線形論理型言語コンパイラ処理系を用いた古典命題線形論理の定理証明システム

田村 直之　番原 睦則

本論文では，著者らの開発した線形論理型言語コンパイラ処理系の LLP を用いて，古典命題線形論理の論理式の証明探索を行うシステム LL2LLP について述べる．本システムは，古典命題線形論理式を直観主義線形論理式に変換し，さらに LLP コンパイラを用いて LLP 抽象機械命令にコンパイルしたのち実行することで高速な証明探索を実現している点に特徴がある．いくつかの線形論理式に対して，既存の線形論理定理証明システムと性能を比較した所，ほとんどの問題でより良い結果が得られた．

This paper describes a theorem prover of propositional classical linear logic implemented on a linear logic programming language developed by the authors. In this LL2LLP system, efficient proof search is realized by transforming classical linear logic formulas into intuitionistic linear logic formulas, and by compiling them into LLP abstract machine instructions. The evaluated performance of LL2LLP was better than other existing provers for most of the benchmark problems.

Classical Propositional Linear Logic Theorem Prover on a Linear Logic Programming Language Compiler System.

Naoyuki TAMURA, 神戸大学学術情報基盤センター, Information Science and Technology Center, Kobe University.

Mutsunori BANBARA, 神戸大学学術情報基盤センター, Information Science and Technology Center, Kobe University.

コンピュータソフトウェア, Vol.22, No.1(2005), pp.98–103.

2004 年 8 月 11 日受付.

1 はじめに

線形論理は 1987 年に Girard が発表して以来 [5], 仕様検証等の分野に応用されており [13] [6] [3] [10] [20], 定理証明技術の発展が期待されている．

既存の線形論理定理証明システムとしては、Tammet による linres, linseq [18]，Mantel らによる linTAP [14] 等がある．linres は命題線形論理式に対して導出原理に類似した方法で前向きに証明探索を行うシステムであるが，様相演算子 (!, ?) の取り扱いが効率的でない．linseq は命題線形論理式に対して後向きに証明探索を行うが，$\otimes$ 等で発生するリソース論理式の分割，$\top$ の取り扱い等で非決定性が大きいという問題点がある．linTAP は命題線形論理式に対して，タブロー法を用いて証明探索を行うシステムであるが，加法的演算子 (&, $\oplus$) を取り扱えない．

一方，線形論理型プログラミング言語について，証明探索を効率化する様々な手法，例えば古典線形論理に対するゴール指向の証明探索手法 [16]，$\otimes$ 等でのリソース分割および $\top$ の非決定性の軽減化手法 [8] [4] [9] [7] が提案されている．また，直観主義線形論理型言語 Lolli については，著者らの開発したコンパイラ処理系 LLP において，それらの手法を利用した効率的な証明探索が実現されている [19] [2] [11] [23] [21].

そこで，本論文では古典線形論理の命題論理式を線形論理型言語 Lolli のプログラムに変換し，LLP コンパイラを用いて抽象機械命令列に翻訳することで，効率的な証明探索を実現する手法を提案する．これは，

Stickel の PTTP [17] の線形論理版といえる．

なお，以下の 2〜3 章では，命題線形論理だけでなく一階述語線形論理を対象に含めた記述を行う．

2 Forum と Lolli

本節では，Miller による古典線形論理型言語 Forum [16] および Hodas らによる直観主義線形論理型言語 Lolli [8] について述べる．

Forum ではプリミティブな論理演算子として $\forall$, $\top$, $\bot$, $\&$, $\wp$ †1, $\multimap$ および直観主義的含意 $\Rightarrow$ (ただし，$B \Rightarrow C$ は $!B \multimap C$ を意味する) を用いており他の論理演算子は以下の同値関係を用いて表現する．

$$0 \equiv \top \multimap \bot \qquad 1 \equiv \bot \multimap \bot$$
$$!B \equiv (B \Rightarrow \bot) \multimap \bot \qquad ?B \equiv (B \multimap \bot) \Rightarrow \bot$$
$$B^{\bot} \equiv B \multimap \bot \qquad \exists x.B \equiv (\forall x.B^{\bot})^{\bot}$$
$$B \oplus C \equiv (B^{\bot} \,\&\, C^{\bot})^{\bot} \qquad B \otimes C \equiv (B^{\bot} \wp C^{\bot})^{\bot}$$

オリジナルの Forum では，様相演算子 ? もプリミティブとしているが，ここでは Hodas らによる体系 [9] に従いプリミティブとしない．

以下では，プリミティブな論理演算子だけを含む線形論理式を考える．

Hodas らによる Forum の体系では，シーケントとして $\Psi; \Delta \longrightarrow \Gamma$ あるいは $\Psi; \Delta \xrightarrow{D} \mathcal{A}$ を用いる．ただし，Ψ は論理式の集合，Δ と Γ は論理式のマルチ集合，D は論理式，$\mathcal{A}$ は原子論理式のマルチ集合である．これらのシーケントは，線形論理での $!\Psi, \Delta \longrightarrow \Gamma$ および $!\Psi, \Delta, D \longrightarrow \mathcal{A}$ にそれぞれ対応する．

Hodas らによる Forum の体系を図 1 に示す．図中 A は原子論理式を表している．なお，Hodas らによる元々の体系では，シーケントの右辺は論理式のリストであるが，ここでは Miller の [15] に従って，論理式のマルチ集合とする．

命題 2.1 (Forum の健全性と完全性) Forum は線形論理に関して健全かつ完全である．

$$\Psi; \Delta \longrightarrow \Gamma \text{ が Forum で証明可能}$$
$$\iff \quad !\Psi, \Delta \longrightarrow \Gamma \text{ が線形論理で証明可能}$$

証明 Andoreoli の Focusing proof [1] を用いた，Miller による証明 [16] を参照． □

Hodas らによる Lolli は，直観主義線形論理に基づいているためプリミティブな論理演算子として $\forall$, $\top$, $\&$, $\multimap$ および $\Rightarrow$ だけを用いる．Lolli の体系は，図 1 の Forum の体系から $\bot$, $\wp$ を含んだ規則を削除することによって得られる．この時，シーケントの右辺はちょうど一つの論理式となる．

命題 2.2 (Lolli の健全性と完全性) Lolli は，$\forall$, $\top$, $\&$, $\multimap$ および $\Rightarrow$ から成る線形論理のフラグメントに関して健全かつ完全である．

$$\Psi; \Delta \longrightarrow G \text{ が Lolli で証明可能}$$
$$\iff \quad !\Psi, \Delta \longrightarrow G \text{ が線形論理で証明可能}$$

証明 Hodas と Miller による証明 [8] を参照． □

3 原子頭部形，ボトム頭部形

本節では，任意の線形論理式を，それと証明可能性が一致し，$\forall$, $\top$, $\&$, $\multimap$ および $\Rightarrow$ だけから成る直観主義線形論理式に変換する方法について述べる．

定義 3.1 (原子頭部形) 線形論理の論理式が以下で定義される P の形をしているとき，**原子頭部形**と呼ぶ $(l, m, n \geq 0)$.

$$P \quad ::= \quad R_1 \,\&\, \cdots \,\&\, R_n$$
$$R \quad ::= \quad \forall \vec{x}.(R_1 \Rightarrow \cdots R_l \Rightarrow P_1 \multimap \cdots P_m \multimap (A_1 \wp \cdots \wp A_n))$$

ただし，A_i は原子論理式，$\forall \vec{x}$ は 0 回以上の全称限量であり，$n = 0$ の時の $R_1 \,\&\, \cdots \,\&\, R_n$ は $\top$ を，$n = 0$ の時の $A_1 \wp \cdots \wp A_n$ は $\bot$ を表すものとする．また，R を**規則**，R 中の $A_1 \wp \cdots \wp A_n$ を**頭部**と呼ぶ．

命題 3.1 任意の線形論理の論理式は，同値な原子頭部形に変換できる．

証明 Andoreoli の論文 [1] に述べられている方法を用いることにより変換できる． □

上記の R の定義中の P_1 が $(\forall \vec{y}.(R'_1 \Rightarrow \cdots R'_{l'} \Rightarrow P'_1 \multimap \cdots P'_{m'} \multimap (A'_1 \wp \cdots \wp A'_{n'}))) \multimap \bot$ である場合，変数名がぶつからないようにすれば，以下のように R を簡単化できる．他の P_i についても同様である．

$$R \quad \equiv \quad \forall \vec{x}.\forall \vec{y}.(R_1 \Rightarrow \cdots R_l \Rightarrow R'_1 \Rightarrow \cdots R'_{l'} \Rightarrow P_2 \multimap \cdots P_m \multimap P'_1 \multimap \cdots P'_{m'} \multimap (A_1 \wp \cdots \wp A_n \wp A'_1 \wp \cdots \wp A'_{n'}))$$

†1 $\wp$ は乗法的論理和 (multiplicative disjunction) を表す．

$$\frac{}{\Psi;\Delta\longrightarrow\top,\Gamma}\ (\mathsf{R}\top) \qquad \frac{\Psi;\Delta\longrightarrow B,\Gamma\quad\Psi;\Delta\longrightarrow C,\Gamma}{\Psi;\Delta\longrightarrow B\,\&\,C,\Gamma}\ (\mathsf{R}\&)$$

$$\frac{\Psi;\Delta\longrightarrow\Gamma}{\Psi;\Delta\longrightarrow\bot,\Gamma}\ (\mathsf{R}\bot) \qquad \frac{\Psi;\Delta\longrightarrow B,C,\Gamma}{\Psi;\Delta\longrightarrow B\,\wp\,C,\Gamma}\ (\mathsf{R}\wp)$$

$$\frac{\Psi;B,\Delta\longrightarrow C,\Gamma}{\Psi;\Delta\longrightarrow B\multimap C,\Gamma}\ (\mathsf{R}\multimap) \qquad \frac{B,\Psi;\Delta\longrightarrow C,\Gamma}{\Psi;\Delta\longrightarrow B\Rightarrow C,\Gamma}\ (\mathsf{R}\Rightarrow)$$

$$\frac{\Psi;\Delta\longrightarrow B[y/x],\Gamma}{\Psi;\Delta\longrightarrow\forall x.B,\Gamma}\ (\mathsf{R}\forall)\quad (y \text{ is not free in the lower sequent of } (\mathsf{R}\forall))$$

$$\frac{\Psi;\Delta\xrightarrow{B}\mathcal{A}}{\Psi;B,\Delta\longrightarrow\mathcal{A}}\ (\mathsf{decide}_1) \qquad \frac{B,\Psi;\Delta\xrightarrow{B}\mathcal{A}}{B,\Psi;\Delta\longrightarrow\mathcal{A}}\ (\mathsf{decide}_2)$$

$$\frac{}{\Psi;\xrightarrow{A}A}\ (\mathsf{initial}) \qquad \frac{\Psi;\Delta\xrightarrow{B}\mathcal{A}}{\Psi;\Delta\xrightarrow{B\,\&\,C}\mathcal{A}}\ (\mathsf{L}\&)\quad\frac{\Psi;\Delta\xrightarrow{C}\mathcal{A}}{\Psi;\Delta\xrightarrow{B\,\&\,C}\mathcal{A}}\ (\mathsf{L}\&)$$

$$\frac{}{\Psi;\xrightarrow{\bot}}\ (\mathsf{L}\bot) \qquad \frac{\Psi;\Delta_1\xrightarrow{B}\mathcal{A}_1\quad\Psi;\Delta_2\xrightarrow{C}\mathcal{A}_2}{\Psi;\Delta_1,\Delta_2\xrightarrow{B\,\wp\,C}\mathcal{A}_1,\mathcal{A}_2}\ (\mathsf{L}\wp)$$

$$\frac{\Psi;\Delta_1\longrightarrow B,\mathcal{A}_1\quad\Psi;\Delta_2\xrightarrow{C}\mathcal{A}_2}{\Psi;\Delta_1,\Delta_2\xrightarrow{B\multimap C}\mathcal{A}_1,\mathcal{A}_2}\ (\mathsf{L}\multimap) \qquad \frac{\Psi;\longrightarrow B\quad\Psi;\Delta\xrightarrow{C}\mathcal{A}}{\Psi;\Delta\xrightarrow{B\Rightarrow C}\mathcal{A}}\ (\mathsf{L}\Rightarrow)$$

$$\frac{\Psi;\Delta\xrightarrow{B[t/x]}\mathcal{A}}{\Psi;\Delta\xrightarrow{\forall x.B}\mathcal{A}}\ (\mathsf{L}\forall)$$

図 1 A variant of a proof system for propositional Forum

定義 3.2 (ボトム頭部形) 線形論理式 B に対して，B 中のすべての原子論理式 A (論理定数は除く) を $A\multimap\bot$ で置き換えた後，その原子頭部形中のすべての規則 $\forall\vec{x}.(I_1\Rightarrow\cdots\Rightarrow I_l\Rightarrow L_1\multimap\cdots\multimap L_m\multimap(A_1\wp\cdots\wp A_n))$ を以下で置き換えた論理式を B の**ボトム頭部形**と呼び $\overline{\overline{B}}$ で表す.

$$\forall\vec{x}.(I_1\Rightarrow\cdots\Rightarrow I_l\Rightarrow L_1\multimap\cdots\multimap L_m\multimap A_1\multimap\cdots A_n\multimap\bot)$$

任意の線形論理式 B について，B とそのボトム頭部形 $\overline{\overline{B}}$ は同値である．この変換は，古典論理の直観主義論理への埋め込みで用いられる double negation 導入と同様の方法である.

次に，新しい原子論理式として $\flat$ を用意し，線形論理式 B 中の $\bot$ を $\flat$ で置き換えた論理式を $B^\flat$ で表す.

命題 3.2 Ψ をボトム頭部形の規則の集合，Δ をボトム頭部形のマルチ集合，$\mathcal{A}$ を原子論理式のマルチ集合，P をボトム頭部形とした時，以下が成立する.

$$\Psi;\Delta,\mathcal{A}\longrightarrow P \text{ が Forum で証明可能}$$
$$\iff \Psi^\flat;\Delta^\flat,\mathcal{A}\longrightarrow P^\flat \text{ が Lolli で証明可能}$$

証明

($\Longrightarrow$) $\Psi;\Delta,\mathcal{A}\longrightarrow P$ の Forum での証明図を考えると，以下の左側の形のシーケントしか現れない.

$\Psi;\Delta,\mathcal{A}\longrightarrow B$	$\Psi^\flat;\Delta^\flat,\mathcal{A}\longrightarrow B^\flat$
$\Psi;\Delta,\mathcal{A}\longrightarrow$	$\Psi^\flat;\Delta^\flat,\mathcal{A}\longrightarrow\flat$
$\Psi;\Delta,\mathcal{A}\xrightarrow{B}$	$\Psi^\flat;\Delta^\flat,\mathcal{A}\xrightarrow{B^\flat}\flat$
$\Psi;\xrightarrow{A}A$	$\Psi^\flat;\xrightarrow{A}A$

これらを右側のシーケントで置き換え，($\mathsf{R}\bot$) の適用を削除し，($\mathsf{L}\bot$) を (initial) と置き換えたものは Lolli での証明図となる.

($\Longleftarrow$) 同様にして証明できる. □

この命題が成立するのはボトム頭部形に対してであって，任意の論理式あるいは原子頭部形に対してではない．たとえば論理式 $P=((A\multimap\bot)\multimap\bot)\multimap A$ は Forum で証明可能であるが，$P^\flat$ は Lolli で証明可能でない．一方 $\overline{\overline{P}}^\flat=((((A\multimap\flat)\multimap\flat)\multimap\flat)\multimap\flat)\multimap(A\multimap\flat)\multimap\flat$ は Lolli でも証明可能である.

命題 3.3 任意の線形論理式 B について，B の線形論理での証明可能性と，$\overline{\overline{B}}^\flat$ の Lolli での証明可能性は一致する.

証明 B とそのボトム頭部形 $\overline{\overline{B}}$ が同値であること，

命題 3.2 より $\overline{\overline{B}}$ の Forum での証明可能性と $\overline{\overline{B}}^{\flat}$ の Lolli での証明可能性が一致することから示せる．□

例 3.1 以下の線形論理式 LE の変換を例に取る．

$$\begin{aligned} LE \quad \equiv \quad & (!(le \otimes a \otimes b \multimap le) \otimes !(le \otimes b \multimap le) \otimes \\ & !(le \multimap 1) \otimes le \otimes a^m \otimes b^n) \multimap 1 \end{aligned}$$

a^m, b^n はそれぞれ m 個の a の $\otimes$-積と n 個の b の $\otimes$-積を表す．この論理式は $m \leq n$ の場合にのみ証明可能である．この時，$\overline{\overline{LE}}^{\flat}$ は以下のようになる．

$$\begin{aligned} \overline{\overline{LE}}^{\flat} \quad \equiv \quad & ((le \multimap \flat) \multimap le \multimap a \multimap b \multimap \flat) \Rightarrow \\ & ((le \multimap \flat) \multimap le \multimap b \multimap \flat) \Rightarrow \\ & (\flat \multimap le \multimap \flat) \Rightarrow \\ & \flat \multimap le \multimap \underbrace{a \multimap \cdots a \multimap}_{m} \underbrace{b \multimap \cdots b \multimap}_{n} \flat \end{aligned}$$

一般に，ボトム頭部形 $\overline{\overline{B}}^{\flat}$ のサイズ (原子論理式の数と論理結合子の数の合計) は，もとの論理式 B のサイズよりも大きくなる．これは，原子頭部形への変換で，$(a \,\&\, b) \wp (c \,\&\, d)$ 等に対する分配則の適用が必要となるためであり，一階述語論理式の節形式への変換と同様の状況である．

4 LL2LLP

線形論理の定理証明システム LL2LLP は，基本的には前節での変換方法に従って，線形論理の論理式を Lolli のプログラムに変換し，それを LLP コンパイラ処理系でコンパイル・実行することで証明探索を行うが，さらに以下の手法を導入する．

規則をそれぞれ別々のプログラム節に分割する．たとえば，前節の $\overline{\overline{LE}}^{\flat}$ 中の $(le \multimap \flat) \multimap le \multimap a \multimap b \multimap \flat$ を $r \multimap \flat$ (r は新しい原子論理式) に置き換え，$(le \multimap \flat) \multimap le \multimap a \multimap b \multimap r$ に相当する以下のプログラム節を追加する．

$$r \texttt{ :- } (le \multimap \flat),\ le,\ a,\ b.$$

また，原子論理式のゴールをボディ部の先頭に移し，リソースが存在しない場合に早めに失敗するようにし，また `once` メタ述語を使用することで，複数の同一リソースが存在する場合の非決定性をなくす．

$$r \texttt{ :- once}(le),\ \texttt{once}(a),\ \texttt{once}(b),\ (le \multimap \flat).$$

さらに，LLP での実行は depth-first で行われるため，Iterative Deepening の手法を用い，完全な証明探索を実現する．すなわち，各プログラム節に探索の深さを表す引数を追加する．証明探索はまず深さ 0 に対して行い，証明が見つからなければ順に深さをインクリメントしながら探索を進める．

5 LL2LLP の性能評価

LL2LLP の性能評価のため，5 種類の線形論理の論理式を用い，linres, linseq, linTAP と証明探索時間を比較した．これらの論理式のうち，F_4 と F_7 は linTAP [14] で，LMSS-2 と LMSS-3 は linres と linseq [18] の性能評価で用いられた論理式である．また，m-MxN は線形論理の論理式を用いて掛け算を表現したものであり，以下の通りである．

$$\begin{aligned} & !((q_0 \otimes b) \multimap q_1) \multimap !(q_0 \multimap (z_b \oplus q_3)) \multimap \\ & !((q_1 \otimes a \otimes c) \multimap (q_1 \otimes d)) \multimap !(q_1 \multimap (z_a \oplus q_2)) \multimap \\ & !((q_2 \otimes d) \multimap (q_2 \otimes a)) \multimap !(q_2 \multimap (z_d \oplus q_0)) \multimap \\ & !((q_3 \otimes a) \multimap q_3) \multimap !(q_3 \multimap (z_a \oplus q_f)) \multimap \\ & !(z_a \multimap q_f) \multimap !((z_a \otimes b) \multimap z_a) \multimap \\ & !((z_a \otimes c) \multimap z_a) \multimap !((z_a \otimes d) \multimap z_a) \multimap \\ & !(z_b \multimap q_f) \multimap !((z_b \otimes a) \multimap z_b) \multimap \\ & !((z_b \otimes c) \multimap z_b) \multimap !((z_b \otimes d) \multimap z_b) \multimap \\ & !(z_c \multimap q_f) \multimap !((z_c \otimes a) \multimap z_c) \multimap \\ & !((z_c \otimes b) \multimap z_c) \multimap !((z_c \otimes d) \multimap z_c) \multimap \\ & !(z_d \multimap q_f) \multimap !((z_d \otimes a) \multimap z_d) \multimap \\ & !((z_d \otimes b) \multimap z_d) \multimap !((z_d \otimes c) \multimap z_d) \multimap \\ & q_0 \multimap \underbrace{a \multimap \cdots a \multimap}_{M} \underbrace{b \multimap \cdots b \multimap}_{N} \underbrace{c \multimap \cdots c \multimap}_{M\,N} q_f \end{aligned}$$

計測は Pentium M 1GHz の Linux マシン上で行い，変換時間およびコンパイル時間を含めて CPU 時間が 5 分以上の場合はタイムアウトとして実行を打ち切った．

linres と linseq は共に Scheme で記述されているため，Scheme 処理系 SCM 5d9 (ライブラリは SLIB 3a1 を使用) を用いて C にコンパイルし，実行を行った．

linTAP を実行する Prolog コンパイラ処理系としては，高速な商用 Prolog コンパイラの一つとして知られている SICStus Prolog 3.7.1 を用いた．

LL2LLP のための LLP コンパイラ処理系としては LLP 0.51 を使用した[†2]．LLP のプログラムは，WAM を拡張した抽象機械 LLPAM の命令列にコンパイルされた後，LLPAM エミュレータにより実行さ

†2 `http://bach.istc.kobe-u.ac.jp/llp/`

表 1 LL2LLP, linres, linseq, linTAP の実行結果

単位：ミリ秒

論理式	LL2LLP 証明探索のみ	LL2LLP 変換・コンパイル・証明探索	linres	linseq	linTAP
F_4	<10	550	21172	?	30
F_7	<10	260	16	48	662
LMSS-2	16696	17234	1322	?	-
LMSS-3	?	?	1624	?	-
m-1x1	10	598	175012	?	-
m-1x2	34	632	279690	?	-
m-2x2	486	1100	?	?	-

”<10”：10 ミリ秒以下　”-”：対象外　”?”：タイムアウト (5 分)

れる．なお，LLP 0.51 の Prolog 処理系としての性能は，SICStus Prolog と比較した場合は 2 倍程度遅く，フリーの処理系の SWI Prolog と比較した場合は 2 倍程度速くなっている [21].

表 1 に，5 種類の論理式に対する LL2LLP, linres, linseq, linTAP の実行結果を示す．LL2LLP については，証明探索のみの実行時間と，変換・コンパイル・証明探索のすべてを含んだ実行時間を表示している．変換およびコンパイルにかかった平均実行時間は，それぞれ 129 ミリ秒，411 ミリ秒だった．

LL2LLP は，LMSS-3 以外の全ての論理式を証明できた (LMSS-3 も 2 時間程度で証明可能である)．なお，いずれの問題についても，ボトム頭部形のサイズ (原子論理式の数と論理結合子の数の合計) は，元の論理式のサイズの 2 倍以下だった．

linseq は F_7 以外全てタイムアウトとなっている．これは $\otimes$ 等で発生するリソース論理式の分割，$\top$ の取り扱い等で非決定性が大きいという問題が原因である．LL2LLP では，LLP コンパイラでこれらの問題に対する軽減化手法が実装されており，効率的な証明探索が可能となっている．

m-1x1 に対して，LL2LLP は linres と比較して，約 300 倍の高速化を実現している．また m-1x2 など掛け合わせる数字が大きい (論理式の数が増える) ほど，高速化の割合は大きくなっており，様相演算子 (!, ?) を含む論理式に対して，LL2LLP の手法が有効であることが確認できる．これは，Forum の体系で様相演算子の Contraction に対する非決定性が軽減されていることによる効果と考えられる．

しかし，様相演算子を含まない比較的大きな論理式 (LMSS-2, LMSS-3 など) に対しては，linres の方が LL2LLP より効率的であり，この点を改善することが今後の課題といえる．

6 おわりに

本論文では，古典命題線形論理の論理式に対して，著者らの開発した線形論理型言語上で証明探索を行うシステム LL2LLP について述べた．LL2LLP は，古典命題線形論理式を直観主義線形論理式に変換し，さらに LLP コンパイラを用いて LLP 抽象機械命令にコンパイルしたのち実行することで高速な証明探索を実現している点に特徴がある．

いくつかの線形論理式に対して，既存の線形論理定理証明システムと性能を比較した所，ほとんどの問題でより良い結果が得られ，本論文での手法の有効性を確認できた．

ただし，LL2LLP の手法は，Stickel の PTTP 技術の命題線形論理へのナイーブな実装であり，多くの改善の余地が残されている．

例えば，規則の適用順序の動的な変更，あるいは静的解析 [12] [22] を用いた規則の適用順序の分析等は有効であろう．

最後に，有益な助言をいただいた査読者，ならびに本研究を進める上で協力していただいた谷澤勉氏に感謝いたします．

参考文献

[1] Andreoli, J.-M.: Logic Programming with Focusing Proofs in Linear Logic, *Journal of Logic and Computation*, Vol. 2,No. 3(1992), pp. 297–347.

[2] Banbara, M. and Tamura, N.: Compiling Re-

sources in a Linear Logic Programming Language, *Proc. of Post-JICSLP'98 Workshop on Parallelism and Implementation Technology for Logic Programming Languages*, June 1998, pp. 32–45.

[3] Bozzano, M. and Delzanno, G.: Automated protocol verification in linear logic, *Proc. of the 4th international Conf. on Principles and practice of declarative programminge*, 2002, pp. 38–49.

[4] Cervesato, I., Hodas, J. S., and Pfenning, F.: Efficient resource management for linear logic proof search, *Proc. of the 1996 Int'l Workshop on Extensions of Logic Programming*, Springer-Verlag LNAI 1050, March 1996, pp. 67–81.

[5] Girard, J.-Y.: Linear Logic, *Theoretical Computer Science*, Vol. 50(1987), pp. 1–102.

[6] Hasebe, K., Jouannaud, J.-P., Kremer, A., Okada, M., and Zumkeller, R.: Formal Verification of Dynamic Real-Time State-Transition Systems Using Linear Logic, **日本ソフトウェア科学会第 *20* 回全国大会講演論文集**, 9 月 2003.

[7] Hodas, J., López, P., Polakow, J., Stoilova, L., and Pimentel, E.: A tag-frame system of resource management for proof search in linear-logic programming, *Annual Conf. of the European Association for Computer Science Logic*, 2002, pp. 167–182.

[8] Hodas, J. S. and Miller, D.: Logic Programming in a Fragment of Intuitionistic Linear Logic, *Information and Computation*, Vol. 110,No. 2(1994), pp. 327–365.

[9] Hodas, J. S. and Polakow, J.: Forum as a Logic Programming Language: Preliminary Results and Observations, *Proc. of the Linear Logic '96 Meeting*, Vol. 3, Tokyo, Japan, Elsevier Electronic Notes in Theoretical Computer Science, 1996.

[10] Hodas, J. S. and Tamura, N.: LolliCoP – a linear logic implementation of a lean connection-method theorem prover for first-order classical logic, *Proc. of the Int'l Joint Conf. on Automated Reasoning 2001*, Springer-Verlag LNCS 2083, June 2001, pp. 670–684.

[11] Hodas, J. S., Watkins, K., Tamura, N., and Kang, K.-S.: Efficient Implementation of a Linear Logic Programming Language, *Proc. of the 1998 Joint Int'l Conf. and Symp. on Logic Programming*, June 1998, pp. 145–159.

[12] Kang, K.-S. and Tamura, N.: A Static Analysis Method for Classical Linear Logic Programming Language, *Electronic Notes in Theoretical Computer Science*, Vol. 30,No. 3(2000).

[13] Kanovich, M. I., Okada, M., and Scedrov, A.: Specifying Real-Time Finite-State Systems in Linear Logic, *Electronic Notes on Theoretial Commputer Science*, Vol. 16,No. 1(1998).

[14] Mantel, H. and Otten, J.: linTAP: A Tableau Prover for Linear Logic, *Proc. of the Int'l Conf. on Automated Reasoning with Analytic Tableaux and Related Methods*, Springer-Verlag LNAI 1617, June 1999, pp. 217–231.

[15] Miller, D.: Overview of linear logic programming. Submitted as a chapter for a book on linear logic, Ehrhard, Press.

[16] Miller, D.: Forum: A Multiple-Conclusion Specification Language, *Theoretical Computer Science*, Vol. 165,No. 1(1996), pp. 201–232.

[17] Stickel, M.: A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler, *Journal of Automated Reasoning*, Vol. 4(1988), pp. 353–380.

[18] Tammet, T.: Proof Strategies in Linear Logic, *Journal of Automated Reasoning*, Vol. 12(1994), pp. 273–304.

[19] Tamura, N. and Kaneda, Y.: Extension of WAM for a linear logic programming language, *Second Fuji Int'l Workshop on Functional and Logic Programming*, World Scientific, Nov. 1996, pp. 33–50.

[20] 田村直之, 番原睦則: LLPTTP: 線形論理型言語コンパイラ処理系を用いた定理証明システム, **コンピュータソフトウェア**, Vol. 20,No. 5(2003), pp. 90–96.

[21] 番原睦則, 姜京順, 田村直之: 線形論理型言語のコンパイラ処理系のための抽象機械について, **コンピュータソフトウェア**, Vol. 18,No. 1(2001), pp. 39–60.

[22] 姜京順, 番原睦則, 田村直之: 古典線形論理型プログラミング言語の静的解析の一手法について, **情報処理学会論文誌*:* プログラミング**, Vol. 41,No. SIG 4 (PRO 7)(2000), pp. 42–55.

[23] 姜京順, 番原睦則, 田村直之: 線形論理型言語の効率的なリソース管理モデル, **コンピュータソフトウェア**, Vol. 18,No. 0(2001), pp. 138–154.