First Order Linear Logic Without Modalities is NEXPTIME-Hard

Patrick Lincoln* Andre Scedrov†

Abstract

The decision problem is studied for the nonmodal or multiplicative-additive fragment of first order linear logic. This fragment is shown to be NEXPTIME*-hard. The hardness proof combines Shapiro's logic programming simulation of nondeterministic Turing machines with the standard proof of the* PSPACE*-hardness of quantified boolean formula validity, utilizing some of the surprisingly powerful and expressive machinery of linear logic.*

1 Introduction

Linear logic, introduced by Girard, is a resource-sensitive refinement of classical logic [10, 29]. Linear logic gains its expressive power by restricting the "structural" proof rules of contraction (copying) and weakening (erasing). The contraction rule makes it possible to reuse any stated assumption as often as desired. The weakening rule makes it possible to use dummy assumptions, *i.e.*, it allows a deduction to be carried out without using all of the hypotheses. Because contraction and weakening together make it possible to use an assumption as often or as little as desired, these rules are responsible for what one may see in hindsight as a loss of control over resources in classical (and intuitionistic) logic. Without contraction or weakening, as in linear logic, propositions may be thought of as process states, events, or resources, which must be carefully accounted for. One may then

*`lincoln@cs.stanford.edu` Computer Science Laboratory, SRI International, Menlo Park, CA 94025-3493. Initial results obtained during the visit to University of Pennsylvania in February 1992, supported by the Institute of Research in Cognitive Science.

†`andre@cis.upenn.edu` Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104-6395. Partially supported by NSF Grant CCR-91-02753 and by ONR Grants N00014-88-K-0635 and N00014-92-J-1916.

isolate two distinct forms of conjunction and disjunction, one form called "multiplicative", and the other "additive". Viewing the hypotheses as resources, a proof of a multiplicative conjunction as a conclusion forbids any sharing between the resources used to establish each conjunct, whereas the additive conjunction requires the sharing of all of the resources. Full linear logic also involves a kind of modality: a "storage" or "reuse" operator, !. Intuitively, the hypothesis $!A$ provides unlimited reuse of the resource A. The usual function type $A \Rightarrow B$ can be recovered as $!A \multimap B$, where $\multimap$ is *linear implication*, which provides the type of functions that "use" their argument exactly once.

Basic constructions on vector spaces provide a first, naive interpretation of linear logic connectives, much as the basic operations on sets provide an interpretation of usual logical connectives. For instance, multiplicative conjunction is interpreted as tensor product. More subtle points of linear logic originate in so-called coherence spaces, which maintain a notion of finite basis and isomorphism with the double dual without imposing isomorphism with the dual. Similar phenomena have been observed in certain natural set-theoretic operations on infinite games [6, 7] and in so-called -autonomous categories [5]. Event spaces, which come about from semantics of concurrency, provide another semantic framework for linear logic [23]. A mathematical setting for dynamic aspects of cut elimination is provided by proof nets [10, 9] and in a deeper sense by Girard's geometry of interaction [11, 12, 8, 24, 1, 13, 14].

Among the computer science ramifications of linear logic, the methods of this paper are closest to what may be broadly called logic programming ramifications, in which computation is expressed by cut-free proof search in certain linear logic theories. Recent topics in this direction include a treatment of object-style inheritance and linear logic programming [3, 4, 2, 15], a treatment of concurrent constraint programming [25], and a closely related treatment of Milner's π-calculus [22].

The remarkable expressiveness of cut-free linear logic proof search as a computational paradigm is also indicated by the complexity and undecidability of provability in fragments of linear logic. These results are consequences of direct, lockstep simulations of computations on generic machines by cut-free proof search in fragments of linear logic. Provability in full propositional (*i.e.*, quantifier-free) linear logic is undecidable [18]. However, even the fragment of linear logic that does not allow modalities is strikingly expressive: provability of multiplicative propositional formulas is NP-complete [17]. In fact, the decision problem for constant-only formulas in multiplica-

tive propositional linear logic is also NP-complete [20]. If the additives are also allowed, provability for propositional formulas is PSPACE-complete [18], and in fact this fragment allows a structural embedding of a cut-free proof system for implicational propositional intuitionistic logic [19]. The decidability of second order propositional linear logic without modalities is an open problem.

In the remainder of the paper we shall restrict our attention to the nonmodal, or multiplicative-additive linear logic with first order quantifiers and function symbols, MALL1. Because the depth of a cut-free proof tree is still linear in the size of the conclusion, as in the case of the propositional multiplicative-additive fragment, it can be seen that without function symbols, MALL1 is still PSPACE-complete. However, in the presence of function symbols the complexity is different in spite of the same bound on the depth. It is shown that provability in MALL1 is NEXPTIME-hard (Theorem 5.4).

The hardness result is achieved through a direct encoding of Turing machine transitions, which are shared via the additives, and the existential quantification over intermediate Turing machine configurations. The linear depth bound on cut-free MALL1 proofs gives an immediate single exponential upper bound on the number of Turing machine transitions that can be used in the entire proof using this encoding. In contrast to previous work on linear logic proof search as a computational paradigm (see above), in which a computation proceeds "upwards" along a proof tree, here the computation steps are applied "horizontally across" the leaves of a proof tree.

It is likely that the membership of MALL1 in NEXPTIME, and hence its NEXPTIME-completeness, may be shown by a decision procedure based on dynamic skolemization. This technique, developed for theorem proving in intuitionistic logic [26], falls outside of the scope of this paper.

The precise descriptions of MALL1 and of the particular version of nondeterministic (single) exponential time Turing machines are given in Section 2. An encoding of nondeterministic exponential time Turing machines by MALL1 formulas is given in Section 3. This encoding is reminiscent of the standard proof of the PSPACE-hardness of quantified boolean formula validity [28, 16]. The encoding is also related to the logic programming simulation of Turing machines given in [27]. In Section 4 it is shown that whenever a nondeterministic exponential time Turing machine accepts, then the MALL1 formula encoding the machine is provable. The converse is shown in Section 5.

We would like to thank Yuri Gurevich for suggesting that the logic programming simulation of Turing machines given in [27] may be a source of

lower bounds for the complexity of MALL1. This suggestion eventually led us to formulate the encoding given in Section 3. We would also like to thank Jean-Marc Andreoli, Jawahar Chirimar, Jean Gallier, Jean-Yves Girard, Carl Gunter, Joshua Hodas, Jim Lipton, Dale Miller, John Mitchell, Jon Riecke, Vijay Saraswat, Phil Scott, Natarajan Shankar, and Tim Winkler for interesting and inspiring discussions.

2 Preliminaries

Let us begin by specifying the logical framework and the particular version of nondeterministic Turing machines considered in this paper.

2.1 Logical Framework

Gentzen-style sequent calculus is the formal logical framework throughout the paper. For our purposes it is convenient to consider sequents of the form $\Gamma \vdash \Delta$ where Γ, Δ are finite multisets of formulas. Note that in standard presentations of sequent calculi, sequents are often built from sets of formulas, where we use multisets here. This difference is crucial. We assume a set of function symbols and predicate symbols, along with their associated arities. Atomic formulas (other than constants $1, -, \top, 0$) are of the form $P(t_1, \ldots, t_n)$ where $t_1, \ldots, t_n$ are terms and P is a predicate symbol of arity n, as usual in first order predicate calculus. Note that there is no equality symbol. The inference rules for the MALL1 sequent calculus are given in Figure 1.

The following notational conventions are observed throughout this paper:

H	Positive atomic formula $P(t_1, \ldots, t_n)$
$H^\perp$	Negated atomic formula $P(t_1, \ldots, t_n)^\perp$
A, B, C	Arbitrary formulas
$\Gamma, \Delta, \Sigma, \Theta, \Xi$	Arbitrary finite multisets of formulas

In the $\forall\mathbf{R}$ and $\exists\mathbf{L}$ rules in Figure 1 it is assumed that X is not free in Γ, Δ. The English names for the rules given in Figure 1 are identities, tensor, linear implication, plus, with, bottom, one, zero, top, universal, existential, and cut, respectively. $\otimes$, $\multimap$, and $^\perp$ are *multiplicative* connectives; 1 and $-$ are *multiplicative* propositional constants. $\oplus$ and & are *additive* connectives; 0 and $\top$ are *additive* propositional constants. Observe the "additive" nature of the quantifiers. For notational convenience, it is often assumed that $\multimap$ and $\otimes$ associate to the right, and that $\otimes$ has higher precedence than $\multimap$.

$$\mathbf{I+}\quad \frac{}{H \vdash H} \qquad \frac{}{H^\perp \vdash H^\perp}\quad \mathbf{I\text{-}}$$

$$\mathbf{IL}\quad \frac{}{H, H^\perp \vdash} \qquad \frac{}{\vdash H, H^\perp}\quad \mathbf{IR}$$

$$\otimes\mathbf{L}\quad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, (A \otimes B) \vdash \Delta} \qquad \frac{\Sigma \vdash A, \Theta \qquad \Gamma \vdash B, \Delta}{\Sigma, \Gamma \vdash (A \otimes B), \Theta, \Delta}\quad \otimes\mathbf{R}$$

$$-\!\circ\mathbf{L}\quad \frac{\Sigma \vdash A, \Theta \qquad \Gamma, B \vdash \Delta}{\Sigma, \Gamma, (A -\!\circ B) \vdash \Theta, \Delta} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash (A -\!\circ B), \Delta}\quad -\!\circ\mathbf{R}$$

$$\oplus\mathbf{L}\quad \frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, (A \oplus B) \vdash \Delta} \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash (A \,\&\, B), \Delta}\quad \mathbf{\&\ R}$$

$$\mathbf{\&\ L1}\quad \frac{\Gamma, A \vdash \Delta}{\Gamma, (A \,\&\, B) \vdash \Delta} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash (A \oplus B), \Delta}\quad \oplus\mathbf{R1}$$

$$\mathbf{\&\ L2}\quad \frac{\Gamma, B \vdash \Delta}{\Gamma, (A \,\&\, B) \vdash \Delta} \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash (A \oplus B), \Delta}\quad \oplus\mathbf{R2}$$

$$-\mathbf{L}\quad \frac{}{- \vdash} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash -, \Delta}\quad -\mathbf{R}$$

$$1\mathbf{L}\quad \frac{\Gamma \vdash \Delta}{\Gamma, 1 \vdash \Delta} \qquad \frac{}{\vdash 1}\quad 1\mathbf{R}$$

$$0\mathbf{L}\quad \frac{}{\Gamma, 0 \vdash \Delta} \qquad \frac{}{\Gamma \vdash \top, \Delta}\quad \top\mathbf{R}$$

$$\forall\mathbf{L}\quad \frac{\Gamma, A[t/X] \vdash \Delta}{\Gamma, \forall X.A \vdash \Delta} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall X.A, \Delta}\quad \forall\mathbf{R}$$

$$\exists\mathbf{L}\quad \frac{\Gamma, A \vdash \Delta}{\Gamma, \exists X.A \vdash \Delta} \qquad \frac{\Gamma \vdash A[t/X], \Delta}{\Gamma \vdash \exists X.A, \Delta}\quad \exists\mathbf{R}$$

$$\frac{\Sigma \vdash A, \Theta \qquad A, \Gamma \vdash \Delta}{\Sigma, \Gamma \vdash \Theta, \Delta}\quad \mathbf{Cut}$$

Figure 1: Rules for MALL1

An *analysis* is the process of applying a rule to a sequent matching the conclusion of the rule in order to generate the corresponding premises. The *principal formula* of the rule is then said to be *analyzed* by the reduction. A MALL1 proof is represented as a tree rooted at its conclusion sequent at the bottom and with the leaves at the top. Given this orientation, the notion of a rule occurring above or below another rule should be clear.

The reader will note that linear negation is a defined concept on composite formulas, not a basic connective. One may define negation by recursion on the structure of formulas: $(H^\perp)^\perp$ is H, $(A \otimes B)^\perp$ is $A \multimap B^\perp$, $(A \multimap B)^\perp$ is $A \otimes B^\perp$, $(A \,\&\, B)^\perp$ is $A^\perp \oplus B^\perp$, $(A \oplus B)^\perp$ is $A^\perp \,\&\, B^\perp$, $(\forall X.A)^\perp$ is $\exists X.A^\perp$, $(\exists X.A)^\perp$ is $\forall X.A^\perp$, $1^\perp$ is $-$, $-^\perp$ is 1, $\top^\perp$ is 0, and $0^\perp$ is $\top$. If Γ consists of $A_1, \ldots, A_n$, then $\Gamma^\perp$ denotes $A_1^\perp, \ldots, A_n^\perp$.

Although the identity rules (*i.e.,* the first four rules in Figure 1) are restricted to atomic formulas H, the analogous identity rules for arbitrary formula A instead of H are derivable. Negation rules are derivable as well, namely from $\Gamma, A \vdash \Delta$ one can infer $\Gamma \vdash A^\perp, \Delta$, and from $\Gamma \vdash A, \Delta$ one can infer $\Gamma, A^\perp \vdash \Delta$. The reader will also observe that the linear logic connective *par* may be defined by letting $A \wp B$ be $A^\perp \multimap B$ and that the *par* rule [10] is derivable. In fact, our sequent calculus presentation is equivalent to the first order sequent calculus in [10] without exponentials $!, \Gamma$. More precisely, a sequent $\Gamma \vdash \Delta$ is provable in MALL1 (without cut) iff the sequent $\vdash \Gamma^\perp, \Delta$ is provable (without cut, respectively) in the first order sequent calculus in [10], excluding exponentials. In particular, cut-elimination for MALL1 follows from the cut-elimination shown in [10], see also [18]. For the record:

Theorem 2.1 *If a sequent is provable in* MALL1*, then it is provable in* MALL1 *without using the* **Cut** *rule.*

We end this section by considering some technical properties of MALL1, in particular so-called *permutability* properties, some of which have been discussed in several texts [2, 21].

The *sign* of a formula is defined using the transformations below:

$$\begin{array}{rcl}
[A \multimap B]^+ & = & [A]^\perp \multimap [B]^+ \\
[A \otimes B]^+ & = & [A]^+ \otimes [B]^+ \\
[\Sigma, A]^+ & = & [\Sigma]^+, [A]^+ \\
[A^\perp]^+ & = & ([A]^\perp)^\perp \\
[A \multimap B]^\perp & = & [A]^+ \multimap [B]^\perp \\
[A \otimes B]^\perp & = & [A]^\perp \otimes [B]^\perp \\
[\Sigma, A]^\perp & = & [\Sigma]^\perp, [A]^\perp \\
[A^\perp]^\perp & = & ([A]^+)^\perp
\end{array}$$

The sign of an instance of a formula A in a sequent $\Sigma \vdash \Delta$ is given by the superscript on A in $[\Sigma]^\perp$ or $[\Delta]^+$. That is, if an instance of formula A ends up as $[A]^+$, then it is *positive*. If an instance of formula A ends up as $[A]^\perp$, then it is *negative*.

A sequent is said to be *balanced* if it has the same number of positive and negative occurrences of atoms other than constants. Otherwise, a sequent is said to be *unbalanced*. A sequent is said to be *multiplicative* if it contains only formulas built of atoms other than $0, \top$, of their negations, or of connectives $\otimes$ or $\multimap$. It is linear logic folklore that all provable multiplicative sequents are balanced, and therefore no unbalanced multiplicative sequents can be provable. This property is readily shown by induction on cut-free proofs.

Proposition 2.2 *A multiplicative sequent is provable only if it is balanced.*

Note that this property fails for full MALL1 and other fragments of linear logic that include the additive connectives and constants.

The following properties all follow by induction on the height of cut-free proof. These are *permutability* properties, as they ensure that if a proof exists, then a proof of a certain permuted form exists. That is, one may permute the order of application of inferences in a proof to achieve a kind of normal form. These properties are used to give control over the shape of linear logic proofs, thus enabling a more direct computational reading.

Proposition 2.3 *If a sequent* $\Gamma, A \oplus B \vdash \Delta$ *is provable in* MALL1*, then so are* $\Gamma, A \vdash \Delta$ *and* $\Gamma, B \vdash \Delta$.

Proposition 2.4 *If a sequent* $\Gamma \vdash A \& B, \Delta$ *is provable in* MALL1*, then so are* $\Gamma \vdash A, \Delta$ *and* $\Gamma \vdash B, \Delta$.

Proposition 2.5 *If a sequent* $\Gamma, A \otimes B \vdash \Delta$ *is provable in* MALL1*, then so is* $\Gamma, A, B \vdash \Delta$.

Proposition 2.6 *If a sequent* $\Gamma \vdash A \multimap B, \Delta$ *is provable in* MALL1*, then so is* $\Gamma, A \vdash B, \Delta$.

Proposition 2.7 *If a sequent* $\Gamma, \exists X.A \vdash \Delta$ *is provable in* MALL1*, then so is* $\Gamma, A[t/X] \vdash \Delta$ *for any term* t.

Proposition 2.8 *If a sequent* $\Gamma \vdash \forall X.A, \Delta$ *is provable in* MALL1*, then so is* $\Gamma \vdash A[t/X], \Delta$ *for any term* t.

Proposition 2.9 *If a sequent* $\Gamma, A \& B \vdash \Delta$ *is provable in* MALL1*, where* Γ, Δ *contain only negative occurrences of* $\&$ *and positive occurrences of* $\oplus$*, then* $\Gamma, A \vdash \Delta$ *or* $\Gamma, B \vdash \Delta$ *is provable in* MALL1.

Proposition 2.10 *If a sequent* $\Gamma \vdash A \oplus B, \Delta$ *is provable in* MALL1*, where* Γ, Δ *contain only negative occurrences of* $\&$ *and positive occurrences of* $\oplus$*, then* $\Gamma \vdash A, \Delta$ *or* $\Gamma \vdash B, \Delta$ *is provable in* MALL1.

Proposition 2.11 *If a sequent* $\Gamma, \forall X.A \vdash \Delta$ *is provable in* MALL1*, where* Γ, Δ *contain only positive occurrences of* $\oplus$ *and* $\exists$ *and only negative occurrences of* $\&$ *and* $\forall$*, then for some term* t*,* $\Gamma, A[t/X] \vdash \Delta$ *is provable in* MALL1.

Proposition 2.12 *If a sequent* $\Gamma \vdash \exists X.A, \Delta$ *is provable in* MALL1*, where* Γ, Δ *contain only positive occurrences of* $\oplus$ *and* $\exists$ *and only negative occurrences of* $\&$ *and* $\forall$*, then for some term* t*,* $\Gamma \vdash A[t/X], \Delta$ *is provable in* MALL1.

2.2 NEXPTIME Turing Machines

For definiteness, let us agree that "Turing machine" means the following kind of nondeterministic machine. These machines have one tape and no transitions are applicable in the final state. More precisely, such a machine is given by the tuple $\langle Q, \mathcal{S}, \mathcal{T}, \delta, q_0, \flat, q_F \rangle$, where:

- Q is a finite set of states,
- $q_0 \in Q$ is the (unique) initial state,
- $q_F \in Q$ is the (unique) final state,

- $\mathcal{T}$ is a finite set of allowable tape symbols, disjoint from Q,
- $\mathcal{S} \subset \mathcal{T}$ is a finite set of input symbols,
- $\flat \in \mathcal{T} \backslash \mathcal{S}$ is the blank symbol,
- δ is the next move relation, a relation from $(Q \setminus \{q_F\}) \times \mathcal{T}$ to $Q \times \mathcal{T} \times \{\mathit{Left}, \mathit{Right}\}$,

For instance, if δ contains move instructions $q_5, c \mapsto q_6, d, \mathit{Left}$ and $q_5, c \mapsto q_3, d, \mathit{Right}$, then one may informally say that the Turing machine M in state q_5, where the tape symbol c is currently under the read/write head, can make one of the following one-step moves: either it erases c, writes d, moves one square to the left, and passes to state q_6, or it erases c, writes d, moves one square to the right, and passes to state q_3.

A *configuration* or *instantaneous description* of a Turing machine, or briefly ID, is given by a quadruple $\langle \alpha_1, q, c, \alpha_2 \rangle$, where $q \in Q$ is the current state, and $\alpha_1, \alpha_2 \in \mathcal{T}^*$ are the used part of the tape to the left and to the right of the Turing machine's read/write head, respectively, and c is the symbol directly underneath the read/write head. Here and throughout the paper "the used portion of the tape" means the smallest contiguous portion of the tape that includes all tape cells with a symbol other than $\flat$ and all tape cells visited by the read/write head.

The machine is given input $t \in \mathcal{S}^*$ by writing $\flat t$ on the otherwise empty tape, one symbol per tape cell, with the read/write head scanning the first symbol before the beginning of t, and the state is q_0, forming the ID $\langle \epsilon, q_0, \flat, t \rangle$. M is said to *accept* input t in exactly n steps, $n \geq 1$, if there exists a sequence of configurations $v_0, \ldots, v_n$, each obtained from the previous by a one-step move, where v_0 is the initial configuration just described and $v_n = \langle \alpha_1, q_F, \flat, \alpha_2 \rangle$. Note that the state of v_i cannot be q_F whenever $0 \leq i < n$. M is said to *accept* input t if there exists $n \geq 1$ such that M accepts t in exactly n steps. Let $\mathcal{L}(M)$, *the language accepted by* M, consist of all $t \in \mathcal{S}^*$ accepted by M.

A Turing machine M is said to be *exponential time* if there exists a polynomial $p(n)$ with nonnegative integer coefficients such that $M \in \text{NTIME}(2^{p(n)})$, that is, for each $t \in \mathcal{L}(M)$, M accepts t in at most $2^{p(n)}$ steps, where n is the length of t. Let NEXPTIME be the class of languages of the form $\mathcal{L}(M)$ for some exponential time Turing machine M [28].

3 Encoding NEXPTIME in MALL1

In this section we give an efficient encoding of any nondeterministic Turing machine and any input by a MALL1 sequent, which also depends on a natural number n. In Sections 4 and 5 it will be shown that the MALL1 sequent is provable if and only if the Turing machine accepts the input in at most 2^n steps.

We translate Turing machine instructions as follows, similarly to the logic programming simulation given in [27]. A left move Turing machine instruction is translated as:

$$\begin{aligned}[q_i, a \mapsto q_j, d, \mathit{Left}] \quad = \quad & \&_{c \in \mathcal{T}} \forall L \forall R\, id(\langle [c|L], q_i, a, R\rangle) \multimap id(\langle L, q_j, c, [d|R]\rangle) \\ & \& \ \forall R\, id(\langle \epsilon, q_i, a, R\rangle) \multimap id(\langle \epsilon, q_j, \flat, [d|R]\rangle),\end{aligned}$$

where the first conjunct describes the behavior at the center and at the right end of the used portion of the tape, while the second conjunct describes the behavior at the left end of the used portion of the tape and the behavior if the tape has not been used yet.

A right move Turing machine instruction is translated as:

$$\begin{aligned}[q_i, a \mapsto q_j, d, \mathit{Right}] \quad = \quad & \&_{c \in \mathcal{T}} \forall L \forall R\, id(\langle L, q_i, a, [c|R]\rangle) \multimap id(\langle [d|L], q_j, c, R\rangle) \\ & \& \ \forall L\, id(\langle L, q_i, a, \epsilon\rangle) \multimap id(\langle [d|L], q_j, \flat, \epsilon\rangle),\end{aligned}$$

where the first conjunct describes the behavior at the center and at the left end of the used portion of the tape, while the second conjunct describes the behavior at the right end of the used portion of the tape and the behavior if the tape has not been used yet.

Given a Turing machine program with m instructions $\delta = \tau_1, \tau_2, \cdots, \tau_m$, we construct the following formula:

$$[\delta] = 1 \,\&\, [\tau_1] \,\&\, [\tau_2] \,\&\, \cdots \,\&\, [\tau_m].$$

In order to encode an exponential number of steps of execution we use a modification of the standard proof of the PSPACE-hardness of quantified boolean formula validity [16]. We develop an encoding function M, where $M_k(V, V')$ is meant to denote that from ID V, one can reach ID V' in at most 2^k steps. A first attempt would be to use the encoding:

$$M_k(V, V') = \exists W. M_{k\perp 1}(V, W) \ \mathit{and}\ M_{k\perp 1}(W, V').$$

However, since $M_{k\perp 1}$ appears twice in the encoding of M_k, this encoding is itself exponential. Instead, let us use the following polynomial encoding, which achieves the same aim. Here and throughout P is a fixed unary predicate symbol distinct from id and $A \to B$ abbreviates $1\&(A \multimap B)$.

$$\begin{aligned} M_0(V,V') &= P(V) \multimap P(V'), \\ M_k(V,V') &= \exists W.\forall Y.\forall Z. \left[\left(\begin{array}{c} (P(Y) \to P(V)) \otimes (id(W) \to P(Z)) \\ \oplus \\ (P(Y) \to id(W)) \otimes (P(V') \to P(Z)) \end{array} \right) \multimap M_{k\perp 1}(Y,Z) \right]. \end{aligned}$$

In this encoding, $M_{k\perp 1}$ only appears once in the encoding of M_k, thus enabling the encoding of an exponential number of steps of computation in a polynomially-sized formula.

Finally, we define the MALL1 encoding of a Turing machine accepting an input in at most 2^n steps as follows. Let end be a fresh propositional symbol. Given a number n, a Turing machine $M = \langle Q, \mathcal{S}, \mathcal{T}, \delta, q_0, \flat, q_F \rangle$, and an input t, let $v_0 = \langle \epsilon, q_0, \flat, t \rangle$ and define the encoding $[M(t)]_n$ as the MALL1 sequent:

$$[\delta], \forall L\, \forall R\, (((P(X) \to id(v_0)) \multimap ((id(\langle L, q_F, \flat, R \rangle) \to P(X')) \multimap M_n(X,X'))) \multimap end) \vdash end.$$

Let k be the length of t. Observe that the sequent $[M(t)]_n$, fully written out, has length polynomial in n, k.

Discussion: Although the complexity class NEXPTIME is defined to be the union of NTIME$(2^{p(n)})$ taken over all polynomials p and the above encoding seemingly captures only NTIME(2^n), it will nevertheless follow that MALL1 is NEXPTIME-hard once the soundness and faithfulness of the encoding has been established. Indeed, a polynomial $p(n)$ may be fixed for our translation by choosing a particular NEXPTIME-complete problem whose decision algorithm M_0 is in NTIME$(2^{p(n)})$. Hence it suffices to consider the encoding $[M_0(t)]_{p(n)}$ on inputs t of size n.

3.1 Encoding logic programs in MALL1

It follows from [27] that nondeterministic Turing machines can be simulated by logic programs with at most one formula in the body of each program clause, *i.e.,* logic programs in which each program clause has one of the

following two forms, where A, A' are atomic formulas:

$$\begin{array}{lcl} A & \leftarrow & A' \\ A & \leftarrow & \end{array}$$

so that the depth complexity of the logic program is the same as time complexity of the nondeterministic Turing machine. Such logic programs can be encoded in MALL1 by a straightforward modification of the above encoding of nondeterministic Turing machines, namely:

$$\begin{array}{lcl} [A_i(\vec{f}(\vec{X})) \leftarrow A_j(\vec{h}(\vec{X}))] & = & \forall \vec{X}\, id(\langle g_i, \vec{f}(\vec{X})\rangle) \multimap id(\langle g_j, \vec{h}(\vec{X})\rangle), \\ [A_i(\vec{f}(\vec{X})) \leftarrow\ \ ] & = & \forall \vec{X}\, id(\langle q_i, \vec{f}(\vec{X})\rangle) \multimap id(\langle g_E, \epsilon\rangle), \end{array}$$

where constant g_E is distinct from the constants g_i, g_j , $\vec{f} = f_1, \ldots f_k$, $\vec{g} = g_1, \ldots g_k$, $\vec{X} = X_1, \ldots X_k$, with k large enough, since a logic program consists of finitely many clauses. It is easy to see that the soundness and faithfulness of this encoding may be shown by a straightforward adaptation of the arguments given in Sections 4 and 5.

4 Soundness of the Encoding

In this section it is shown that the encoding given in Section 3 is sound, *i.e.,* if a Turing machine M accepts an input t in at most 2^n steps, then the sequent $[M(t)]_n$ is provable in MALL1. The main technical point is given by

Lemma 4.1 *If a Turing machine with program δ can pass from a* ID *v to a* ID *v' in at most 2^n steps, then for any natural numbers j, k and any natural numbers $i \leq j$ and $m \leq k$, the sequent*

$$[\delta],\quad \begin{array}{c} P(Y_j) \rightarrow P(Y_{j\perp 1}), \\ \cdots, \\ P(Y_i) \rightarrow id(v), \\ id(v') \rightarrow P(Z_m), \\ \cdots, \\ P(Z_{k\perp 1}) \rightarrow P(Z_k) \end{array} \quad \vdash M_n(Y_j, Z_k)$$

is provable in MALL1.

Proof. Let Γ consist of $P(Y_j) \to P(Y_{j\perp 1}), \ldots, P(Y_i) \to id(v)$ and let Δ consist of $id(v') \to P(Z_m), \ldots, P(Z_{k\perp 1}) \to P(Z_k)$. The argument is by induction on n. If $n = 0$, then $M_0(Y_j, Z_k) = P(Y_j) \multimap P(Z_k)$. There are two cases: either the Turing machine idles or it makes one step. If the machine makes one step from v to v' by applying an instruction $\tau \in \delta$, then $id(v) \multimap id(v')$ is an instance of a conjunct C in $[\tau]$ and thus a MALL1 proof of the sequent

$$C, \Gamma, \Delta, P(Y_j) \vdash P(Z_k)$$

is readily obtained by several instances of $\mathbf{I}+$, $\multimap\ \mathbf{L}$, and by $\forall\ \mathbf{L}$. Then a required MALL1 proof follows by $\multimap\ \mathbf{R}$ and several instances of $1\mathbf{L}$ and $\&\ \mathbf{L}$. If the machine idles, then v is the same as v' and the argument is similar by using 1 instead of C.

Assume that the statement of the lemma holds for n. If the machine passes from ID v to ID v' in at most 2^{n+1} steps, let w be an ID such that the machine passes from v to w in at most 2^n steps and from w to v' also in at most 2^n steps. Then let π be a MALL1 proof of the sequent

$$[\delta], \Gamma, P(Y_{j+1}) \to P(Y_j), id(w) \to P(Z_{k+1}) \vdash M_n(Y_{j+1}, Z_{k+1})$$

and let π' be a MALL1 proof of the sequent

$$[\delta], \Delta, P(Y_{j+1}) \to id(w), P(Z_k) \to P(Z_{k+1}) \vdash M_n(Y_{j+1}, Z_{k+1})$$

both of which exist by the induction hypothesis. Let ρ be the MALL1 proof of

$$[\delta], \Gamma, \Delta, P(Y_{j+1}) \to P(Y_j), id(w) \to P(Z_{k+1}) \vdash M_n(Y_{j+1}, Z_{k+1})$$

obtained from π by several instances of $1\mathbf{L}$ and $\&\ \mathbf{L}$. Use $\otimes\ \mathbf{L}$ to obtain the MALL1 proof σ of

$$[\delta], \Gamma, \Delta, (P(Y_{j+1}) \to P(Y_j)) \otimes (id(w) \to P(Z_{k+1})) \vdash M_n(Y_{j+1}, Z_{k+1})$$

Let σ' be the MALL1 proof of

$$[\delta], \Gamma, \Delta, (P(Y_{j+1}) \to id(w)) \otimes (P(Z_k) \to P(Z_{k+1})) \vdash M_n(Y_{j+1}, Z_{k+1})$$

obtained similarly from π'. Then the required MALL1 proof of

$$[\delta], \Gamma, \Delta \vdash M_{n+1}(Y_j, Z_k)$$

can be obtained from σ, σ' by $\oplus\ \mathbf{L}$, $\multimap\ \mathbf{R}$, $\forall\ \mathbf{R}$, $\forall\ \mathbf{R}$, and $\exists\ \mathbf{R}$. ∎

The soundness of the encoding now follows:

Theorem 4.2 *For any natural number n, Turing machine M, and input t, if M accepts t in at most 2^n steps, then the sequent $[M(t)]_n$ is provable in* MALL1.

Proof. Let $M = \langle Q, \mathcal{S}, \mathcal{T}, \delta, q_0, \flat, q_F \rangle$, $v_0 = \langle \epsilon, q_0, \flat, t \rangle$ be the initial ID, and let $v' = \langle \ell, q_F, \flat, r \rangle$ be the final ID in an accepting computation of M on t. By Lemma 4.1, let π be a MALL1 proof of

$$[\delta], P(X) \multimap id(v_0), id(v') \multimap P(X') \vdash M_n(X, X')$$

A MALL1 proof of $[M(t)]_n$ may then be constructed from π first by using two instances of $\multimap$ **R**, then $\multimap$ **L** whose other premise is the identity $end \vdash end$, and finally two instances of $\forall$ **L**. ∎

5 Faithfulness of the Encoding

This section is concerned with the faithfulness of the encoding given in Section 3. That is, given a Turing machine M, an input t , a natural number n, and a MALL1 proof of $[M(t)]_n$, then M accepts t in at most 2^n steps.

First let us observe

Proposition 5.1 *For any Turing Machine $M = \langle Q, \mathcal{S}, \mathcal{T}, \delta, q_0, \flat, q_F \rangle$, well-formed Turing machine* ID *v, instruction $\tau \in \delta$, and universally quantified conjunct C in the formula $[\tau]$, if $id(v) \multimap id(v')$ is an instance of C for some term v' , then v' is a well-formed Turing machine* ID *and M can make a transition from* ID *v to* ID *v' in one step.*

Proof. By cases on the instruction τ and its translation $[\tau]$ given in Section 3. ∎

Permutability properties discussed in Section 2 will now be used to show that without loss of generality, a cut-free proof of $[M(t)]_n$ consists only of sub-proofs of sequents of the form $[\delta], \Xi \vdash M_k(X, Y)$, where $k < n$ and Ξ includes exactly two occurences of the predicate symbol id, one negative and one of positive. Informally, one may read such a sequent as asserting that the machine instructions δ can be used to move from the negative occurrence of id in Ξ to the positive occurrence of id in Ξ in at most 2^n steps. This informal reading is made formal below. The key point is that a cut-free proof of such a sequent can be read as a description of a Turing machine computation. Recall that $A \to B$ abbreviates $1 \& (A \multimap B)$.

Lemma 5.2 *Let $M = \langle Q, \mathcal{S}, \mathcal{T}, \delta, q_0, \flat, q_F \rangle$ be a Turing machine and n a natural number. For any well-formed Turing machine* ID *v, for any natural numbers j, k and any natural numbers $i \leq j$ and $m \leq k$, if a sequent*

$$[\delta], \Sigma, \Theta, \quad \begin{array}{c} P(Y_j) \multimap P(Y_{j\perp 1}), \\ \cdots, \\ P(Y_i) \multimap id(v), \\ id(v') \multimap P(Z_m), \\ \cdots, \\ P(Z_{k\perp 1}) \multimap P(Z_k), \end{array} \quad \vdash M_n(Y_j, Z_k)$$

is provable in MALL1 *for some term v', where Σ is a union of disjoint multisets of the form $P(Y_{j'}) \multimap P(Y_{j'\perp 1}), \ldots, P(Y_{i'}) \multimap id(u)$ for some natural numbers $i' \leq j' < i$ and some term u, where Θ is a union of disjoint multisets of the form $id(u') \multimap P(Z_{m'}), \ldots, P(Z_{k'\perp 1}) \multimap P(Z_{k'})$ for some natural numbers $m' \leq k' < m$ and some term u', and where no atomic formula with predicate symbol P occurs as a subformula in two distinct multisets in Σ, Θ, then v' is a well-formed Turing machine* ID*, and M can make a transition from* ID *v to* ID *v' in at most 2^n steps.*

Proof. Let Γ consist of $P(Y_j) \multimap P(Y_{j\perp 1}), \ldots, P(Y_i) \multimap id(v)$ and let Δ consist of $id(v') \multimap P(Z_m), \ldots, P(Z_{k\perp 1}) \multimap P(Z_k)$. Γ, Δ will be called the *active* assumptions, while Σ, Θ will be called the *passive* assumptions. The argument is by induction on n . If $n = 0$, then by Propositions 2.6 and 2.9 we may assume without loss of generality that the bottommost proof rules in the given cut-free proof are, in reverse order, $\multimap \mathbf{R}$ and a string of $\&\mathbf{L}$'s, the latter analyzing $[\delta], \Gamma, \Delta, \Sigma, \Theta$. After this analysis, the provable premise must be of the form:

$$C, \Gamma', \Delta', \Sigma', \Theta', P(Y_j) \vdash P(Z_k),$$

where C is a conjunct in $[\delta]$ and where $\Gamma', \Delta', \Sigma', \Theta'$ consist of linear implications or of 1's. If C is $[\tau]$ for some instruction $\tau \in \delta$, then by Proposition 2.11 we may assume that the next higher rule is $\forall\ \mathbf{L}$ that analyzes C, hence that its provable premise is

$$A, \Gamma', \Delta', \Sigma', \Theta', P(Y_j) \vdash P(Z_k)$$

where A is a linear implication and an instance of C. This sequent is multiplicative and hence by Proposition 2.2 it must be balanced. Because of the conditions on Σ, Θ it follows that Σ', Θ' must consist entirely of 1's,

that Γ', Δ' must consist entirely of linear implications, and that A is in fact $id(v) \multimap id(v')$. In other words, the only way to achieve a balanced sequent is to drop the passive assumptions and to instantiate C as $id(v) \multimap id(v')$. By Proposition 5.1 it is then readily seen that v' is a well-formed machine ID and that M can make a transition from v to v' in one step. By similar reasoning, if A is 1 , then v' must be the same as v and the machine idles.

In the induction step, assume that the statement of the lemma holds for n and consider a cut-free MALL1 proof of

$$A, \Gamma, \Delta, \Sigma, \Theta \vdash M_{n+1}(Y_j, Z_k).$$

By Propositions 2.12, 2.8, 2.6, and 2.3 we may assume without loss of generality that the bottommost rules used are, in reverse order: $\exists$ **R**, $\forall$ **R**, $\forall$ **R**, $\multimap$ **R**, and $\oplus$ **L**. This last step yields two branches, each of which may be assumed by Proposition 2.5 to have $\otimes$ **L** as the bottommost rule. The two branches are cut-free proofs of

$$[\delta], \Gamma, \Delta, \Sigma, \Theta, P(Y_{j+1}) \multimap P(Y_j), id(w) \multimap P(Z_{k+1}) \vdash M_n(Y_{j+1}, Z_{k+1})$$

and

$$[\delta], \Gamma, \Delta, \Sigma, \Theta, P(Y_{j+1}) \multimap id(w), P(Z_k) \multimap P(Z_{k+1}) \vdash M_n(Y_{j+1}, Z_{k+1}).$$

This is the key point of this proof. One reads the given cut-free proof as a description of Turing machine computation from ID v to ID v' by normalizing the given cut-free proof by permutations, finding the two branches above, and reading them as Turing machine computations from ID v to ID w, and ID w to ID v'. The induction hypothesis applies to both branches. In the first branch, the active assumptions are $P(Y_{j+1}) \multimap P(Y_j), \Gamma, id(w) \multimap P(Z_{k+1})$ and the passive assumptions are Δ, Σ, Θ. In the second branch, the active assumptions are $P(Y_{j+1}) \multimap id(w), \Delta, P(Z_k) \multimap P(Z_{k+1})$ and the passive assumptions are Γ, Σ, Θ. Utilizing these two instances of the induction hypothesis, we conclude that w and hence v' are well-formed machine ID's and that M can make a transition from ID v to ID w in at most 2^n steps and also from ID w to ID v' in at most 2^n steps. Therefore M can make a transition from ID v to ID v' in at most $2^n + 2^n = 2^{n+1}$ steps. ∎

Thus in the full proof tree, one reads the complete Turing machine computation across the top of the proof, with all the action happening at applications of $\multimap$ **L** rule. The machinery utilized throughout the remainder of the proof tree exists to create an exponential number of copies of the

instruction set, and provide the glue to hold the computation together. The application of this lemma to the initial ID yields the faithfulness theorem:

Theorem 5.3 *For any natural number n, Turing machine M, and input t, if the sequent $[M(t)]_n$ is provable in* MALL1*, then M accepts t in at most 2^n steps.*

Proof. Let $v_0 = \langle \epsilon, q_0, \flat, t \rangle$ be the initial ID. By Proposition 2.11 we may assume without loss of generality that the two bottommost rules in a cut-free MALL1 proof of

$$[\delta], \forall L\, \forall R\, (((P(X) \to id(v_0)) \multimap ((id(\langle L, q_F, \flat, R\rangle) \to P(X')) \multimap M_n(X, X'))) \multimap end) \vdash end$$

are the instances of $\forall$ **L**. Therefore, we may consider a cut-free MALL1 proof of

$$[\delta], (((P(X) \to id(v_0)) \multimap ((id(\langle \ell, q_F, \flat, r\rangle) \to P(X')) \multimap M_n(X, X'))) \multimap end) \vdash end$$

for some terms ℓ, r. It may be readily seen that the analysis of $[\delta]$ in this cut-free proof may always be postponed above the instance of the $\multimap$ **L** rule that analyzes the formula

$$((P(X) \to id(v_0)) \multimap ((id(\langle \ell, q_F, \flat, r\rangle) \to P(X')) \multimap M_n(X, X'))) \multimap end.$$

Because $end, 0, \top, -$ do not occur in $[\delta], M_n(X, X')$, the left premise of this instance of $\multimap$ **L** may not contain end, and thus we may consider a cut-free MALL1 proof of

$$[\delta] \vdash (P(X) \to id(v_0)) \multimap ((id(\langle \ell, q_F, \flat, r\rangle) \to P(X')) \multimap M_n(X, X')).$$

By two applications of Proposition 2.6, we may consider a cut-free MALL1 proof of

$$[\delta], P(X) \to id(v_0), id(\langle \ell, q_F, \flat, r\rangle) \to P(X') \vdash M_n(X, X').$$

Lemma 5.2 now applies with Σ, Θ empty. Hence $\langle \ell, q_F, \flat, r\rangle$ is a well-formed ID, in fact the final ID, and M can reach it from v_0 in at most 2^n steps. ∎

Theorems 4.2 and 5.3 yield:

Theorem 5.4 *Provability in* MALL1 *is* NEXPTIME*-hard.*

References

[1] S. Abramsky and R. Jagadeesan. New foundations for the geometry of interaction. In *Proc. 7-th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California*, pages 211–222. IEEE Computer Society Press, Los Alamitos, California, June 1992.

[2] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 1992.

[3] J.-M. Andreoli and R. Pareschi. Logic programming with sequent systems: a linear logic approach. In *Proc. Workshop on Extensions of Logic Programming, Tuebingen*. Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, 1990.

[4] J.-M. Andreoli and R. Pareschi. Linear objects: Logical processes with built-in inheritance. *New Generation Computing*, 9, 1991.

[5] M. Barr. **-Autonomous Categories*, volume 752 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1979.

[6] A. Blass. Degrees of indeterminacy of games. *Fundamenta Mathematicae*, 77:151–166, 1972.

[7] A. Blass. A game semantics for linear logic. *Annals Pure Appl. Logic*, 56:183–220, 1992. Special Volume dedicated to the memory of John Myhill.

[8] V. Danos. La Logique Linéaire Appliquée à l'Étude de Divers Processus de Normalisation at Principalement du Lambda-Calcul. Thèse de Doctorat, Université de Paris VII, 1990.

[9] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

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

[11] J.-Y. Girard. Geometry of interaction I: Interpretation of system F. In *Logic Colloquium '88*, Amsterdam, 1989. North-Holland.

[12] J.-Y. Girard. Geometry of interaction II: Deadlock-free algorithms. In: Springer LNCS 417, 1990.

[13] G. Gonthier, M. Abadi, and J.-J. Levy. The geometry of optimal lambda reduction. In *Proc. 19-th Annual ACM Symposium on Principles of Programming Languages, Albuquerque, New Mexico*. ACM Press, New York, NY, January 1992.

[14] G. Gonthier, M. Abadi, and J.-J. Levy. Linear logic without boxes. In *Proc. 7-th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California*, pages 223–234. IEEE Computer Society Press, Los Alamitos, California, June 1992.

[15] J.S. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. In *Proc. 6-th Annual IEEE Symposium on Logic in Computer Science, Amsterdam*, pages 32–42. IEEE Computer Society Press, Los Alamitos, California, July 1991. Full paper to appear in *Information and Computation*.

[16] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, 1979.

[17] M. Kanovich. Horn programming in linear logic is NP-complete. In *Proc. 7-th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California*, pages 200–210. IEEE Computer Society Press, Los Alamitos, California, June 1992.

[18] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Annals Pure Appl. Logic*, 56:239–311, 1992. Special Volume dedicated to the memory of John Myhill.

[19] P. Lincoln, A. Scedrov, and N. Shankar. Linearizing intuitionistic implication. In *Proc. 6-th Annual IEEE Symposium on Logic in Computer Science, Amsterdam*, pages 51–62. IEEE Computer Society Press, Los Alamitos, California, July 1991. Full paper to appear in *Annals of Pure and Applied Logic*.

[20] P. Lincoln and T. Winkler. Constant-Only Multiplicative Linear Logic is NP-Complete. Manuscript to be submitted to *Proc. MFPS 8*, Oxford, 1992.

[21] P.D. Lincoln. *Computational Aspects of Linear Logic*. PhD thesis, Stanford University, 1992.

[22] D. Miller. The π-calculus as a theory in linear logic: Preliminary results. Technical Report MS-CIS-92-48, Computer Science Department, University of Pennsylvania, June 1992. Submitted. Available using anonymous ftp from host ftp.cis.upenn.edu and the file pub/papers/miller/pic.dvi.Z.

[23] V.R. Pratt. Event spaces and their linear logic. In *AMAST '91: Algebraic Methodology and Software Technology, Iowa City, 1991*, Workshops in Computing, pages 1–23. Springer-Verlag, 1992.

[24] L. Regnier. λ-calcul et Réseaux. Thèse de Doctorat, Université de Paris VII, 1992.

[25] V. Saraswat and P. Lincoln. Higher-order, linear, concurrent constraint programming. Draft, January 1993.

[26] N. Shankar. Proof search in the intuitionistic sequent calculus. In *Proc. of the 11th International Conference on Automated Deduction LNAI 607*, pages 522–536, June 1992.

[27] E.Y. Shapiro. Alternation and the computational complexity of logic programs. *Journal of Logic Programming*, 1:19–33, 1984.

[28] L. Stockmeyer. Classifying the computational complexity of problems. *Journal of Symbolic Logic*, 52:1–43, 1987.

[29] A.S. Troelstra. *Lectures on Linear Logic*. CSLI Lecture Notes No. 29. Center for the Study of Language and Information, Stanford University, 1992.