First-order linear logic without modalities is NEXPTIME-hard

P. Lincoln*

Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA

A. Scedrov**

Department of Mathematics, University of Pennsylvania, 209 South 33rd Street, Philadelphia, PA 19104-6395, USA

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 [3, 20]. 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

*Initial results obtained during the visit to University of Pennsylvania in February 1992, supported by the Institute for Research in Cognitive Science.


0304-3975/94/$07.00 © 1994 – Elsevier Science B.V. All rights reserved

SSDI 0304-3975(94)00107-T
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 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 → B can be recovered as !A → B, where → is linear implication, which provides the type of functions that "use" their argument exactly once. An overview of linear logic may be found in [15].

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 [16]. Recent topics in this direction include a treatment of object-style inheritance and linear logic programming [1, 2, 4], a treatment of concurrent constraint programming [14], and a closely related treatment of Milner's π-calculus [12].

The 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. Provability of purely multiplicative propositional formulas is NP-complete [6]. In fact, the decision problem for constant-only formulas in multiplicative propositional linear logic is also NP-complete [11]. If the additives are also allowed, provability for propositional formulas is PSPACE-complete [8]. Finally, if the modalities are also allowed, full propositional (i.e., quantifier-free) linear logic is undecidable [8]. The decidability of the propositional (i.e., quantifier-free) fragment without additives and the decidability of second-order propositional linear logic without modalities are open problems.

In the remainder of the paper we shall restrict our attention to the non-modal, or multiplicative-additive linear logic with first-order quantifiers and function symbols, MALL. 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, MALL is still PSPACE-complete. However, in the presence of function symbols it can be shown that provability in MALL is NEXPTIME-hard in spite of the same bound on the depth (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 MALL 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
First-order linear logic without modalities

(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.

The membership of MALL in NEXPTIME, and hence its NEXPTIME-completeness, has been shown subsequently by the first author and Shankar [10] using a decision procedure based on dynamic skolemization. This technique was developed in [17] for theorem proving in intuitionistic logic.

The precise descriptions of MALL 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 MALL formulas is given in Section 3. This encoding is reminiscent of the standard proof of the PSPACE-hardness of quantified boolean formula validity [19,13,5]. The encoding is also related to the logic programming simulation of Turing machines given in [18]. In Section 4 it is shown that whenever a nondeterministic exponential time Turing machine accepts, then the MALL formula encoding the machine is provable. The converse is shown in Section 5.

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 $\Gamma, \Delta$ 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 $1$, $\bot$, $T$, $0$) 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 MALL sequent calculus are given in Fig. 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,
- $\Gamma, \Delta, \Sigma, \emptyset, \Xi$ Arbitrary finite multisets of formulas.

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

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: $(\neg H')'$ is $H$, $(A \otimes B)'$ is $A \neg (A \otimes B) = A \neg B$, $(A \neg B)'$ is $A \neg B$, $(A \otimes B)'$ is $A \neg (A \otimes B)$, $(\forall X . A)'$ is $(\exists X . A)'$, $(\exists X . A)'$ is $(\forall X . A)'$.

Fig. 1. Rules for mall1.
Although the identity rules, i.e., the first four rules in Fig. 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 \( \Gamma, A \vdash A \) one can infer \( \Gamma \vdash A \perp \), \( A \), and from \( \Gamma \vdash A \), \( A \) one can infer \( \Gamma \), \( A \perp \vdash A \). The reader will also observe that the linear logic connective \textit{par} may be defined by letting \( A \Implies B \) be \( A \perp \Implies B \) and that the \textit{par} rule [3] is derivable. In fact, our sequent calculus presentation is equivalent to the first order sequent calculus in [3] without exponentials \(!, ?. More precisely, a sequent \( \Gamma \vdash A \) is provable in \textsc{moll} \1 (without cut) iff the sequent \( \Gamma \vdash \Gamma \perp \), \( A \) is provable (without cut, respectively) in the first order sequent calculus in [3], excluding exponentials. In particular, cut-elimination for \textsc{moll} \1 follows from the cut-elimination shown in [3], see also [8]. For the record we have the following theorem.

**Theorem 2.1.** If a sequent is provable in \textsc{moll} \1, then it is provable in \textsc{moll} \1 without using the Cut rule.

We end this section by considering some technical properties of \textsc{moll} \1, in particular the so-called \textit{permutability} properties, some of which have been discussed in several texts [1, 7].

The **sign** of a formula is defined using the transformations below, where \( @ \) is any binary connective except \( \neg \) and where \( Q \) is either of the two quantifiers:

\[
\begin{align*}
[A \Implies B]^+ &= [A]^\neg \Implies [B]^+, \\
[A @ B]^+ &= [A]^+ @ [B]^+, \\
[QX. A]^+ &= QX. [A]^+, \\
[A^\perp]^+ &= ([A]^\neg)^+, \\
[A \Implies B]^- &= [A]^\neg \Implies [B]^-, \\
[A @ B]^- &= [A]^+ @ [B]^-, \\
[QX. A]^- &= QX. [A]^-, \\
[A^\perp]^- &= ([A]^\neg)^+.
\end{align*}
\]

The sign of an instance of a formula \( A \) in a sequent \( \Sigma \vdash A \) is given by the superscript of \( A \) in \( \Sigma \Implies \neg \) or \( [A]^+ \). 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 \([A]^\neg \), 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, T, of their negations, or of connectives \( \otimes \) or \( \neg \). 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 \( \Gamma, A \otimes B \vdash \Delta \) 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 \( \Gamma \vdash A \rightarrow B, \Delta \) is provable in MALL1, then so is \( \Gamma \vdash A, \Delta \) and \( \Gamma \vdash B, \Delta \).

**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 \( \Gamma, \Delta \) 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 \( \Gamma, \Delta \) 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 \( \Gamma, \Delta \) 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 $\Gamma \vdash \exists X. A$, $A$ is provable in MALL, where $\Gamma, A$ contain only positive occurrences of $\Box$ and $\exists$ and only negative occurrences of $\&$ and $\forall$, then for some term $t$ $\Gamma \vdash A[t/X] \vdash A$ is provable in MALL.

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 $\langle Q, \mathcal{S}, \mathcal{T}, \delta, q_0, \mathcal{B}, q_F \rangle$, 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 $Q$,
- $\mathcal{S} \subseteq \mathcal{T}$ is a finite set of input symbols,
- $\mathcal{B} \in \mathcal{T} \setminus \mathcal{S}$ is the blank symbol,
- $\delta$ is the next move relation, a relation from $(Q \setminus \{q_F\}) \times \mathcal{T}$ to $Q \times \mathcal{T} \times \{\text{Left}, \text{Right}\}$.

For instance, if $\delta$ contains move instructions $q_5 \mapsto q_6, c \mapsto q_3, d, \text{Left}$ and $q_5 \mapsto q_3, d, \text{Right}$, 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 $\iota$, is given by a quadruple $\langle q, q_0, \mathcal{B}, t \rangle$, where

- $q \in Q$ is the current state,
- $q_0 \in Q$ is the initial state,
- $\mathcal{B} \in \mathcal{T}$ is the blank symbol,
- $t \in \mathcal{T}^*$ is the used portion of the tape.

The machine is given input $t \in \mathcal{T}^*$ by writing $\mathcal{B}t$ 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 initial configuration $\langle q_0, \mathcal{B}, t \rangle$. $M$ 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 $v_n = \langle q, q_F, \mathcal{B}, \mathcal{B} \rangle$. Note that the state of $v_i$ cannot be $q_F$ whenever $0 < i < n$. $M$ 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 $t \in \mathcal{T}^*$ 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 $\text{NEXPTIME}$ be the class of languages of the form $\mathcal{L}(M)$ for some exponential time Turing machine $M[19]$. 
3. Encoding NEXPTIME in MALL1

In this section we given 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 [18]. Let $id$ be a unary predicate symbol, $\lfloor \cdot \lfloor \cdot \rceil$ a binary function symbol, and $\langle \cdot, \cdot, \cdot, \cdot \rangle$ a function symbol of arity four. A left move Turing machine instruction is translated as

$$[q, a \mapsto q_j, d, \text{Left}] = \forall L \forall R \exists \langle L, q_i, a, [d|R] \rangle \because id(\langle L, q_j, c, [d|R] \rangle)$$

$$\& \forall R \exists \langle L, q_j, c, [d|R] \rangle \because id(\langle L, q_j, c, [d|R] \rangle),$$

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

$$[q_1, a \mapsto q_j, d, \text{Right}] = \forall L \forall R \exists \langle L, q_i, a, [c|R] \rangle \because id(\langle c, q_j, a, [d|R] \rangle)$$

$$\& \forall L \exists \langle L, q_i, a, [c|R] \rangle \because id(\langle c, q_j, a, [d|R] \rangle),$$

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 $\delta = \tau_1, \tau_2, \ldots, \tau_m$, we construct the following formula:

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

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 [19,13]. We develop an encoding function $M$, where $M_k(V, V')$ is meant to denote that from $\text{id } V$, one can reach $\text{id } V'$ in at most $2^k$ steps. A first attempt would be to use the encoding:

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

However, since $M_{k-1}$ 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, but which uses only one copy of $M_{k-1}$ to assert existence of paths from $V$ to $W$ and from $W$ to $V'$. Here and throughout $P$ is a fixed unary predicate symbol.
First-order linear logic without modalities

distinct from \( id \) and \( A \rightarrow B \) abbreviates \( 1 \& (A \rightarrow B) \).

\[
M_0(V, V') = P(V) \rightarrow P(V'),
\]

\[
M_k(V, V') = \exists W . \forall Y . \forall Z . \left( (P(Y) \rightarrow P(V)) \bigoplus (id(W) \rightarrow P(Z)) \right) \rightarrow M_{k-1}(Y, Z).
\]

In this encoding, \( M_{k-1} \) 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 \( \text{mall}\) encoding of a Turing machine accepting an input in at most \( 2^n \) steps as follows. Let \( \text{end} \) be a fresh propositional symbol. Given a number \( n \), a Turing machine \( M = \langle Q, \mathcal{P}, \mathcal{T}, \delta, q_0, b, q_F \rangle \), and an input \( t \), let \( v_0 = \langle \epsilon, q_0, b, t \rangle \) and define the encoding \( [M(t)]_n \) as the \( \text{mall} \) sequent:

\[
C_{\delta, V, L, V_R}((P(X) \rightarrow id(v_0)) \rightarrow ((id(L, q_F, b) \rightarrow P(X')) \rightarrow M_n(X, X')) \rightarrow \text{end}).
\]

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

3.1. Encoding logic programs in \( \text{mall} \)

It follows from [18] 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:

\( A \leftarrow A' \)

\( A \leftarrow \)

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 \( \text{mall} \) by a straightforward modification of the above encoding of nondeterministic Turing machines, namely:

\[
[A_i(f(X)) \leftarrow A_j(h(X)))] = \forall \tilde{X} . \mathit{id}(\langle g_i, f(\tilde{X}) \rangle) \leftarrow \mathit{id}(\langle g_j, h(\tilde{X}) \rangle),
\]

\[
[A_i(f(X)) \leftarrow] = \forall \tilde{X} . \mathit{id}(\langle g_i, f(\tilde{X}) \rangle) \leftarrow \mathit{id}(\langle g_E, \epsilon \rangle),
\]

where constant \( g_E \) is distinct from the constants \( g_1, g_j, \tilde{f} = f_1, \ldots, f_k, \tilde{h} = h_1, \ldots, h_k, \tilde{X} = X_1, \ldots, X_k \), 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 $\text{MALL}_1$. The main technical point is given by

**Lemma 4.1.** If a Turing machine with program $\delta$ can pass from an $id$ $v$ to an $id$ $v'$ 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

$$P(Y_j) \rightarrow P(Y_{j-1}),$$

$$\ldots,$$

$$[\delta], \quad P(Y_j) \rightarrow id(v),$$

$$id(v') \rightarrow P(Z_m),$$

$$\ldots,$$

$$P(Z_{k-1}) \rightarrow P(Z_k)$$

is provable in $\text{MALL}_1$.

**Proof.** Let $\Gamma$ consist of $P(Y_j) \rightarrow P(Y_{j-1}), \ldots, P(Y_0) \rightarrow id(v)$ and let $\Delta$ consist of $id(v') \rightarrow P(Z_m), \ldots, P(Z_{k-1}) \rightarrow P(Z_k)$. The argument is by induction on $n$. If $n = 0$, then $M_0(Y_j, Z_k) = P(Y_j) \rightarrow P(Z_k)$.

There are two cases: either 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 $id(v) \rightarrow id(v')$ is an instance of a conjunct $C$ in $[\tau]$ and thus a $\text{MALL}_1$ proof of the sequent

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

is readily obtained by several instances of $I+, \rightarrow L$, and by $\forall L$. Then a required $\text{MALL}_1$ proof follows by $\rightarrow R$ and several instances of $\perp L$ and $\& L$. 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 $\pi$ be a $\text{MALL}_1$ proof of the sequent

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

and let $\pi'$ be a $\text{MALL}_1$ proof of the sequent

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

both of which exist by the induction hypothesis. Let $\rho$ be the $\text{MALL}_1$ proof of

$$[\delta], \Gamma, \Delta, P(Y_{j+1}) \rightarrow P(Y_j), id(w) \rightarrow P(Z_{k+1}) \vdash M_n(Y_{j+1}, Z_{k+1})$$
obtained from $\pi$ by several instances of $1L$ and $\&L$. Use $\otimes L$ to obtain the MALL1 proof $\sigma$ 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 $\sigma'$ 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 $\pi'$. Then the required MALL1 proof of

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

can be obtained from $\sigma, \sigma'$ by $\oplus L$, $\neg R$, $\forall R$, $\exists R$, and $3R$. □

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

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

A MALL1 proof of $[M(t)]_n$ may then be constructed from $\pi$ first by using two instances of $\neg R$, then $\neg L$ 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 the following proposition.

**Proposition 5.1.** For any Turing machine $M = \langle Q, \mathcal{S}, \mathcal{F}, \delta, q_0, \beta, q_F \rangle$, well-formed Turing machine ID $v$, instruction $\tau \in \delta$, and universally quantified conjunct $C$ in the formula $[[\tau]]$, if $id(v) \rightarrow id(v')$ 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 $\tau$ 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
sequences of the form \([\delta], \Xi \vdash M_k(X, Y)\), where \(k < n\) and \(\Xi\) includes exactly two occurrences of the predicate symbol \(id\), one negative and one positive. Informally, one may read such a sequent as asserting that the machine instructions \(\delta\) can be used to move from the negative occurrence of \(id\) in \(\Xi\) to the positive occurrence of \(id\) in \(\Xi\) 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 \(A \rightarrow B\) abbreviates \(1 & (A \rightarrow B)\).

**Lemma 5.2.** Let \(M = \langle Q, \mathcal{F}, \delta, q_0, q_f \rangle\) be a Turing machine and \(n\) a natural number. For any well-formed Turing machine \(\mathcal{I} \mathcal{D} v\), for any natural numbers \(j, k\) and any natural numbers \(i \leq j\) and \(m \leq k\), if a sequent

\[
P(Y_j) \rightarrow P(Y_{j-1}),
\]

\[
\ldots,
\]

\[
[\delta], \Sigma, \Theta, \quad P(Y_i) \rightarrow id(v),
\]

\[
id(v') \rightarrow P(Z_m), \quad \vdash M_n(Y_j, Z_k)
\]

\[
\ldots,
\]

\[
P(Z_{k-1}) \rightarrow P(Z_k)
\]

is provable in \(\text{MLL}^1\) for some term \(v'\), where \(\Sigma\) is a union of disjoint multisets of the form \(P(Y_j) \rightarrow P(Y_{j-1}), \ldots, P(Y_i) \rightarrow id(u)\) for some natural numbers \(i \leq j < i\) and some term \(u\), where \(\Theta\) is a union of disjoint multisets of the form \(id(v') \rightarrow P(Z_m), \ldots, P(Z_{k-1}) \rightarrow P(Z_k)\) 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 \(\Sigma, \Theta\), then \(v'\) is a well-formed Turing machine \(\mathcal{I} \mathcal{D}\), and \(M\) can make a transition from \(\mathcal{I} \mathcal{D} v\) to \(\mathcal{I} \mathcal{D} v'\) in at most \(2^n\) steps.

**Proof.** Let \(\Gamma\) consist of \(P(Y_j) \rightarrow P(Y_{j-1}), \ldots, P(Y_i) \rightarrow id(v)\) and let \(\Delta\) consist of \(id(v') \rightarrow P(Z_m), \ldots, P(Z_{k-1}) \rightarrow P(Z_k)\). \(\Gamma, \Delta\) will be called the active assumptions, while \(\Sigma, \Theta\) 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, \(\exists & R\) and a string of \&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 \(I', \Delta', \Sigma', \Theta'\) consist of linear implications or of \(I's\). If \(C\) is \([\tau]\) for some instruction \(\tau \in \delta\), then by Proposition 2.11 we may assume that the next higher rule is \(\forall L\) 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 $\Sigma, \Theta$ it follows that $\Sigma', \Theta'$ must consist entirely of 1's, that $\Gamma', \Delta'$ must consist entirely of linear implications, and that $A$ is in fact $id(v) \dashv id(v')$. In other words, the only way to achieve a balanced sequent is to drop the passive assumptions and to instantiate $C$ as $id(v) \dashv id(v')$. 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 MALL proof of

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

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: $\exists R, \forall R, \forall R, \neg R$, and $\otimes L$. This last step yields to 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

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

and

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

This is the key point of the argument. 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 $P(Y_{j+1}) \rightarrow P(Y_j), \Gamma, id(w) \rightarrow P(Z_{k+1})$ and the passive assumptions are $A, \Sigma, \Theta$. In the second branch, the active assumptions are $P(Y_{j+1}) \rightarrow id(w), \Delta, P(Z_k) \rightarrow P(Z_{k+1})$ and the passive assumptions are $\Gamma, \Sigma, \Theta$. 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 $\neg L$ 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 MALL, then $M$ accepts $t$ in at most $2^n$ steps.
Proof. Let \( v_0 = \langle \epsilon, q_0, b, t \rangle \) 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

\[
[\delta], \forall \ell, \forall R(\langle P(X) \rightarrow \text{id}(v_0) \rangle \rightarrow \langle \text{id}(\langle \ell, q, b, \rho \rangle \rightarrow P(X') \rangle \rightarrow M_n(X, X'))
\]

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

\[
[\delta], \langle (P(X) \rightarrow \text{id}(v_0)) \rightarrow \langle \text{id}(\langle \ell, q, b, \rho \rangle \rightarrow P(X') \rangle \rightarrow M_n(X, X') \rangle
\]

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

\[
\langle P(X) \rightarrow \text{id}(v_0) \rangle \rightarrow \langle \text{id}(\langle \ell, q, b, \rho \rangle \rightarrow P(X') \rangle \rightarrow M_n(X, X') \rangle
\]

Because \( \text{end} \), \( 0 \), \( T \), \( \bot \) do not occur in \([\delta]\), \( M_n(X, X') \), the left premise of this instance of \( \neg L \) may not contain \( \text{end} \), and thus we may consider a cut-free MALL1 proof of

\[
[\delta], \langle (P(X) \rightarrow \text{id}(v_0)) \rightarrow \langle \text{id}(\langle \ell, q, b, \rho \rangle \rightarrow P(X') \rangle \rightarrow M_n(X, X') \rangle
\]

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

\[
[\delta], \langle (P(X) \rightarrow \text{id}(v_0)) \rightarrow \langle \text{id}(\langle \ell, q, b, \rho \rangle \rightarrow P(X') \rangle \rightarrow M_n(X, X') \rangle
\]

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

\[
[\delta], \langle (P(X) \rightarrow \text{id}(v_0)) \rightarrow \langle \text{id}(\langle \ell, q, b, \rho \rangle \rightarrow P(X') \rangle \rightarrow M_n(X, X') \rangle
\]

Lemma 5.2 now applies with \( \Sigma, \Theta \) empty. Hence \( \langle \ell, q, b, \rho \rangle \) is a well-formed ID, in fact the final ID, and \( M \) can reach it from \( v_0 \) in at most \( 2^n \) steps. \( \square \)

The complexity class NEXPTIME is defined to be the union of \( \text{NTIME}(2^{p(n)}) \) taken over all polynomials \( p \). Our encoding \([M(t)]_n\) seemingly captures only \( \text{NTIME}(2^n) \). Nevertheless, it follows from Theorems 4.2 and 5.3 that MALL1 is NEXPTIME-hard. Indeed, a polynomial \( p(n) \) may be fixed for our encoding by choosing a particular NEXPTIME-complete problem whose decision algorithm \( M_0 \) is in \( \text{NTIME}(2^{p(n)}) \) [13]. Hence it suffices to consider the encoding \([M_0(t)]_{p(n)}\) on inputs \( t \) of size \( n \).

Theorem 5.4 Provability in MALL1 is NEXPTIME-hard.

Acknowledgment

We would like to thank Yuri Gurevich for suggesting that the logic programming simulation of Turing machines given in [18] 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-Yves Girard, Jean-Marc Andreoli, Jawahar Chirimar, Jean Gallier, Carl Gunter, Joshua Hodas, Jim Lipton, Narciso Marti-Oliet, Dale Miller, John Mitchell, Jon Riecke, Vijay Saraswat, Phil Scott, Natarajan Shankar, and Tim Winkler for interesting and inspiring discussions.
References