談話室

Call by Need 再　考†

外　山　芳　人††

リダクションシステムでは与えられた対象（項，記号列，グラフ等）を，書き換え規則にもとづいて次々と書き換える（すなわち**リダクション**する）ことによって計算過程を実現している[10]．**項書き換えシステム**[10]，**ラムダ計算**[1]，**コンビネータシステム**[8]などは良く知られたリダクションシステムである．リダクションシステムでは書き換え可能な部分（**リデックス**という）をどのような順序で書き換えて行くかが重要な問題となる．なぜなら，答え（**正規形**）がリダクションによって得られるか否か，あるいは答えに何回の書き換えで到達できるか，といった問題は書き換え順序の選び方と密接に関係しているからである．

書き換え順序を定める規則は**リダクションの戦略**と呼ばれる．リダクションシステムを計算システムと見なした場合，その特徴の多くはリダクションの戦略によって決定される．また逆に，さまざまな計算システムを，リダクションの戦略として特徴づけることも可能である．

たとえば，プログラミング言語を計算機上で実現する際，手続き文のパラメータ引き渡し方法として，**名前よび**（**call by name**），**値よび**（**call by value**），**番地よび**（**call by reference**）などが知られているが，このうち**名前よび**と**値よび**は，それぞれ**最左最外リダクション**（**leftmost outermost reduction**：最左最外リデックスを常に書き換える），**最左最内リダクション**（**leftmost innermost reduction**：最左最内リデックスを常に書き換える）という戦略として，リダクションシステムの中でモデル化されることは広く知られている．（この対応から，それぞれのリダクションの戦略を，**名前よび**，**値よび**という場合もある[10]．）また，データフロー計算機における**データ駆動**（**data driven**）型計算は**最内リダクション**（内側のリデックスから常に書き換える）によって特徴づけられるし，**要求駆動**（**demand driven**）型計算は**最外リダクション**の変形と見なすことができる[5]．

最近**必須よび**（**call by need**）と呼ばれる戦略がしばしば話題となっている[1),4),6),10]．必須よびという名称は，名前よびや値よびのように手続き文のパラメータ引き渡し方式との対応からきた名称ではなく，最初からリダクションの戦略の意味で用いられた[1),4),6]．必須よびのねらいを一言で述べるなら，無駄な書き換えはできるだけ省き，本当に必要な部分の書き換えのみを行うことで効率の良いリダクションを実現することである．**遅延評価**（**lazy evaluation**）[3]や**要求駆動**[5]といった類似の概念が活発に研究されているのも，このような戦略の重要さが広く認められているからである．

ところで，無駄な計算をしない戦略というのは，はなはだ不明確な概念である．あるシステムにおいて効率の良い戦略であっても，他のシステムにおいては効率の悪い戦略となる場合もあるからだ．したがって，必須よびという術語は，用いられる状況が異なると，異なる戦略を指すこともあり，誤解の生ずる原因となりやすい．著者は以前に本誌の解説[10]の中で，必須よびについて簡単にふれたが，誌面の制約等で十分な説明ができなかった．ここでは再度必須よびの意味を整理し，混乱を防ぐための私案を述べてみたい．

必須よびという術語は，次の（1），（2）のどちらかの意味で用いられることが多い．

（1）　リダクションの各ステップで最左最外リデックスを常に書き換える戦略[1]．ただし，書き換えの際に必要となる部分項のコピーは計算コストを増加させるので行わず，コピーするかわりにポインタによってひとつの部分項を共有する（**コピー無しリダクション**：**noncopying reduction**）．この戦略は**最左最外コピー無しリダクション**（**leftmost outermost noncopying reduction**）とも呼ばれる[10]．さらに**delay rule**[9]，**strictly leftmost outermost noncopying sequence**[7]，**normal graph reduction**[8]と呼ばれる場合もある．

† A Comment on Call by Need by Yoshihito TOYAMA (Musashino Electrical Communication Laboratory, N. T. T.).

†† 日本電信電話公社武蔵野電気通信研究所

(2) リダクションの各ステップで，**必須リデックス (needed redex)** を常に書き換える戦略[4),6),10)]．必須リデックスとは，正規形を得るためには必ず書き換えねばならないリデックスのことである．(1)と異なり，書き換えるべきリデックスのシンタックス上での位置は固定されていないことに注意する．書き換えられるリデックスは，必須リデックスの出現位置に依存してリダクションの各ステップごとに定められる．

最左最外リデックスが常に必須リデックスとなるリダクションシステムのクラスに対しては，(1)の戦略は極めて有効であることが知られている．たとえば，**再帰的プログラム**[9)]や**コンビネータ論理**では，(1)は最適戦略となっており，最小の書き換え回数で正規形が得られる[10)]．また，**ラムダ計算**[1)]や**関数的プログラム**[8)]の計算方法としても，最適に近い効率の良い戦略となる．しかし，(1)の欠点は，最左最外リデックスが必須リデックスとならないリダクションシステムの場合には，効率の良い戦略でないばかりか，正規形が得られるかどうかも保証されていないことである．

一方(2)は，特定のシステムに限らず，どのようなリダクションシステムに対しても，正規形が存在するなら必ず得られることが保証されている[4),6)]．(1)のように特定のシステムに依存しないので，個々のシステムのシンタックス上の構造を離れた抽象的なレベルでの戦略を論ずる場合に有用な概念である．しかし，必須リデックスが簡単な手続きで求まる場合以外は，実際的な戦略として用いることは難しい．

以上のように必須よびという術語は，全く異なる2つの意味で用いられる．たとえば，『必須よびにより，計算コストを最小にできる.』といった場合には，必須よびは(1)の意味で用いられているし，『どのようなシステムでも，必須よびによって必ず答えが得られる.』といった場合には(2)の意味になる．しかし，時には，どちらの意味で解釈しても不都合が生じないこともある*．そのような状況では，この術語の意味が混乱してくることは避けがたい．

必須よびに限らず，リダクションの戦略を示す術語には混乱の生じやすいものが多い．これらの混乱をできる限り少なくするひとつの方法は，良く知られているように，書き換えるべきリデックスの位置を戦略の名称とすることである[10)]．たとえば，誤解を生みやすい名前よび，値よびといった戦略名はできるだけ使用せず，**最左最外リダクション**，**最左最内リダクション**を用いる**．また，必須よびを(1)の意味で用いる場合には，**最左最外コピー無しリダクション**という術語を用いる．一方，(2)の戦略では書き換えられるリデックスの位置が固定されていないから，その位置を名称とすることができない．したがって，著者としては**必須よび**を(2)の意味にのみ使用することを提案したい．

必須よび以外にも，リダクションの戦略に関連したさまざまな概念が知られている．(たとえば，遅延評価[3)]，要求駆動[5)]，部分計算 (partial evaluation)[2)]，等) これらについても，ここで試みたような整理を行うことが必要であると感じている．それは，概念のあいまいさから生ずる混乱や誤解を防ぐという消極的な理由だけではない．むしろ，リダクションの戦略というひとつの枠組みを通してこれらの概念を見直すことにより，これまであいまいだったさまざまな計算システムの相互関係を共通の土台の上で議論することが可能となり，これらの領域に関する我々の理解が一層深まると著者は信ずるからである．

謝辞 「必須よびとは何か」という議論によって本稿執筆のきっかけを与えていただき，また原稿についても適切なご意見をいただいた正田輝雄氏（東京都立大）に深謝します．

* コピー無しリダクションシステムで，必須リデックスが常に最左最外リデックスとなっている場合には，(1)と(2)の戦略は一致する[6)]．

** 名前よび，値よびは，最外リダクション，最内リダクションの意味で用いられる場合がある．そのような状況では名前よび，値よびという術語は“最左”が付くか否かが，あいまいである．

参 考 文 献

1) Aiello, L. and Prini, G.: An Efficient Interpreter for the Lambda-calculus, J. Compt. Syst. Sci. 23, pp. 383-424 (1981).
2) Futamura, Y.: Partial Computation of Programs, Lecture Notes in Comput. Sci. 147, Springer-Verlag, pp. 1-35 (1982).
3) Henderson, P. and Morris, J.-H.: A Lazy Evaluator, ACM Sympo., Principle of Programming Languages, pp. 95-103 (1976).
4) Huet, G. and Levy, J.-J.: Call by Need Computations in Nonambiguous Linear Term Rewriting Systems, Rapport Laboria 359, IRIA (1979).
5) Keller, R. M., Lindstrom, G. and Patil, S.: A Loosely-coupled Applicative Multi-processing System, Proc. AFIPS, pp. 613-623 (1979).
6) Lévy, J.-J.: Optimal Reductions in the Lambda-calculus, To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Eds. Seldin, J. P. and Hindley, J. R.,

Academic Press, pp. 159-191 (1980).
7) O' Donnell, M.: Computing in Systems Described by Equations, Lecture Notes in Comput. Sci. 58, Springer-Verlag (1977).
8) Turner, D. A.: A New Implementation Technique for Applicative Languages, Softw. Prac. Exper. 9, pp. 31-49 (1979).
9) Vuillemin, J.: Correct and Optimal Implementations of Recursion in a Simple Programming Language, J. Comput. Syst. Sci. 9-3, pp. 332-354 (1974).
10) 二木, 外山: 項書き換え型計算モデルとその応用, 情報処理, Vol. 24, No. 2, pp. 133-146 (1983).

(昭和58年6月14日受付)