Deciding Provability of Linear Logic Formulas

To appear in *Advances in Linear Logic*
London Mathematical Society Lecture Notes Series, Cambridge Univ. Press

Patrick Lincoln*

December 15, 1994

A great deal of insight about a logic can be derived from the study of the difficulty of deciding if a given formula is provable in that logic. Most first order logics are undecidable and are linear time decidable with no quantifiers and no propositional symbols. However, logics differ greatly in the complexity of deciding propositional formulas. For example, first-order classical logic is undecidable, propositional classical logic is NP-complete, and constant-only classical logic is decidable in linear time. Intuitionistic logic shares the same complexity characterization as classical logic except at the propositional level, where intuitionistic logic is PSPACE-complete. In this survey we review the available results characterizing various fragments of linear logic. Surprises include the fact that both propositional and constant-only linear logic are undecidable. The results of these studies can be used to guide further proof-theoretic exploration, the study of semantics, and the construction of theorem provers and logic programming languages.

1 Introduction

There are many interesting fragments of linear logic worthy of study in their own right, most described by the connectives which they employ. Full linear logic includes all the logical connectives, which come in three dual pairs: the exponentials ! and ?, the additives & and $\oplus$, and the multiplicatives $\otimes$ and $\parr$.

*SRI International Computer Science Laboratory, Menlo Park CA 94025 USA. Work supported under NSF Grant CCR-9224858. lincoln@csl.sri.com
http://www.csl.sri.com/lincoln/lincoln.html

For the most part we will consider fragments of linear logic built up using these connectives in any combination. For example, full linear logic formulas may employ any connective, while multiplicative linear logic (MLL) formulas contain only the multiplicative connectives $\otimes$ and $\parr$,and multiplicative-additive linear logic (MALL) formulas contain only the multiplicative and additive connectives $\otimes$, $\parr$, $\&$, and $\oplus$. In some cases it is easier to read a formula such as $A \parr B$ using $A^{\perp} \multimap B$ (which may be read right-to-left as the definition of the connective $\multimap$). Using the connective $\multimap$ one can define other more specific fragments such as the Horn fragment of MLL [25], but these results will be largely omitted here.

In order to gain an intuition about provability, we will usually be speaking informally of a computational process searching for a proof of a formula from the bottom up in a sequent-calculus. Thus given a conclusion sequent, we attempt to find its proof by trying each possible instance of each sequent proof rule. This point of view directly corresponds to the computational model of logic programming. Reading the sequent rules bottom-up can then lead to insights about the meanings of those rules. For example, the contraction rule can be seen as copying the principle formula, and the weakening rule can be seen as throwing it away. Not all complexity results directly flow from this viewpoint, but it is a useful starting point.

Linear logic has a great control over resources, through the elimination of weakening and contraction, and the explicit addition of a reusable (modal) operator. As will be surveyed below, the combination of these features yields a great deal of expressive power.

2 Propositional Linear Logic

The propositional fragment is considered first, since these results are central to the results for first order and constant-only logics.

2.1 Full Propositional Linear Logic

Although propositional linear logic was known to be very expressive, it was thought to be decidable for some time before a proof of undecidability surfaced [32, 31]. Briefly, the proof of undecidability goes by encoding an undecidable halting problem. A proof, read bottom up, directly corresponds to a computation. The proof of the undecidability of full linear logic proceeds by reduction of a form of alternating counter machine to propositional linear logic. An and-branching two-counter machine (ACM) is a nondeterministic machine with a finite set of states. A configuration is a triple $\langle Q_i, A, B \rangle$, where Q_i is a state, and A and B are natural numbers, the values of two counters. An

ACM has a finite set of instructions of five kinds: **Increment-A**, **Increment-B**, **Decrement-A**, **Decrement-B**, and **Fork**. The **Increment** and **Decrement** instructions operate as they do in standard counter machines [39]. The **Fork** instruction causes a machine to split into two independent machines: from state $\langle Q_i, A, B\rangle$ a machine taking the transition $Q_i\mathbf{Fork}Q_j, Q_k$ results in two machines, $\langle Q_j, A, B\rangle$ and $\langle Q_k, A, B\rangle$. Thus an instantaneous description is a **set of** machine configurations, which is accepting only if all machine configurations are in the final state, and all counters are zero. ACM's are essentially alternating Petri nets, and have an undecidable halting problem. It is convenient to use ACM's as opposed to standard counter machines to show undecidability, since zero-test has no natural counterpart in linear logic, but there is a natural counterpart of **Fork**: the additive conjunction &. The remaining ACM instructions may be encoded using techniques very similar to the well-studied Petri net reachability encodings [8, 18, 38, 9, 16]. The full proof of undecidability is presented in [32].

2.2 Propositional Multiplicative-Additive Linear Logic

The multiplicative-additive fragment of linear logic (MALL) excludes the reusable modals !, ?. Thus, every formula is "used" at most once in any branch of any cut-free MALL proof. Also, in every non-cut MALL rule, each hypothesis sequent has a smaller number of symbols than the conclusion sequent. This provides an immediate linear bound on the depth of cut-free MALL proofs. Since MALL enjoys a cut-elimination property [17], there is a nondeterministic PSPACE algorithm to decide MALL sequents based on simply guessing and checking the proof, recoding only the branch of the sequent proof from the root to the current point.

To show that MALL is PSPACE-Hard, one can encode classical quantified boolean formulas (QBF). For simplicity one may assume that a QBF is presented in prenex form. The quantifier-free formula may be encoded using truth tables, but the quantifiers present some difficulty. One may encode quantifiers using the additives: $\forall x$ as $(x\&x^{\perp})$, and $\exists x$ as $(x \oplus x^{\perp})$. This encoding has incorrect behavior in that it does not respect quantifier order, but using multiplicative connectives as "locks and keys" one can enforce an ordering upon the encoding of quantifiers to achieve soundness and completeness. The full proof of PSPACE-completeness is presented in [32].

2.3 Propositional Multiplicative Linear Logic

The multiplicative fragment of linear logic contains only the connectives $\otimes$ and $\parr$ (or equivalently $\otimes$ and $\multimap$), a set of propositions, and the constants 1 and $\bot$. The decision problem for this fragment is in NP, since an entire cut-

free multiplicative proof may be guessed and checked in polynomial time (note that every connective is analyzed exactly once in any cut-free MLL proof). The decision problem is NP-hard by reduction from 3-Partition, a problem which requires a perfect partitioning of groups of objects in much the same way that linear logic requires a complete accounting of propositions [23, 24, 25]. The proof of correctness of the encoding makes heavy use of the 'balanced' property of MLL, which states that if a formula is provable in MLL, then the number of positive and negative occurrences of each literal are equal. This property can be used as a necessary condition to provability in MLL theorem provers or logic programming systems.

3 Constant-Only Linear Logic

3.1 Constant-Only Multiplicative Linear Logic

Some time ago, Girard developed a necessary condition for the provability of COMLL expressions based on the following definition a function M from constant multiplicative linear expressions to the integers:

$$\begin{aligned} M(1) &= 1 \\ M(-) &= 0 \\ M(A \parr B) &= M(A) + M(B) \\ M(A \otimes B) &= M(A) + M(B) - 1 \end{aligned}$$

Girard showed that if a constant-only MLL formula A is provable then $M(A) = 1$. There was a question about whether some similar measure might be used on constant-only MLL formulas that would be necessary and sufficient for provability. It turns out that there is no efficiently computable measure function on this class of formulas, as shown by an encoding of 3-Partition in constant-only MLL [36]. This work points out that the multiplicative constants 1 and $-$ have very 'propositional' behavior. The bottom line is that even for constant-only expressions of MLL deciding provability is NP-complete. This result has had an impact on the study of proof nets. Later results have generalized this result by providing general translations from arbitrary balanced MLL propositional formulas to constant-only MLL formulas [22, 26]. Together with the NP-completeness of propositional MLL, these translations provide an alternate proof of the NP-completeness of constant-only MLL.

3.2 Constant-Only Full Linear Logic

Amazingly, constant-only full linear logic is just as difficult to decide as propositional full linear logic [26]. Extending the work mentioned above, it is possible to translate any full propositional LL formula into a constant-only formula

preserving provability using enumerations of constant-only formulas. Since propositional linear logic is undecidable, so then is constant-only propositional linear logic. This is remarkable, since the building blocks of expressions are so elementary. In fact, the encodings can be tuned to produce very restricted formulas containing multiple copies of only one constant (either 1 or $-$).

3.3 Constant-Only Multiplicative-Additive Linear Logic

The encodings mentioned above can be seen to produce only polynomial growth in the size of formulas. Thus directly translating a class of PSPACE-hard propositional MALL formulas into constant-only MALL immediately produces the result that constant-only MALL is PSPACE-complete [26].

4 First Order Linear Logic

4.1 Full First Order Linear Logic

Girard's translation of first-order classical logic into first order linear logic [17] demonstrates that first order linear logic is undecidable. One could imagine coding up a Turing machine where the instructions are exponential formulas containing implications from one state to another, and the current state of the machine are represented using first order term structure. The conclusion sequent would contain these instructions, an initial state, and a final state. The exponential nature of instructions allows them to be copied and reused arbitrarily often. The quantifier rules allow the instructions to be instantiated to the current state. Thus a Turing machine computation could be read from any cut-free proof of this conclusion sequent bottom up, the intermediate states appearing directly in the sequents all along the way.

4.2 First Order Multiplicative-Additive Linear Logic

Without the exponentials, first order MALL is decidable [35]. Intuitively, this stems from the lack of the ability to copy the instructions for reuse. However, there is no readily apparent decision procedure for this fragment since the quantifier rules allow sequents of arbitrary size to appear even in cut-free proofs. The technique showing decidability sketched here [35] provides a tight complexity bound for first order MALL and MLL.

4.2.1 Deciding first order MALL

The key problem to deciding first order MALL is the lack of control over the existential rule. Reading the rule bottom up we have no idea how to

guess or bound the size of the instantiating term. However, this is a false unboundedness. In classical logic, one can apply Skolemization to remove quantifiers altogether, with changes to the proof rule for identity to require unification. If we could obtain a similar result for linear logic, we could then obtain an immediate bound on the size of instantiations of terms, and thus a bound on the size of the entire sequent proof. Unfortunately, Skolemization is unsound in linear logic, as the following example demonstrates:

$$\vdash (\exists x.p^{\perp} \parr q^{\perp}(x)), (\forall y.q(y)) \otimes p$$

This formula is unprovable in first order linear logic, but when Skolemized it becomes $\vdash p^{\perp} \parr q^{\perp}(v), q(c) \otimes p$ which unfortunately is provable in linear logic augmented with unification $(v \leftarrow c)$.

One can view Skolemization in classical logic as the combination of three techniques: converting the formula to prenex form, permuting the use of quantifier rules below propositional inferences, and changing the quantifier rules to instantiate quantifiers with specific (bounded) terms. Decidability depends only on the last, which is fortunate since neither of the other techniques apply to linear logic in their full generality. In [35, 13] proof systems are developed where the quantifier rules are converted into a form without unbounded guessing. The resulting system generates cut-free proofs of at most exponential size for first-order MALL formulas. It is possible to immediately generate a standard first-order linear logic proof from a proof in this modified system. Thus this fragment can be decided in NEXPTIME by guessing and checking the entire proof in the modified system.

4.2.2 Hardness of first-order MALL

As shown above, first-order MALL is decidable, and at most NEXPTIME-hard. By the propositional result sketched above, it was known to be at least PSPACE-hard. The gap was closed by developing a direct encoding of nondeterministic exponential time Turing machines [33]. This encoding is reminiscent of the standard proof of the PSPACE-hardness of quantified boolean formula validity [43, 21], and is related to the logic programming simulation of Turing machines given in [42]. This encoding in first order MALL formulas is somewhat unique in that the computation is read 'across the top' of a completed cut-free proof, rather than 'bottom up', which is utilized in most of the above-described results. The result is that Turing machine instructions are not copied as one moves up the proof tree, but instead are shared (additively) between branches. This gives an immediate exponential time limit to the machine, since the propositional structure of first-order MALL gives rise to at most a single exponential number of leaves of the proof tree.

4.3 First Order Multiplicative Linear Logic

The same proof system used to show the decidability of first order MALL [35] can be used to show that first order MLL is decidable. In fact, this procedure generates first order MLL proofs that are at most polynomial size. Thus one can guess and check an entire first order MLL proof in polynomial time. In other words, first order MLL is in NP. The propositional hardness result for the purely propositional case can be used to show that first order MLL is NP-hard, and thus NP-complete.

5 Other Fragments

There are many related problems of interest. A few representative 'nice' fragments and some other interesting cases are sketched here.

5.1 Multiplicative-Exponential Linear Logic

The multiplicative-exponential (MELL) fragment is currently of unknown complexity. By Petri net reachability encodings [8, 18, 38, 9, 16], it must be at least EXPSPACE-hard. Although Petri net reachability is decidable, there is no known encoding of MELL formulas in Petri nets. A proof of decidability of MELL may therefore lead to a new proof of the decidability of Petri net reachability, and therefore be of independent interest. More effort has been fruitlessly expended on the decidability of MELL than any other remaining open problems in this area.

5.2 Higher-Order Linear Logic

Amiot has shown that MLL (and MALL) with first and second order quantifiers and appropriate function symbols is undecidable [1, 2].

In recent work, pure second order intuitionistic MLL (IMLL2) has been shown to be undecidable, through the encoding of second order intuitionistic logic [34]. The key point is that it is possible to encode contraction and weakening using second order formulas.

$$C \triangleq \forall X.X \multimap (X \otimes X)$$

$$W \triangleq \forall X.X \multimap 1$$

$$[\Sigma \vdash_{LJ2} \Delta] \triangleq C, C, C, W, [\Sigma] \vdash_{\text{IMLL2}} [\Delta]$$

C encodes contraction and W encodes weakening. A second-order intuitionistic logic (LJ2) sequent can be translated directly into IMLL2, and by adding

enough copies of C and W one can preserve provability. By the undecidability of LJ2 shown in [37, 15], IMLL2 is undecidable. This result can be extended to show the undecidability of pure second order IMALL, but has not yet been extended to pure second order MLL. The decision problem for second order MLL where quantifiers cannot be instantiated with quantified formulas is also still open. This latter fragment could correspond to the logic of a polymorphic type system for a programming language.

5.3 Intuitionistic Fragments

For all of the main fragments considered above, the complexity of the decision problem is unchanged when moving from the full two sided sequent calculus to the Intuitionistic version, where the right-hand side is restricted to a single formula (and $⅋$ is replaced by $\multimap$ in the multiplicatives, and negated propositions are disallowed). However, for the case of second order MLL and second order MALL no result is known, although IMLL2 and IMALL2 are known to be undecidable. In some of the more restricted cases the intuitionistic restriction does effect expressiveness [23]

5.4 Others

The !, $\otimes$ propositional fragment, allowing arbitrary two sided sequents of propositional formulas using only ! and $\otimes$, has been shown to be decidable and non-trivial (NP-hard) [14]. This fragment is of interest as it relates to full MELL.

The Horn fragment of MLL is NP-complete, and that the purely implicative fragment built from only $\multimap$ and the single constant $-$ is NP-complete [26].

Many fragments of linear logic with a single propositional literal (and no constants) match the complexity of the corresponding constant-only fragments, which in turn match the complexity of the propositional fragments with arbitrary numbers of propositional symbols. This has been shown for full linear logic, MALL, MLL, and MELL [26]. For example, full propositional linear logic is undecidable. Therefore, constant-only linear logic is undecidable, as is single-literal propositional linear logic. Of particular interest are the results about MELL, where these results show that the reachability problem for arbitrary Petri nets can be encoded in single-literal MELL. In fact, further, Petri net reachability can be encoded in the fragment containing only !, $\multimap$, and a single propositional symbol. It is remarkable that the decidability of this very small fragment is still open.

Independently, Danos and Winkler have both shown that the constant-only additive-exponential linear logic evaluates in linear time. Even if multiplicative constants are allowed this same result holds. In this fragment there are only eight 'values': $-$, 1, ?1, !$-$, $\top$, 0, $- \oplus 1$, and $-\&1$. It happens that all

expressions involving only the multiplicative and additive constants, and the additive and exponential connectives is equivalent, up to provability, of one of these eight formulas. For example, $!\top = 1$, $!?1 = 1$, $!(- \oplus 1) = 1$, and $-\&(- \oplus 1) = -$.

A variant of MELL without unrestricted exchange (commutativity), but with the additional property that exponentials can commute (and thus exponentials enjoy all the structural rules, while other formulas exhibit none) has been studied [47, 32]. It has been found to be undecidable by encoding Turing machine tapes directly in the sequent [32]. Since the sequent comma is not commutative, the entire state of the tape including the current state of the machine and position of the read head is immediately apparent from a sequent encoding. Instructions are encoded as exponential formulas that are copied and then commute to their location of application, where they are applied to change the state of the tape. However, noncommutative variants of linear logic have some problematic aspects, and there are some seemingly arbitrary choices to be made, so these fragments are somewhat speculative.

Variants of linear logic with unrestricted weakening (sometimes called Affine Logic) have also been studied [32, 7]. Here again the logics are somewhat speculative, although there is a close relationship with direct logic [27, 11]. Some fragments of linear logic with weakening have the same complexity as the same fragment without weakening. For example, just as for linear logic, full first-order affine logic is undecidable, as can be seen by the fact that Girard's encoding of classical logic into linear logic [17] is also sound and complete as a translation into affine logic. Some fragments of affine logic are easier to decide than their linear counterpart. For example, propositional affine logic is decidable [28], where propositional linear logic is not [32]. Finally, some fragments of affine logic are harder to decide than their linear counterpart. For example, the extended Horn fragment +HL is (PSPACE-complete) in affine logic, but (NP-complete) in linear logic [23].

Finally, variants of linear logic with unrestricted contraction are very similar to relevance logic [3]. Urquhart has shown that some propositional variants are undecidable, and has studied other fragments [45, 46]. However, the results regarding relevance logic are very different in character than those described above, since they rely in an essential way on a distributivity that appears in relevance logic but does not appear in linear logic.

6 Conclusions

This survey has sketched the basic approaches used in the study of the complexity of deciding linear logic formulas. This area has led to some new understanding of the fragments in question, and has pointed out some gaps in

current understanding. Of the remaining open problems, perhaps the decidability of propositional MELL and the decidability of pure second-order MALL are of the most interest.

There are some surprisingly rich fragments of linear logic, and surprisingly few differences between the complexity of many fragments at the first order, propositional, and constant-only levels. For example, even constant-only full linear logic is undecidable (as are the first-order and propositional fragments), and first-order MLL, propositional MLL, and constant-only MLL are all NP-complete. However, MALL is PSPACE-complete at the constant-only and propositional levels, but is NEXPTIME-complete at the first-order level.

This area of study is directly relevant to the logic-programming use of linear logic, where linear logic sequents are taken to be logic programs which execute by performing proof search [20, 5, 4, 19]. This area of research is also directly relevant to the construction of linear logic theorem provers [40, 44, 35, 6, 11, 10, 41]. The results here also lead into the study of semantics of linear logic, pointing to deep connections between various fragments of linear logic and familiar structures from computer science [12, 29, 30]. In particular, work has progressed in attempting to find viewpoints where the proof theory of linear logic can be viewed as a machine. For example, Kanovich's results derive from his view of fragments of linear logic as acyclic programs with stack. Turing machines can be seen as described above in special first-order encodings, but various counter or Minsky machines can be seen as somewhat more direct interpretation of the propositional fragments.

Finally, readers should not interpret the above results negatively: the fact that linear logic is expressive is an important feature. Classical logic is degenerate in its small number of well-behaved fragments of different complexity. Linear logic's rich structure simply provides more detail than many other logics. This detail negates the possibility of simple decision procedures, but can carry important information regarding computational content, where other logics record only simple binary results. That is, linear logic is not about "Truth"; it is about computation.

References

[1] Amiot. Decision Problems for Second Order Linear Logic Without Exponentials. Draft, 1990.

[2] G. Amiot. *Unification et logique du second ordre.* PhD thesis, Université Paris 7 (Denis Diderot), 1994.

[3] Anderson, A.R. and N.D. Belnap, Jr. *Entailment. Volume 1.* Princeton University Press, Princeton, New Jersey, 1975.

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

[5] 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.

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

[7] S. Artemov and A. Kopylov. Full propositional affine logic is decidable. Email Message 260 to Linear Mailing List, June 1994.

[8] A. Asperti. A logic for concurrency. Technical report, Dipartimento di Informatica, Universitá di Pisa, 1987.

[9] A. Asperti, G.-L. Ferrari, and R. Gorrieri. Implicative formulae in the 'proofs as computations' analogy. In *Proc. 17-th ACM Symp. on Principles of Programming Languages, San Francisco*, pages 59–71, January 1990.

[10] G. Bellin. *Mechanizing Proof Theory: Resource-Aware Logics and Proof-Transformations to Extract Implicit Information*. PhD thesis, Stanford University, 1990.

[11] G. Bellin and J. Ketonen. A decision procedure revisited: Notes on direct logic, linear logic, and its implementation. *Theoretical Computer Science*, 95:115–142, 1992.

[12] 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.

[13] S. Cerrito. Herbrand methods in linear logic. Unpublished Draft, 1992.

[14] J. Chirimar and J. Lipton. Provability in TBLL: A Decision Procedure. In *Computer Science Logic '91. Lecture Notes in Computer Science, Springer*, 1992.

[15] D.M. Gabbay. *Semantical investigations in Heyting's intuitionistic logic*. Reidel, Dordrecht, 1981.

[16] V. Gehlot and C.A. Gunter. Normal process representatives. In *Proc. 5-th IEEE Symp. on Logic in Computer Science, Philadelphia*, June 1990.

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

[18] C.A. Gunter and V. Gehlot. Nets as tensor theories. In G. De Michelis, editor, *Proc. 10-th International Conference on Application and Theory of Petri Nets, Bonn*, pages 174–191, 1989.

[19] J. Harland and D. Pym. The uniform proof-theoretic foundation of linear logic programming. Technical Report ECS-LFCS-90-124, Laboratory for the Foundations of Computer Science, University of Edinburgh, November 1990.

[20] 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*. Draft available using anonymous ftp from host ftp.cis.upenn.edu and the file pub/papers/miller/ic92.dvi.Z.

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

[22] H. Jervell. Eliminating variables in balanced Horn sequents. Draft.

[23] M. Kanovich. The multiplicative fragment of linear logic is NP-complete. Email Message, 1991.

[24] M. Kanovich. The multiplicative fragment of linear logic is NP-complete. Technical Report X-91-13, Institute for Language, Logic, and Information, June 1991.

[25] 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.

[26] M. Kanovich. Simulating linear logic in 1-only linear logic. Technical Report 94-02, CNRS, Laboratoire de Mathematiques Discretes, January 1994.

[27] J. Ketonen and R. Weyhrauch. A decidable fragment of predicate calculus. *Theoretical Computer Science*, 32:297–307, 1984.

[28] A. Kopylov. Decidability of linear affine logic. Technical Report 94-32, CNRS, Laboratoire de Mathématiques Discrètes, October 1994. Available by ftp from `lmd.univ-mrs.fr:/pub/kopylov/fin.dvi`.

[29] Y. Lafont and T. Streicher. Game semantics for linear logic. In *Proc. 6-th Annual IEEE Symposium on Logic in Computer Science, Amsterdam*, pages 43–50. IEEE Computer Society Press, Los Alamitos, California, July 1991.

[30] P. Lincoln, J. Mitchell, and A. Scedrov. Stochastic interaction and linear logic. To appear Cambridge University Press London Mathematical Society Lecture Notes Series.

[31] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 662–671, 1990.

[32] 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.

[33] P. Lincoln and A. Scedrov. First order linear logic without modalities is NEXPTIME-hard. *Theoretical Computer Science*, 135(1):139–154, December 1994. Available using anonymous ftp from host ftp.cis.upenn.edu and the file pub/papers/scedrov/mall1.dvi.

[34] P. Lincoln, A. Scedrov, and N. Shankar. Decision problems for second order linear logic. Email Message 278 to Linear Mailing List, October 1994.

[35] P. Lincoln and N. Shankar. Proof search in first-order linear logic and other cut-free sequent calculi. In *Proc. 9-th IEEE Symp. on Logic in Computer Science,Paris*, July 1994.

[36] P. Lincoln and T. Winkler. Constant-Only Multiplicative Linear Logic is NP-Complete. *Theoretical Computer Science*, 135(1):155–169, December 1994.

[37] M.H. Loeb. Embedding first-order predicate logic in fragments of intuitionistic logic. *Journal of Symbolic Logic*, 41:705–718, 1976.

[38] N. Martí-Oliet and J. Meseguer. From Petri nets to linear logic. In: Springer LNCS 389, ed. by D.H. Pitt et al., 1989. 313-340.

[39] M. Minsky. Recursive unsolvability of post's problem of 'tag' and other topics in the theory of turing machines. *Annals of Mathematics*, 74:3:437–455, 1961.

[40] G. Mints. Resolution Calculus for the First Order Linear Logic. *Journal of Logic, Language and Information*, 2:59–83, 1993.

[41] H. Schellinx. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1:537–559, 1991.

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

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

[44] T. Tammet. Proof search strategies in linear logic. Programming Methodology Group Report 70, Chalmers University, 1993.

[45] A. Urquhart. The undecidability of entailment and relevant implication. *Journal of Symbolic Logic*, 49:1059–1073, 1984.

[46] A. Urquhart. The complexity of decision procedures in relevance logic. Technical Report 217/89, Department of Computer Science, University of Toronto, 1989.

[47] D.N. Yetter. Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.