GHCサブセット逐次型処理系のデバッグ環境

6D-4

江崎令子'　宮崎敏彦"　太細孝'
'三菱電機(株)情報電子研究所　"(財)新世代コンピュータ技術開発機構

1. はじめに

並列プログラムのデバッギングは、逐次プログラムのデバッギングと比較すると、かなり困難である。そのため、並列プログラムのためのデバッギング・ツールの開発は、並列型言語の普及にともない、非常に重要な課題となると考えられる。

本稿では、先に報告した並列論理型言語FlatGHC の逐次型処理系［江崎86］におけるデバッグ環境、特に、Process Oriented Tracer について報告する。

このトレーサの特徴としては、

① Boxモデル［Clocksin81］を取り入れ、 Boxに対応するものとして、従来のPrologのトレーサのように手続き(procedure)だけでなく、プロセス(process)を取り挙げている。

② GHCの実行過程をプロセスと見なしている。

等がある。以下では、これらの特徴に基づき、Process Oriented Tracer の設計方針と概要について述べる。

2. GHCのプログラミング・スタイル

並列論理型言語 GHCでプログラムを記述するに当たって、“ユーザの定義する実行の単位”という概念を取り入れたプログラミング・スタイルを考えることができる。この単位をプロセスと呼んでいる。

本来、 GHCにおいては、プロセスという概念は用いられていない［Ueda85］。しかし、実際にプログラムの設計を行っていく上で、“実行の単位”という考える枠組みを取り入れることは、非常に有用である。

以下に示す、「エラトステネスのふるい」のプログラム(prime generator)の例を考えてみよう。

```
primes(N):- true  | generate(2,N,A),
                    sift(A,B),
                    outstream(B).
```

図1 prime generator

ここで、generate、sift、outstream を、それぞれ別のプロセスと考えることは自然である。primesをよびだすと、これらのプロセスは、共有変数A、Bを介して、通信を行うと考えることができる。(図2)

このような見方を取り入れることにより、GHC のプログラムは、プロセスとそれを結ぶ通信路によって構成されたネットワークと見なすことができる。このことは、並列プログラムの設計を行う場合に助けとなる指針を、プログラミング・スタイルとして与えたものと考えることができる。

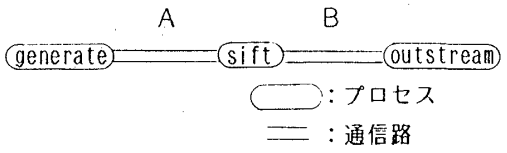

図2 プロセス・ネットワーク

3. 並列プログラムのデバッガ

並列プログラムのデバッギングが、逐次プログラムに較べて困難な理由として、実行が半順序に(すなわち、人間にとっては交錯して)行われるため理解しにくい、ということが挙げられる。この問題を意識せずに並列プログラムをデバッグするためのツールとしては、アルゴリズミック・デバッガ［竹内86］が考えられる。

アルゴリズミック・デバッガは、バグ検出アルゴリズム“Divide and Query”により、プログラムの仕様に対する必要最小限の情報を受けて、デバッガ自身がバグの位置を同定するものである。アルゴリズミック・デバッガを用いると、プログラムの流れを理解していなくても、入出力の関係に基づいてデバッグすることが可能であるため、ユーザの負担も軽く、並列プログラムのデバッギングの持つ問題を、ある意味では解決したことになる。しかし、アルゴリズミック・デバッガにおいては、プログラムの実行がブラックボックスとなっているため、プログラム全体の流れがわかりにくく、プログラムの動作を理解する助けになるとは言い難い。そこで、プログラムの流れをより良く理解する助けとなるデバッギング・ツールとして Boxモデルに基づいたトレーサを考えることにする。

Boxモデルに基づいたPrologのトレーサは、手続きの呼び出しを Boxで表し、この Boxに対する制御の流れの情報を四個の端子で表すもので、プログラムの動作を理解することを容易にする。しかし、これをそのまま並列プログラム(ここでは GHC)に適用したのでは、並列計算に対する情報が交錯して、かえってわかりにくくなる可能性がある。そこで、前節で述べたプロセスの概念を取り入れた、Process Oriented Tracer を考えることにした。

Process Oriented Tracer for FlatGHC System
Reiko EZAKI', Toshihiko MIYAZAKI", Takashi DASAI'
'MITSUBISHI ELECTRIC Corp., "ICOT

4. Process Oriented Tracer

FGHC処理系に置けるProcess Oriented Tracer は、次のような特徴を持っている。

① Boxモデルを GHC向きに変更し、Prolog的なトレース機能を実現している。

②プロセスの概念を用い、ユーザの考える通りにプロセスを定義することができる。

以下に、この二点の概要を述べる。

（１）GHC 向き Boxモデル

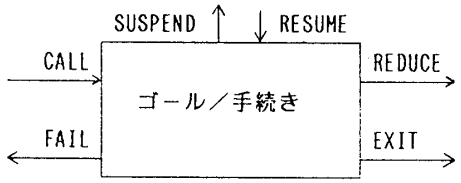

図３　FGHC向き Boxモデル

Boxはゴール／手続きの呼び出しに相当し、各端子は、それぞれ以下のような制御情報を示す。

CALL……　手続きの呼び出し。

REDUCE……　新たなゴールの列に展開された。

FAIL……　呼び出しに対応するすべての候補節が失敗した。

EXIT……　呼び出されたゴールの実行が終了した。

SUSPEND …　ゴールの実行が中断した。

RESUME……　いったん中断したゴールがスケジュールされた。

これらの制御情報により、FGHC処理系における実行の状態をすべて表すことが可能である。

（２）プロセス

Process Oriented Tracer におけるプロセスは、ユーザの考える実行単位であり、GHC のプログラミング・スタイルと同じであっても良いし、実行トレースの過程で、異なったものとすることもできる。ユーザに対しては、ゴールの呼出し時にプロセスとして定義を行う手段を動的に提供しているので、ユーザは、トレース時に、自分の考えるプログラムのプロセス構造をそのまま反映することが可能である。この様なプロセスの概念を取り入れることにより、ユーザはトレース時に、自分のみたいプロセスにだけ注目することが可能となる。

以上の要素を取り入れ、プロセスのウィンドウへのアサイン、プロセス単位のスパイ、などの機能を加えて作成したのが Process Oriented Tracerである。その実行画面イメージを図４に示す。

5. まとめと課題

ユーザのプログラムをプロセス単位に扱うことの出来る Process Oriented Tracerを作成した。これにより、逐次プログラムのデバッグを行う場合と同様の感覚で並列プログラムのデバッグを行うことができる。

今後の課題としては、interactive なユーザのプロセス定義なしにプロセスを分離することができるよう、静的にプロセス構造の定義が行える記述言語のようなものを考える必要があると思われる。

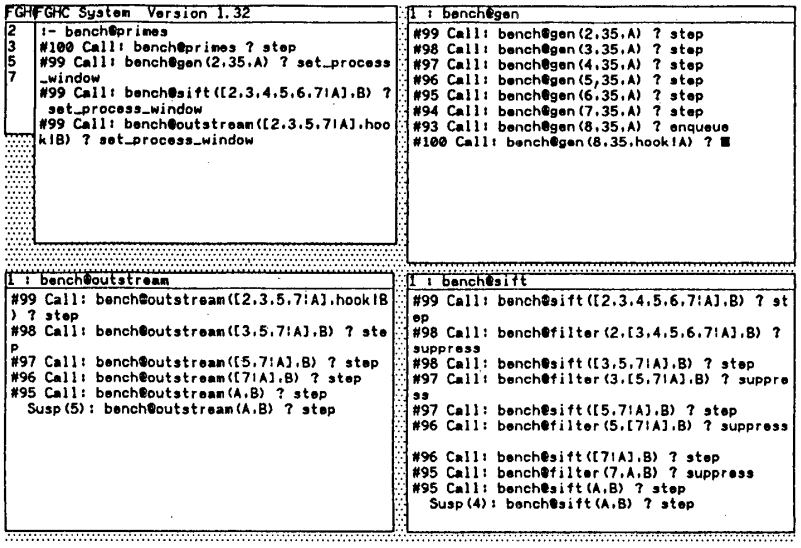

図４　Process Oriented Tracer

＜参考文献＞

[Ueda85] Kazunori,Ueda:
"Guarded Horn Clauses",
ICOT Technical Report 1985.

[Clocksin81] Clocksin,W.F.,Melish,C.S
"Programming in Prolog",
Springer-Verlag 1981.

[竹内86]:竹内彰一：
“GHC プログラムのアルゴリズミック・デバッギングについて”，
第３２回情処全国大会，no.2G-5.

[江崎86]:江崎他；
“ GHCサブセット逐次型処理系の作成”，
第３２回情処全国大会，no.2G-4.

[田村86]:田村他；
“並列論理型言語P-PROLOGのデバッガの設計”，
第３２回情処全国大会，no.2G-2.