KL1 の要求駆動型実行方式

4 E − 4

宇佐 治彦, 近山 隆
東京大学 工学系研究科*

1 はじめに

並列論理型言語 KL1 は各ゴールの実行順が任意であるため, 様々なスケジューリングが可能である. 例えば, 手続き駆動 (PIM, KLIC など), データ駆動 (上田 [1]), スレッド単位のスケジューリング (Tick, 中島など [2]) などが提案・実装されている. 一方, LISP や関数型言語では遅延評価を拡張した形で並列実行が行なわれることが多い. 遅延評価では, データが必要になった時点で計算が開始されるため,

- **最終結果に寄与しない不要な計算を除去できる.**
- **必要なデータを直ちに生成することにより, 並列処理においてデータ要求に対するレスポンスが向上する.**

といった利点があるが, 並列論理型言語においては関数型と異なり要求が明示的でないことなどから, 要求駆動によるスケジューリング方式はあまり試みられていない. 本研究では, コンパイル時に変数の値を生成・利用する関係を解析し, 実行中に変数の値が必要になったときに, その値を決定するゴールを実行する要求駆動型のスケジューリング方式を設計し, 実装・評価を行なった.

2 単純な要求駆動型実行

ある変数の値が必要になったときに, その値を具体化するゴールを実行するのが要求駆動型のスケジューリングである. 例えば,

```
p(X,Y) :- a(X,A), q(A,Y).
a(X,A) :- b(X,B), c(B,C), d(C,A).
```

という節が定義されており, それぞれのゴールの第1引数が入力, 第2引数が出力であるとき, あるゴール g(Y,Z) を実行するときに Y の値が必要であった場合は, Y を具体化するために 図 1 に示したように実行される.

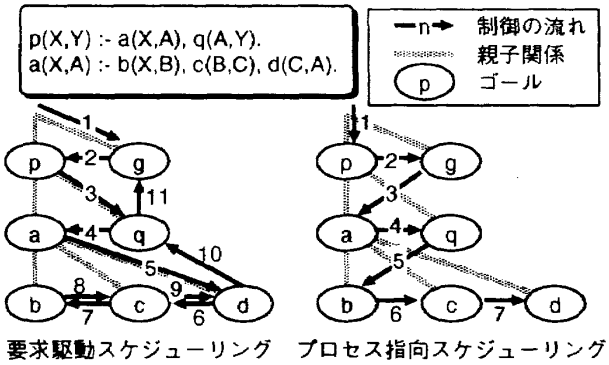

図 1: 要求駆動とプロセス指向実行の比較

3 KL1 と KLIC の拡張

KL1 で要求駆動ゴールを扱うためには,

> **ある変数 X を具体化するゴール *Goal* の評価を遅延し, X の具体値が必要になったときに *Goal* を実行する**

という機構が必要である. このために, 以下の拡張を行なった.

KL1 の拡張: $Goal$@demand($[X_0, X_1, ..., X_{n-1}]$) という形式のプラグマを追加する. これにより $X_0, X_1, ..., X_{n-1}$ のいずれか 1 つの値が必要なときに *Goal* を実行することを指示する.

KL1 処理系の拡張: X を通常の変数とは異なる特殊形式の変数として, *Goal* に関連付けておき, X が参照されたときに X は通常の変数に戻してから *Goal* を実行する.

この拡張により, ユーザはプラグマを付加して適切な要求駆動ゴールを指定できるようになった.
要求駆動ゴールは, その出力データが必要なくなれば実行されない. これは論理的には, まだ証明されていない定理を証明済であるとすることであり, **健全でない実行を許す非論理的な性質** である. しかし, KL1 では論理的性質よりもデータの入出力関係に注目してプログラミングをすることが多いため, 大きな問題とはならない. それ以上に, **余分な計算を省ける可能性があることは大きな利点である.**

*Demand Driven Scheduling of KL1 Programs
Haruhiko Usa and Takashi Chikayama
{usa, chikayama}@logos.t.u-tokyo.ac.jp
School of Engineering, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, Japan

4　要求駆動実行ゴールの決定

モード解析を行ない，各データを生成するゴールが判明すれば，単純な要求駆動型実行は可能である．しかし，図1のように実行中断が多発するため，なんらかの基準でデータの必要度を解析し，すぐに必要な計算は遅延しない必要がある．KL1ではガード単一化と組み込み述語の実行のときに変数の具体値が必要になる．従って，抽象実行などによって，ゴールが生成するデータの消費ゴールを追跡すると，その必要度を解析できると考えられる．しかし，次のような簡単な例でも，その解析は困難である．

```
1:  main(L,R) :- foo(L,0,R).
2:  foo([],   C0,C) :- C=C0.
3:  foo([N|L],C0,C) :-
        bar(N,C0,C1), foo(L,C1,C).
```

foo/3は再帰的にリストLに対しbar/3の処理を行うが，bar/3の生成データを参照しなくてもfoo/3は実行を終了できるので，**bar/3は要求駆動で実行する方が良いと判断できる**．しかし，bar/3の処理が簡単に終わるような場合は実行時間・消費メモリとも数十倍以上大きくなってしまう．従って，極端に性能が低下することを防ぐためには，この例では**bar/3は遅延せずに実行すると判定する方が良い**．このように，データの必要度を判定するのは非常に困難である．従って，本研究では，「しばらく必要ない可能性の高いデータを生成するゴール」を遅延評価する，という方針をとった．要求駆動ゴールを決定するアルゴリズムは以下の通りである．

- **生成するデータが他のゴールの入力引数になっている場合は，そのゴールは遅延しない**
- **生成するデータが構造体の内部に格納される場合は，要求駆動ゴール候補にする**
- **あるゴールが要求駆動で実行される場合は，その子ゴールも実質的に要求駆動になっているので，要求駆動ゴール候補の子ゴールは自己再帰を除いて，遅延評価しない**

5　実装と評価

前節のアルゴリズムに基づいて，要求駆動型実行用コードを生成するコンパイラをKLIC 3.002をもとに実装した．

gencon, handshake, hanoi, n-queen, primesの5種類の問題について，

original: プラグマなし，原則としてプロセス指向スケジューリングではサスペンドしない．

priority: 一部のゴールを低優先度で実行．プラグマは適切と思われる箇所に人手で加えた．

demand: original版を本処理系でコンパイル．

という3通りの方式で実行したときの実行時間（秒）とヒープサイズ（KW）を測定し，結果を図2にまとめた．この結果，**要求駆動型実行によって，実行時間はオリジナルの2.5倍以内，消費メモリは問題サイズによらず一定**という性能が達成された．

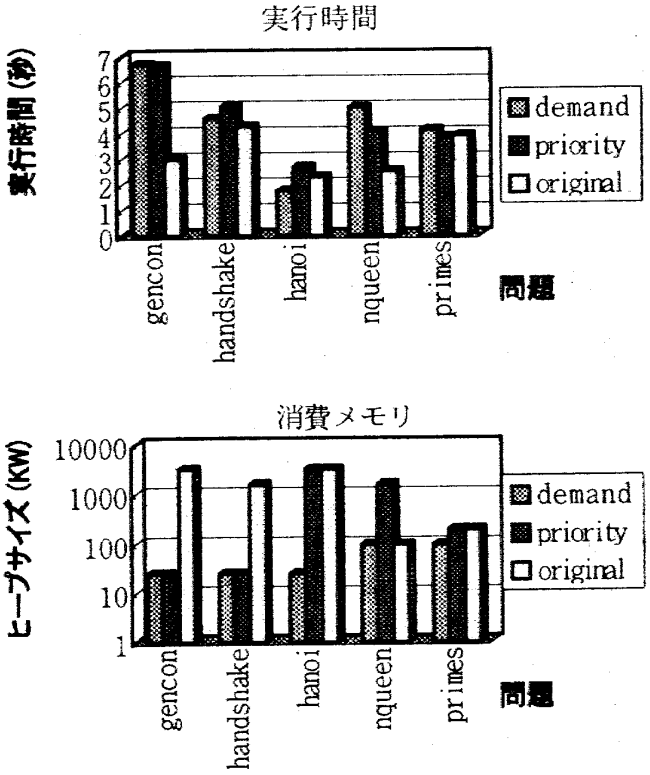

図2: 相対実行時間と消費メモリサイズ

6　おわりに

提案したアルゴリズムによって十分効率良く要求駆動型実行が可能であることが確認された．今後は，要求駆動ゴールを判定するために，さらに精度の高い解析を行い，様々な応用プログラムを対象にして，評価を行う必要がある．

参考文献

[1] K. Ueda, M. Morita: A New Implementation Technique for Flat GHC. In *Proc. Seventh Int. Conf. on Logic Programming*, pp. 3–17, The MIT Press, 1990.

[2] H. Nakashima: An Optimization Technique of Efficient Goal Scheduling for KL1 Programs. *http://www.icot.or.jp/*, 1996 and 1997.