown in Derivation 3). Then the sequent $!^{h}\{\Gamma_H\}, \mathcal{T} \longrightarrow G$ is provable in SELL if and only if it is provable using the rules in $\mathcal{R}$.*

Proof The soundness of the Derivations 2 and 3 is not an issue as they are obtained by using valid applications of SELL's rules. For showing the completeness direction, we rely on the completeness of SELLF (Theorem 2.3) and Lemma 4.1.

Figure 8 contains the focused derivation introducing a P-formula to the left, where the end sequent, as discussed in Section 4, contains only P-formulas and T-formulas. Hence the linear context (Γ) is empty. Notice that this derivation is very similar to the Derivation 2. In particular, the branches to the left have a indexed context $\mathcal{K}_1^i <_{e_{lh}}$ which do not contain policy rules as they must be weakened by the introduction of the $!^{e_{lh}}$. Moreover, its right-most-branch has the T-formulas $T_1', \ldots, T_l'$ in the context, where the eigenvariables replace the variables $\vec{x}$. (The T_j' formulas will be eventually be moved to the indexed context in the negative phase by using the $!^{x}{}_l$ rules. This is not

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\mathcal{K}_1^1 <_{e_{lh}} : \cdot] \longrightarrow [T_1]}{[\mathcal{K}_1^1 <_{e_{lh}} : \cdot] \longrightarrow T_1}\ []_r}{[\mathcal{K}_1^1 : \cdot] -_{!^{e_{lh}} T_1}\to}\ !^{e_{lh}}_r \quad \cdots \quad \dfrac{\dfrac{[\mathcal{K}_1^m <_{e_{lh}} : \cdot] \longrightarrow [T_m]}{[\mathcal{K}_1^m <_{e_{lh}} : \cdot] \longrightarrow T_m}\ []_r}{[\mathcal{K}_1^m : \cdot] -_{!^{e_{lh}} T_m}\to}\ !^{e_{lh}}_r}{[\mathcal{K}_1 : \cdot] -_{!^{e_{lh}} T_1 \otimes \cdots \otimes !^{e_{lh}} T_m}\to}\ \otimes_r \quad \dfrac{\dfrac{[\mathcal{K}_2 : \cdot], T_1', \ldots, T_l' \longrightarrow [G]}{[\mathcal{K}_2 : \cdot], \exists \vec{x}.[T_1' \otimes \cdots \otimes T_l'] \longrightarrow [G]}\ \exists_l, \otimes_l}{[\mathcal{K}_2 : \cdot] \xrightarrow{\exists \vec{x}.[T_1' \otimes \cdots \otimes T_l']} [G]}\ R_l}{[\mathcal{K}_1 \otimes \mathcal{K}_2 : \cdot] \xrightarrow{[!^{e_{lh}} T_1 \otimes \cdots \otimes !^{e_{lh}} T_m] \multimap \exists \vec{x}.[T_1' \otimes \cdots \otimes T_l']} [G]}\ \multimap_l}{[\mathcal{K}_1 \otimes \mathcal{K}_2 : \cdot] \xrightarrow{\forall \vec{y}.[!^{e_{lh}} T_1 \otimes \cdots \otimes !^{e_{lh}} T_m] \multimap \exists \vec{x}.[T_1' \otimes \cdots \otimes T_l']} [G]}\ n \times \forall_l}{[\mathcal{K}_1 \otimes \mathcal{K}_2 : \cdot] \longrightarrow [G]}\ D_l$$

Figure 8: Focused derivation introducing a P-formula on the left. Here $\mathcal{K}_1 = \mathcal{K}_1^1 \otimes \cdots \mathcal{K}_1^m$.

$$\dfrac{\dfrac{\dfrac{\dfrac{[\mathcal{K}_1 <_{e_{lh}} : \cdot] \longrightarrow [T]}{[\mathcal{K}_1 <_{e_{lh}} : \cdot] \longrightarrow T}\ []_r}{[\mathcal{K}_1 : \cdot] -_{!^{e_{lh}} T}\to}\ !^{e_{lh}}_r \quad \dfrac{\dfrac{}{[\mathcal{K}_2 : \cdot] \longrightarrow \top}\ \top_r}{[\mathcal{K}_2 : \cdot] -_{\top}\to}\ R_r}{[\mathcal{K}_1 \otimes \mathcal{K}_2 : \cdot] -_{!^{e_{lh}} T \otimes \top}\to}\ \otimes_r}{[\mathcal{K}_1 \otimes \mathcal{K}_2 : \cdot] \longrightarrow [!^{e_{lh}} T \otimes \top]}\ D_r$$

Figure 9: Focused derivation introducing a G-formula on the right.

shown in that derivation.) From the completeness of the focusing discipline, the end-sequent is provable in SELL if and only if it is provable using the focused derivation. However, the focuses derivation in Figure 8 does not impose any restrictions on the number of formulas appearing in the image of the context $\mathcal{K}_1^i <_{e_{lh}}$ for any $1 \leq i \leq m$, only that it does not contain any P-formulas. But from Lemma 4.1, we know that if they contain more than one formula, then the sequent is not provable. Therefore, the end-sequent is provable if and only if it is proved using the focused derivation where the image of $\mathcal{K}_1^i <_{e_{lh}}$ contains exactly one T-formula for all $1 \leq i \leq m$. This derivation corresponds exactly to the Derivation 2.

The focusing derivation introducing a G-formula on the right is depicted in Figure 9. The reasoning is similar as for the derivation introducing P-formulas. From the introduction of $!^{e_{lh}}$ on the right, the context $\mathcal{K}_1 <_{e_{lh}}$ does not contain any P-formulas. Moreover, from Lemma 4.1 it should contain exactly one T formula, corresponding exactly to the Derivation 3. □

Comparison with existing logics In order to illustrate the importance of $!^{e_{lh}}$ for proof search, consider the following clause which could be written in the logic presented in Section 2.1 or in [14] and the clauses, Θ_M, in Figure 7 encoding a two-counter Minsky machine M: $(i)\, A\, says\, a_k \multimap K\, has\, F$, where F is an arbitrary formula. The formula (i) specifies that if the principal A says a_k, then the principal K has the formula F. A derivation introducing (i) has the following shape:

$$\dfrac{!^h\{\Theta_M\}, \Gamma \longrightarrow A\, says\, a_k \quad !^h\{\Theta_M\}, \Gamma', K\, has\, F \longrightarrow G}{!^h\{\Theta_M\}, \Gamma, \Gamma' \longrightarrow G}\ (i)$$

As we have shown above, to prove the left-premise of the derivation above is undecidable in general. Therefore, checking whether one can use the clause (i) during proof search is not easy in general. On the other hand, by using $!^{e_{lh}}$ all premises except the right-most in a derivation introducing a P-formula (see Equation 2) can be proved (see Lemma 5.3) since those premises do not contain any P-formulas.

Adding knowledge leads to undecidability From the grammar shown above, one is not allowed to use formulas of the form $K\ knows\ P$. If we allow such formulas, then one can easily show that the provability problem is undecidable.

The proof of undecidability follows from a sound and complete encoding of the Horn implication problem with existentials, which has been shown to be undecidable even without function symbols [11]. In particular, we translate a Horn clause of the form $\forall \vec{y}.[A_1 \wedge \cdots \wedge A_n \supset \exists \vec{x}.A]$, as

$$\forall \vec{y}.[K\ knows\ A_1 \otimes \cdots \otimes K\ knows\ A_n \supset \exists \vec{x}.K\ knows\ A],$$

where $A, A_1, \ldots, A_n$ are atomic formulas and where we use a single principal K. Since $knows$ formulas are unrestricted, one can easily show, by induction on the height of derivations, the soundness and completeness of this translation. That is an atomic formula A is provable from a a Horn theory if and only if the formula $K\ knows\ A$ is provable from its translation. We leave the details to the reader.

Remark: One could refine P-formulas even further. For instance, one could allow formulas in the post-condition of an action to also have bangs marked with some subexponential index, $loc(k)$, denoting the location where some credential is stored. Then by using the same indexes in the bangs of the formulas appearing in the pre-condition, one could further enforce that a formula should be only proved using the facts that are in some particular location. For example, the formula $!^{loc(k1)}T \multimap\ !^{loc(k2)}T'$ specifies that the formula T should be proved using only the formulas in $loc(k1)$ and that the formula T' is to be stored in location $loc(k2)$. This seems to be an interesting application of subexponentials for which leave as future work.

5. PSPACE-completeness

This section shows that the provability problem for a fragment of the system introduced in Section 4 is PSPACE-complete. We use most of the machinery used in [18, 17] on the complexity of the reachability problem for MSRs and the machinery introduced in Section 4. In particular, based on a similar notion given in [21], we assume that all policy rules are *balanced*, that is, the number of facts in the pre and post conditions of actions is the same. Formally, in Eq. (1) $m = k$. That is, our policy rules are *balanced bipoles*. This restriction enforces that whenever a policy rule is used during proof search the number of T-formulas in the resulting right-most sequent in Derivation 2 does not change.

As in [21, 19], we assume a finite alphabet, $\mathcal{L}$, with no function symbols. Notice, however, that due to nonce creation, there can be an arbitrary number of symbols in a proof.

5.1. PSPACE-hardness

The PSPACE-hardness proof for balanced bipoles follows the same lines as the PSPACE-hardness in [18, 17], for which we briefly sketch. We encode a non-deterministic Turing Machine $\mathcal{M}$ that accepts in space n. We use a single principal K. Given $\mathcal{M}$, we construct a set of balanced policy rules as follows. First we introduce the following proposition $R_{i,\xi}$ which denotes that "i^{th} cell contains the symbol ξ", where $0 \leq i \leq n+1$ and ξ is a symbol from the alphabet of $\mathcal{M}$. Moreover, the proposition $S_{i,q}$ denotes that "the j^{th} cell is being scanned by $\mathcal{M}$ at state q". Assume without loss of generality that $\mathcal{M}$ has a single accepting state q_f and that all accepting configurations are of the same form, scanning cell v.

Then a machine configuration of $\mathcal{M}$ where it scans the cell j is state q and the string $\xi_0\xi_1 \ldots \xi_{n+1}$ is encoded as the multiset of T-formulas (specified in Section 4) $K\,\mathrm{has}\,(S_{j,q}), K\,\mathrm{has}\,(R_{0,\xi_0}), \ldots, K\,\mathrm{has}\,(R_{n+1,\xi_{n+1}})$. Finally, we encode by using $5(n+2)$ P-formulas, shown below, an instruction, γ, of $\mathcal{M}$ of the form $q\xi \to q'\eta D$ denoting "if in state q looking at symbol ξ, replace it by η, move to the direction D and to state q'.

$$
\begin{array}{l}
!^{e_{lh}}(K\,\mathrm{has}\,S_{i,q}) \otimes !^{e_{lh}}(K\,\mathrm{has}\,R_{i,\xi}) \multimap (K\,\mathrm{has}\,F_{i,\gamma}) \otimes (K\,\mathrm{has}\,R_{i,\xi})\\
!^{e_{lh}}(K\,\mathrm{has}\,F_{i,\gamma}) \otimes !^{e_{lh}}(K\,\mathrm{has}\,R_{i,\xi}) \multimap (K\,\mathrm{has}\,F_{i,\gamma}) \otimes (K\,\mathrm{has}\,H_{i,\gamma})\\
!^{e_{lh}}(K\,\mathrm{has}\,F_{i,\gamma}) \otimes !^{e_{lh}}(K\,\mathrm{has}\,H_{i,\gamma}) \multimap (K\,\mathrm{has}\,G_{i,\gamma}) \otimes (K\,\mathrm{has}\,H_{i,\gamma})\\
!^{e_{lh}}(K\,\mathrm{has}\,G_{i,\gamma}) \otimes !^{e_{lh}}(K\,\mathrm{has}\,H_{i,\gamma}) \multimap (K\,\mathrm{has}\,G_{i,\gamma}) \otimes (K\,\mathrm{has}\,R_{i,\eta})\\
!^{e_{lh}}(K\,\mathrm{has}\,G_{i,\gamma}) \otimes !^{e_{lh}}(K\,\mathrm{has}\,H_{i,\xi}) \multimap (K\,\mathrm{has}\,S_{i_D,q'}) \otimes (K\,\mathrm{has}\,R_{i,\eta})
\end{array}
$$

where $0 \leq i \leq n+1$, $F_{i,\gamma}, G_{i,\gamma}$ and $H_{i,\gamma}$ are auxiliary atomic formulas, and $i_D = i+1$ if D is "right", $i_D = i-1$ if D is "left" and $i_D = i$ otherwise.

From Theorem 4.2, each policy rule above can be interpreted as a multiset-rewrite rule which replaces T-formulas. For instance, the introduction of the first rule shown above behaves as a rule that replaces $K\,\mathrm{has}\,S_{i,q}, K\,\mathrm{has}\,R_{i,\xi}$ by $K\,\mathrm{has}\,F_{i,\gamma}, K\,\mathrm{has}\,R_{i,\xi}$.

The five policy rules above, when applied in sequence, that is one after another, corresponds to the following sequence of rewrites

$$
\begin{array}{l}
K\,\mathrm{has}\,S_{i,q}, K\,\mathrm{has}\,R_{i,\xi} \longrightarrow K\,\mathrm{has}\,F_{i,\gamma}, K\,\mathrm{has}\,R_{i,\xi} \longrightarrow\\
K\,\mathrm{has}\,F_{i,\gamma}, K\,\mathrm{has}\,H_{i,\gamma} \longrightarrow K\,\mathrm{has}\,G_{i,\gamma}, K\,\mathrm{has}\,R_{i,\eta} \longrightarrow K\,\mathrm{has}\,S_{i_D,q'}, K\,\mathrm{has}\,R_{i,\eta}
\end{array}
$$

which corresponds to the execution of the instruction γ, as the state is altered from q to q', the contents of the i^{th} cell is updated accordingly, from ξ to η, and the cell being read is also changed to i_D. One can show using the similar reasoning as in [18, 17], that the sequence of rewrites above is the only valid one. Further details can be found in [18, 17, Theorem 2]. Finally, when the final state is reached, one can finish the proof as follows:

$$
\dfrac{\dfrac{\dfrac{}{K\,\mathrm{has}\,S_{v,q_f} \longrightarrow K\,\mathrm{has}\,S_{v,q_f}}}{K\,\mathrm{has}\,S_{v,q_f} \longrightarrow !^{e_{lh}}(K\,\mathrm{has}\,S_{v,q_f})} \quad \dfrac{}{\Gamma \longrightarrow \top}}{\Gamma, K\,\mathrm{has}\,S_{v,q_f} \longrightarrow !^{e_{lh}}(K\,\mathrm{has}\,S_{v,q_f}) \otimes \top}
$$

We can then formally prove the following theorem.

Theorem 5.1. *Let $\mathcal{M}$ be a Turing machine that accepts in space n, I be an initial configuration, and q_f its final state. Let $\Gamma_{\mathcal{M}}$ be the set of balanced policy rules specifying*

$\mathcal{M}$'s instructions and Γ_I be the multiset of T-formulas encoding the configuration I as described above. Then $\mathcal{M}$ reaches the final configuration from I if and only if the sequent $!^{h}\{\Gamma_{\mathcal{M}}\}, \Gamma_I \longrightarrow\ !^{e_{lh}}(K\ \mathsf{has}\ S_{v,q_f}) \otimes \top$ is provable in SELL$_{\Sigma^{LH}_{\mathcal{K}}}$, where $\mathcal{K} = \{K\}$.

Given the theorem above, we can conclude the PSPACE-hardness of the provability problem for balanced bipoles.

Corollary 5.2. *The provability problem for balanced bipoles is PSPACE-hard.*

5.2. PSPACE upper bound

The PSPACE upper bound is more interesting and is where the machinery introduced in Section 2.2 pays off. Our PSPACE upper bound is on the following assumptions/inputs:

- $\mathcal{L}$ is finite first-order alphabet without function symbols with J predicate symbols and D constant symbols;
- k is an upper bound on the arity of predicate symbols;
- $\mathcal{P}$ is a finite multiset of balanced bipoles specifying the policy rules;
- $\mathcal{T}$ is a multiset of exactly m T-formulas specifying the initial contents of the sequent. Recall that since all policy rules are balanced bipoles, a policy rule removes and adds the same number of T-formulas from a sequent;
- G is G-formula appearing at the right-hand-side of the sequent.

The problem is to determine whether the sequent $!^{h}\{\mathcal{P}\}, \mathcal{T} \longrightarrow G$ is provable or not in SELL. Since SELL admits cut-elimination, it is enough to determine whether there is or not a *cut-free* proof introducing the sequent above.

Our PSPACE upper bound is proved by using some of the machinery in [18, 17] and the connections between proof search and MSR execution described in Section 4. However, a main difference to [18, 17] is that here we need to show that it is possible to determine in PSPACE whether one can use a policy rule while searching for a proof. In particular, as illustrated in the Derivation 2 in Section 4, we need to show that one can determine in PSPACE whether a sequent of the form $T_1 \longrightarrow T_2$ is provable or not, where T_1 and T_2 are T-formulas.

We define the size of a T-formula, F, written $|F|$, inductively as follows: $K\ \mathsf{has}\ T = K\ \mathsf{says}\ T = |T| + 1$, and the size of atomic formulas is zero, *i.e.*, $|A| = 0$. The following lemma provides a polynomial bound on the number of steps one needs to take in order to check whether a derivation is a proof the sequent $T_1 \longrightarrow T_2$. The lemma's proof follows from the observation that any (cut-free) derivation introducing the sequent $T_1 \longrightarrow T_2$ does not branch and has its height bounded by $|T_1| + |T_2|$.

Lemma 5.3. *Let T_1 and T_2 be two arbitrary T-formulas. The problem of determining whether the sequent $T_1 \longrightarrow T_2$ is provable or not is in NP. In particular, it takes $|T_1| + |T_2|$ steps to check whether an arbitrary cut-free derivation is a proof of the sequent $T_1 \longrightarrow T_2$.*

We also show by induction that, while searching for a (cut-free) proof, the size of T-formulas does not grow, *i.e.*, one cannot obtain T-formulas of arbitrary sizes.

Lemma 5.4. *Let $\mathcal{S} = !^{h}\{\mathcal{P}\}, \mathcal{T} \longrightarrow G$ be a sequent, such that the size of any occurrence of a T-formula (including sub-formulas) in $\mathcal{S}$ is bounded by M. Let Ξ be an arbitrary cut-free derivation introducing $\mathcal{S}$. Then the size of all occurrences of T-formulas (including sub-formulas) in Ξ are also bounded by M.*

From the parameters above, we obtain M by checking which T-formula appearing anywhere in $\mathcal{P}, \mathcal{T}$ and G, including subformulas, has the greatest size. In typical specifications, such as those given in [14], the value of M is less than 3. Given the lemmas above, we can conclude that the problem of determining whether a policy rule's pre-condition is derivable from some given T-formulas is in PSPACE.

We can now use the machinery given in [18, 17]. First we show the following upper bound on the number of different T-formulas in the system. Notice that following [18, 17], we fix a set with $2mk$ fresh constants to be used as nonces whenever needed. Using the same reasoning as [18, 17], we can show that with this number of constants one can always guarantee the freshness of nonces.

Lemma 5.5. *Let $\mathcal{L}$ be a finite alphabet and let M be an upper bound on the size of T-formulas. Then there are at most $MJ(D + 2mk)^k$ different T-formulas in the system, where the parameters are described above.*

The following theorem formalizes the PSPACE upper bound for the provability problem when using balanced bipoles.

Theorem 5.6. *Given a finite alphabet $\mathcal{L}$, a multiset $\mathcal{P}$ of balanced bipoles, a multiset $\mathcal{T}$ of T-formulas, and a G-formula G, then there is an algorithm that determines whether a sequent $!^{h}\{\mathcal{P}\}, \mathcal{T} \longrightarrow G$ is provable or not and runs in PSPACE with respect to the following parameters:*

1. *M is the upper bound on the size of T-formulas;*
2. *J and D are the number of predicates and constant symbols, respectively, in the alphabet $\mathcal{L}$;*
3. *m is the number of facts in $\mathcal{T}$;*
4. *k is an upper bound on the arity of predicate symbols in the alphabet $\mathcal{L}$.*

Proof The upper bound proof follows the same lines as described in [18, 17]. We just sketch it here.

We use the fact that PSPACE is equal to NPSPACE [31]. Let $i = 0$ and $C_i = \mathcal{T}$ and $G = T_G \otimes \top$. Check whether any formula in $\mathcal{T}$ entails T_G. If so, then output yes. If $i > mMJ(D + 2mk)^k$, then it means that we have encountered the same sequent twice, hence output no. Otherwise, choose non-deterministically a formula P in $\mathcal{P}$ such that its pre-condition is derivable from some formulas $T_1, \ldots, T_n$ in C_i. Construct C_{i+1} from C_i by replacing the T-formulas $T_1, \ldots, T_n$ by the post-condition of P. If necessary chose fresh nonces from the set of $2mk$ constants available. Finally let $i := i + 1$ and repeat.

We show that this algorithm runs in PSPACE. In particular, we can store in PSPACE the set of T-formulas in C_i since it has the same size as the size of $\mathcal{T}$. This is because all formulas in $\mathcal{P}$ are balanced bipoles. Moreover, we can store in PSPACE the value of i in binary as shown below:

$$\log(mMJ(D + 2mk)^k) = \log(m) + \log(M) + \log(J) + k \log(D + 2mk).$$

Finally from Lemma 5.3 and 5.4, one can always check in PSPACE whether the precondition of a formula in $\mathcal{P}$ is derivable from C_i. Hence the algorithm runs in polynomial space. □

6. Example

We show how to specify the student registration similar to the example described in [14] by using balanced bipoles. This example consists of a university registration example, where students may register to courses, which take place at specific timesslots. There are two main principals, a calendar, *cal*, which authorizes free time slots available, and a registrar, *reg*, that controls the entire registration process. We assume the following set of atomic formulas:

- $slot(S, T)$ denoting that the student S is available at time T;
- $cr(S, av/C)$ denoting that the student S has one available credit (av) or that he used a credit to register in the course C;
- $seat(C, av/S)$ denoting that a seat of the course C is available or occupied by the student S;
- $reg(S, C, T)$ denoting the student S is registered at the course C at the time T;
- $course(C, T)$ denoting the course C runs at time slot T.

The goal is to specify this system in such a way that (1) no student registers for more than a stipulated number of credits, (2) a student does not register for two courses that have the same timeslots, and (3) a maximum registration limit is respected. Here, for simplicity, we assume that each course requires one credit. (It is also possible to specify the general case, but that would require more rules.)

We assume that at the beginning of the semester, the registrar issues an initial number of certificates of the form $reg\,says\,(cr(S, av))$, for each student, and an initial number of certificates of the form $reg\,says\,(seat(C))$ and a unique certificate for each course C $reg\,says\,(course(C, T))$. Moreover, students get one certificate from the calendar of the form $cal\,says\,(slot(S, T, no))$ for all timeslots T.

The policy specifying this scenario is depicted in Figure 10. It specifies that if the course C at time T has an available seat and the student S has an available credit and is has the timeslot T available, then the student can register causing the seat to be occupied by the student, one of the student's credit to no longer be available and the calender to allocate the timeslot T of the student S as attending the course C.

Notice that since this policy rule behaves as a rewriting rule, it is straightforward to show that the requirements above for this scenario are all satisfied.

7. Conclusions and Related Work

This paper proposed a framework for specifying linear authorization logics that allow one to specify a wider range of policies. We then investigated the complexity

$$\forall C, S, T.[!^{e_{lh}} reg\ says\ (course(C, T)) \otimes !^{e_{lh}} reg\ says\ (seat(C, av)) \otimes \\ !^{e_{lh}} reg\ says\ (cr(S, av)) \otimes !^{e_{lh}} cal\ says\ (slot(S, T)) \multimap \\ reg\ says\ (course(C, T)) \otimes reg\ says\ (seat(C, S)) \otimes \\ reg\ says\ (cr(S, C)) \otimes cal\ says\ (reg(S, C, T))]$$

Figure 10: Balanced bipole specifying when a student may register a course.

of several linear authorization logics including existing logics. We have shown that the provability problem for the propositional multiplicative fragment is undecidable. Then by demonstrating novel connections to multiset rewriting systems, we have also identified a first-order fragment that is PSPACE-complete.

As previously discussed, we improve the work in [14] by proposing a general framework where different linear authorization logics can be specified, which allow for more policies to be specified. For instance, it does not seem possible to specify in the logic proposed in [14] when one is disallowed to use some policies in order to prove a formula. As illustrated by our complexity results, this extra expressiveness seems key to specify tractable fragments for these logics.

Cervesato and Scedrov [6] proposed a framework based on multiset rewriting (MSR) for specifying concurrent processes and also relate their system to linear logic provability. We share some of their concerns, in particular, in conciliating the fact that processes may run forever, while proofs are finite. Our solutions to this problem are similar. While [6] considers open derivations, we close them by using $\top$. [10] applies some of the ideas in [6] to the linear authorization logic proposed in [14]. From our work it seems possible to recover some of the results in [10]. Similarly to our work, [10] also makes use of a focused proof system to reason about policies. We strongly believe that the same reasoning techniques used in [10] can also be apply in our work.

On the complexity of authorization logics, [13] shows that provability problem for propositional classical authorization logics is also PSPACE-complete. On the other hand, there has also been a number of complexity results on linear logic (too many to cite them all here). For instance, [22] investigates the complexity of many fragments of propositional linear logic. In particular, [22] shows that the multiplicative additive fragment with exponentials is undecidable. The unpublished note [7] also shows that the propositional multiplicative fragment of linear logic with subexponentials is undecidable. However, up to our knowledge, this paper contains the first complexity results on linear authorization logics.

This paper also continues the on-going program of investigating MSR systems with balanced actions. In a series of papers [21, 19, 18, 17], we have investigated together with others the complexity for the reachability problem for such MSR systems. This paper capitalized and extends [21, 19, 18, 17] by investigating systems with modalities. For instance, we use the same ideas proposed in [18, 17] to overcome the fact that actions may create fresh values and therefore a proof may contain an unbounded number of symbols. Our PSPACE upper bound algorithm is a conservative extension of the PSPACE upper bound algorithms proposed in [21, 19, 18, 17].

As shown in [29], SELL provides a powerful framework for not only encoding proof systems and logics but also reasoning about them. In fact, we were able to automatically check that LAL admits cut-elimination using the encoding described in Section 2.1 by using the tool TATU. More details can be found at TATU's homepage [30].

The paper [10] proposes a focused proof system for LAL there relabeled as linear epistemic logic. Given our alternative encoding in remark in Section 2.3, it seems possible to recover all their results in SELLF. In particular, the focusing behaviors obtained by using our encoding and the proofs obtained by focused proof system proposed in [10] are the same. However, [10] proves that the completeness of their focused proof system, something that we do not do here. It would be interesting to check whether the proof can be obtained somehow from the encodings given in Section 2.1 and Section 2.3.

Recently, Kanovich *et al.* extended the system in [18, 17] to include explicit time [20]. There facts include a timestamp and rules may have a guard involving timestamps, which specify the temporal condition for a rule to be applicable. It seems possible to extend our framework to accomodate such construct. In particular, one would need to extend SELL with definitions [32] in order to capture the computation done with timestamps, similar to what was done in [28].

Finally, in our framework, it is not yet possible to quantify over principals. This would require the quantification over subexponential indexes. There are some challenges in proposing a system with such quantifiers that admits cut-elimination due to the relation among subexponential indexes. One needs to take extra care for the new principal case with the new quantifiers. This is our current subject of investigation.

Acknowledgments: We thank Deepak Garg, Cedric Fournet, Max Kanovich, Tajana Ban Kirigin, Hubert Comon-Lundh, Andre Scedrov, and Alwen Tiu for fruitful discussions.

[1] M. Abadi. Logic in access control (tutorial notes). In A. Aldini, G. Barthe, and R. Gorrieri, editors, *FOSAD*, volume 5705 of *Lecture Notes in Computer Science*, pages 145–165. Springer, 2009.

[2] M. Abadi, M. Burrows, B. W. Lampson, and G. D. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.

[3] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[4] A. W. Appel and E. W. Felten. Proof-carrying authentication. In *ACM Conference on Computer and Communications Security*, pages 52–62, 1999.

[5] K. D. Bowers, L. Bauer, D. Garg, F. Pfenning, and M. K. Reiter. Consumable credentials in linear-logic-based access-control systems. In *NDSS*. The Internet Society, 2007.

[6] I. Cervesato and A. Scedrov. Relating state-based and process-based concurrency through linear logic (full-version). *Inf. Comput.*, 207(10):1044–1077, 2009.

[7] K. Chaudhuri. On the expressivity of two refinements of multiplicative exponential linear logic. Unpublished, 2009.

[8] V. Danos, J.-B. Joinet, and H. Schellinx. The structure of exponentials: Uncovering the dynamics of linear logic proofs. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Kurt Gödel Colloquium*, volume 713, pages 159–171. Springer, 1993.

[9] P. de Groote, B. Guillaume, and S. Salvati. Vector addition tree automata. In *LICS*, pages 64–73. IEEE Computer Society, 2004.

[10] H. DeYoung and F. Pfenning. Reasoning about the consequences of authorization policies in a linear epistemic logic. Workshop on Foundations of Computer Security, Aug. 2009.

[11] N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[12] M. Fairtlough and M. Mendler. Propositional lax logic. *Inf. Comput.*, 137(1):1–33, 1997.

[13] D. Garg and M. Abadi. A modal deconstruction of access control logics. In R. M. Amadio, editor, *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 216–230. Springer, 2008.

[14] D. Garg, L. Bauer, K. D. Bowers, F. Pfenning, and M. K. Reiter. A linear logic of authorization and knowledge. In D. Gollmann, J. Meier, and A. Sabelfeld, editors, *ESORICS*, volume 4189 of *Lecture Notes in Computer Science*, pages 297–312. Springer, 2006.

[15] D. Garg and F. Pfenning. Non-interference in constructive authorization logic. In *CSFW*, pages 283–296. IEEE Computer Society, 2006.

[16] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[17] M. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. *Inf. Comput.* Accepted for Publication.

[18] M. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. In P. Degano, S. Etalle, and J. D. Guttman, editors, *Formal Aspects in Security and Trust*, volume 6561 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2010.

[19] M. Kanovich, P. Rowe, and A. Scedrov. Policy compliance in collaborative systems. In *CSF '09: Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, pages 218–233, Washington, DC, USA, 2009. IEEE Computer Society.

[20] M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, C. L. Talcott, and R. Perovic. A rewriting framework for activities subject to regulations. In A. Tiwari, editor, *RTA*, volume 15 of *LIPIcs*, pages 305–322. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[21] M. I. Kanovich, P. Rowe, and A. Scedrov. Collaborative planning with confidentiality. *J. Autom. Reasoning*, 46(3-4):389–421, 2011.

[22] P. Lincoln, J. C. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. In *FOCS*, volume II, pages 662–671. IEEE, 1990.

[23] D. Miller and E. Pimentel. A formal framework for specifying sequent calculus proof systems. Journal submission, July 2011.

[24] D. Miller and A. Saurin. From proofs to focused proofs: a modular proof of focalization in linear logic. volume 4646, pages 405–419, 2007.

[25] M. Minsky. Recursive unsolvability of post's problem of 'tag' and other topics in the theory of turing machines. *Annals of Mathematics*, 1961.

[26] V. Nigam. *Exploiting non-canonicity in the Sequent Calculus*. PhD thesis, Ecole Polytechnique, Sept. 2009.

[27] V. Nigam. On the complexity of linear authorization logics. In *LICS*, pages 511–520. IEEE, 2012.

[28] V. Nigam and D. Miller. Algorithmic specifications in linear logic with subexponentials. pages 129–140, 2009.

[29] V. Nigam, E. Pimentel, and G. Reis. Specifying proof systems in linear logic with subexponentials.

[30] V. Nigam, E. Pimentel, and G. Reis. TATU. `http://www.logic.at/people/giselle/tatu/`.

[31] W. J. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[32] P. Schroeder-Heister. Rules of definitional reflection. In M. Vardi, editor, *Eighth Annual Symposium on Logic in Computer Science*, pages 222–232. IEEE Computer Society Press, IEEE, June 1993.