線形論理と 並列 LISP 言語との同型対応

2 P − 4

筑後 友香　中西 正和
慶應義塾大学大学院 理工学研究科 計算機科学専攻

1. はじめに

直観主義論理と λ 計算との間には，論理式と型，証明と項を対応づける Curry-Howard 同型と呼ばれる対応がある [7]．これはプログラム合成の分野に応用することができる．一方，1987 年 Girard[6] によって提案された線形論理は，数量の概念を含み，論理式による並行な動作の記述や資源に関する記述が可能であるという点で様々な分野から注目されている．そのなかで，従来の Curry-Howard 同型の概念を線形論理に対して導入し [2]，線形 λ 計算を構成する試みが行われている [1]．本研究では，線形に拡張した Curry-Howard 同型を実際に用いて並列なプログラムの合成を行うことを試みる．

2. 型付き λ 計算と直観主義論理の Curry-Howard 同型

直観主義論理と型の概念を導入した λ 計算との間に同型対応をとることができる．

2.1 直観主義論理と自然演繹

Heyting は，論理式 A が証明可能であるとは，A が成り立つ具体的な証拠が必ず見つかることであると考えた [4]．この解釈に基づく証明を構成的な証明という．

自然演繹は，各論理記号に関する導入と除去の規則を持ち，構成的な証明を行う．例えば，論理式 $A \to B$ を証明するには，前提として論理式 A を仮定し，そこから論理式 B を導かなければならない．構成的な解釈では，$A \vee \neg A$ という論理式は A か $\neg A$ かのどちらか一方が証明可能でなければならず，必ずしも正しいとは限らない．従って，$A \vee \neg A$ の成り立たない直観主義論理に対して，自然演繹による構成的な証明は有効である．

2.2 Curry-Howard 同型

λ 式に型を導入することによって，論理式は型，証明は関数型プログラム，証明の正規化は項計算とみなすことができる [7]．型の構文は例えば次の通りに定義することができる [5]．

- 原始型 $T_1, \cdots, T_n$ は型である．
- U と V が型のとき，$U \wedge V$ と $U \to V$ は型である．
- 以上のものだけが型である．

具体的には，次のように項割り当てと証明の手続きを一致させることができる．

$$\dfrac{\dfrac{\dfrac{a : A \qquad b : B}{a \wedge b : A \wedge B}}{\lambda a.a \wedge b : A \to A \wedge B}}{\lambda b.\lambda a.a \wedge b : B \to (A \to A \wedge B)}$$

3. 線形 λ 計算と直観主義線形論理の Curry-Howard 同型

線形論理には，乗法的演算子 ($\otimes$, $\parr$) と加法的演算子 ($\&$, $\oplus$) の異なる意味をもつ演算子が存在し，従来の論理では扱うことができなかった数量の概念を含んだ命題や，同時性をもった命題を扱うことができる [3]．また，並行プロセスの記述が可能で，プロセス代数の CCS と対応づけることができる．

3.1 直観主義線形論理と自然演繹

直観主義線形論理は，従来の直観主義論理を，各論理式が 1 度しか使われてはならないように改良したものである [2]．直観主義論理の証明の中では，仮定した論理式を何度でも必要なだけ用いることが許されていたが，直観主義線形論理では仮定を 1 度しか用いることができない．このことによって制限された記述

Isomorphism between linear logic and parallel Lisp

Yuka CHIKUGO Masakazu NAKANISHI

Department of Computer Science, Faculty of Science and Technology, Keio University 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan

力を補う論理記号として！が用意されている．論理式 $!A$ は，論理式 A を必要な回数だけ複製することを許す演算子である．線形な場合の自然演繹も，自然演繹と同様に導入と除去の推論規則をもつ．

3.2 線形な場合の Curry-Howard 同型

直観主義線形論理に対して Curry-Howard 同型を導入し，線形λ計算を構成することができる [1]．直観主義線形論理の推論規則は全て，線形λ計算の型付けの推論規則となる．型は例えば次のように定義される [1]．

- 原始型 $T_1, \cdots, T_n$ は型である．
- U と V が型のとき，$U \otimes V$, $U \multimap V$, $!U$ は型である．
- 以上のものだけが型である．

具体的には，次のように項割り当てと証明の手続きを一致させることができる．

$$\dfrac{\dfrac{\dfrac{a : A \quad b : B}{a \otimes b : A \otimes B}}{\lambda a.a \otimes b : A \multimap A \otimes B}}{\lambda b.\lambda a.a \otimes b : B \multimap (A \multimap A \otimes B)}$$

4. 本研究の手法

本研究では，Curry-Howard 同型を用いて Lisp のプログラムから並列 Lisp のプログラムを合成する．そして，Curry-Howard 同型を用いたプログラム合成がどの程度有効であるか検討する．

直観主義論理とλ計算の間に Curry-Howard 同型がとれることから，λ計算によって意味づけされる Lisp のプログラムと直観主義論理との間に対応をとることが可能である．同様に直観主義線形論理と並列 Lisp の間にも同型対応をつけることが可能であり，Lisp プログラムから並列 Lisp プログラムを合成する．

4.1 Lisp の式と 直観主義論理の論理式の対応

Lisp の S 式のうち `if` や `lambda` で表されるプログラムについて，操作的意味論を次のように直観主義論理の自然演繹の推論規則に対応づける．

$$\dfrac{u : A}{(\texttt{if}\ A\ u\ v) : A \vee \neg A} \qquad \dfrac{v : \neg A}{(\texttt{if}\ A\ u\ v) : A \vee \neg A}$$

$$\begin{array}{c}[x : A] \\ \cdots \\ u : B\end{array}$$

$$\dfrac{}{(\texttt{lambda}\ x.u) : A \to B} \qquad \dfrac{x : A \quad f : A \to B}{(f\ x) : B}$$

4.2 並列 Lisp の式と 直観主義線形論理の論理式の対応

Lisp の S 式である `if` や `lambda`, `furture` で表されるプログラムの操作的意味論を，次のように直観主義線形論理の自然演繹の推論規則に対応づける．

$$\dfrac{u : A}{(\texttt{if}\ A\ u\ v) : A \oplus A^{\perp}} \qquad \dfrac{v : A^{\perp}}{(\texttt{if}\ A\ u\ v) : A \oplus A^{\perp}}$$

$$\begin{array}{c}[x : A] \\ \cdots \\ u : B\end{array}$$

$$\dfrac{}{(\texttt{lambda}\ x.u) : A \multimap B} \qquad \dfrac{x : A \quad f : A \multimap B}{(f\ x) : B}$$

$$\dfrac{u : A \quad v : B}{(\texttt{future}\ u)\ \textit{in}\ v : A \otimes B}$$

5. おわりに

現在，提案手法について実装，及び実験を行っている．結果については発表時に報告を行う．

参考文献

[1] T. Braüner, A general adequacy result for a linear functional language, *Theoretical Computer Science*, 177, pp. 27–58, 1997.

[2] N. Benton, G. Bierman, B. Paiva and M. Hyland, A Term Calculus for Intuitionistic Linear Logic, *International conference. TLCA*, pp. 75–90, 1993.

[3] S. Abramsky, Proofs as processes, *Theoretical Computer Science*, 135, pp. 5–9, 1994.

[4] J.-Y. Girard, Y. Lafont, and P. Taylor, *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Science*. Cambridge University Press, 1989.

[5] H. P. Barendregt, Functional programming and lambda calculus, *Handbook of Theoretical Computer Science* (ed. J. van Leeuwen) Volume B, Elsevier, pp. 320–363, 1990.

[6] J.-Y. Girard, Linear Logic, *Theoretical Computer Science*, 50, pp. 1–102, 1987.

[7] W. A. Howard, The formulae-as-type notion of construction, in: *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, New York, pp. 479–490, 1980.

[8] D. Albrecht, J. N. Crossley, J. S. Jeavons, New Curry-Howard terms for full linear logic, *Theoretical Computer Science*, 185, pp. 217–235, 1997.