

並列論理型言語 Fleng のデバッガ HyperDEBU のトップレベルウインドウ

2K-3

館村 純一, 小池 汎平, 田中 英彦

{tatemura,koike,tanaka}@mtl.t.u-tokyo.ac.jp

東京大学工学部*

1 はじめに

並列プログラムのデバッグは逐次プログラムと比べて極めて困難である。これは、複数のプロセスが互いに干渉しながら同時に動作しているため、実行の理解・制御が困難であることが原因であり、その動作にしばしば非決定性が含まれることが更にデバッグを困難にしている。デバッガはプログラムの実行をモデル化しユーザはそれを通してプログラムを観察・制御するものであると考えられるから、この問題を解決するには、細粒度並列プログラムの実行を表現するのに適したモデルを導入しなければならない。

我々の研究室で開発された並列論理型言語 Fleng は、GHC などの Committed-Choice 型言語の一種であり、並列実行の単位がゴールという細粒度なものとなっている。我々は Fleng プログラムの実行を互いに通信し合うプロセスとしてモデル化し、これに基づいたマルチウインドウデバッガ HyperDEBU を開発した [1]。

HyperDEBU には、プログラムの実行を視覚的に取り扱うためのグローバルなビューとしてトップレベルウインドウが用意されている。本稿では、このトップレベルウインドウの機能について述べる。

2 HyperDEBU

HyperDEBU は、多次元的情報である並列プログラム実行過程を多次元インタフェースを用いて観察・制御するデバッガである。これは、細粒度並列プログラムの実行過程において制御 / データの流れが形成する複雑なグラフ構造を観察 / 操作するための多様な視野としてウインドウを提供する。

HyperDEBU は、(1) トップレベルウインドウ (2) プロセスウインドウ (3) ストラクチャ・ブラウザ / インスペクタから構成されている。

3 HyperDEBU のトップレベルウインドウ

トップレベルウインドウはプログラム (の実行) のグローバルな観察・制御を行なうためのウインドウであり、以下のような機能が提供される。

- ソース・コードからの静的な情報の取り扱い
- ユーザからの付加情報の活用

*The Top-level Window of HyperDEBU : the Debugger for Fleng Programs

Junichi TATEMURA, Hanpei KOIKE, Hidehiko TANAKA,
the University of Tokyo

- プログラムの実行・制御 (初期ゴールの実行)
- 実行 (の全体像) の視覚的表示

これらの機能を用いて、ユーザがデバッグを行なう手順は、

1. ソースから得られる情報を参照しながら、ユーザがデバッガに情報を与える。(HyperDEBU では、この情報のことを「ブレイクポイント」と呼んでいる。)
2. デバッガはこの情報をプログラムの実行制御・視覚化に活用する。
3. デバッガによってウインドウ上に表現された実行過程に対してユーザがインタラクティブに操作を行なう。

というものになる。さらに詳しい実行の観察・制御をする場合は、このウインドウからプロセスウインドウを開く。トップレベルウインドウの役割として、これら各ウインドウの管理をすることもあげられる。

以下では、トップレベルウインドウの機能を実現する上で主要な構成要素となる (1) 視覚化 (2) ブレイクポイント (3) プログラム・コードのブラウジング機能のそれぞれについて述べる。

4 トップレベルウインドウにおけるプログラム実行の視覚化

HyperDEBU では Fleng プログラムの実行を図 1 のようなプロセスとして取り扱っている。

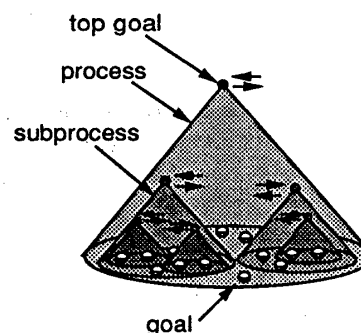


図 1: プロセスの概念図

ここでは、任意の一つのゴール (top goal) と、そのゴールから生成される全てのゴール (subgoals) をまとめて一つのプロセスととらえている。プロセスの内部には、またサブプロセスを考えることができる。

図 2 はトップレベルウインドウにおけるプロセスの表示の例であるが、これは図 1 を上から見たところにあたる。各プロセ

スは、ウインドウの中に表示された矩形によって表現され、矩形に付けられた模様はプロセスの状態を表現する。矩形の中に表示された矩形はプロセスとサブプロセスの関係を表している。この矩形にマウスカーソルを持っていくとゴール表示用のウインドウに該当するトップゴールが表示される。これらの表示は、実行状態を反映して動的に変更され、プロセスの生成や状態変化、トップゴールの引数のデータの変化が把握できる。

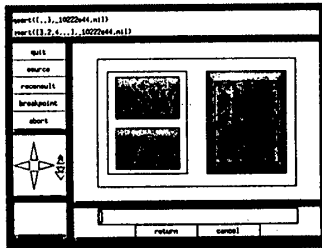


図 2: トップレベルウインドウによるプログラム実行の視覚化

プログラムの実行の視覚化には制御 / データそれぞれに関する視覚化があるが、ここで扱っているものは制御の流れに関する視覚化といえる。一方、データの流れの視覚化は現在ではまだ検討中であるが、ストリーム通信に着目した視覚化を考えている。

Fleng の様な細粒度並列プログラムでは、個々のゴール・個々のデータを全て視覚化したのでは複雑になり過ぎる。そのためトップレベルウインドウでは、視覚化の単位としてゴール単位よりも大きな粒度のプロセスを用いることによりこの問題を解決した。これは、ある特定のゴールについて、それをトップゴールとしたプロセスを表示することで実現されている。また、データの視覚化でも、特定のデータの流れだけを表示することになる。ここで、何が「特定」かはユーザーの主観によるところが大きく、ソースプログラムのみからそれを推測するのは現実的でない。HyperDEBU では、ユーザーがブレークポイントによってこれを指示することにより、ユーザーの意図を反映させることが可能である。

このようにしてかなり高い抽象度で視覚化されたプログラム実行情報について、より詳しい情報を扱う場合は、視覚化されたプロセス (ウインドウ上の矩形) を直接操作してプロセスウインドウを開く。

5 ブレークポイント

並列プログラムでは実行の流れが複数あり、「ブレークポイントで停止して実行状態を見る」というような逐次プログラムのデバッグ手法は適用できない。

デバッグにおける「ブレークポイント」を拡張して考えれば、デバッグがユーザーから実行前に予め与えられた知識ととらえられる。HyperDEBU でブレークポイントと呼んでいるものは、ブレークポイントの「場所」と、そこで処理系が行なう「処理」を組にして指定するものであり、デバッグはこの情報をプログラムの実行制御・視覚化に活用する。ブレークポイントで指定する処理には、(1) そのゴールの実行の停止 (2) そのゴールをトップゴールとしたプロセスの生成 (3) 実行履歴の制御などが

あげられる。

デバッグとは、プログラムの実行情報を持ったデバッガに対してユーザーが情報を与えていくうちにバグのある場所が絞り込まれていく過程と捉えられる。ブレークポイントはこのための情報を予め与えておくものであり、静的な情報やユーザーがプログラムに持たせる意味などをデバッグに活用することができる。このことから、ブレークポイントの指定によるバグへのより早い絞り込みが期待される。また、デバッガが実行時に記憶すべき実行情報の削減の効果もあり、実行メモリ・実行時間の節約に役立つ。

6 プログラム・コードのブラウジング機能

ブレークポイントは実行の前に与えなければならない。そこでその時のユーザーの負担を解消するためにこれをサポートするツールを用意する。

ブレークポイントをつけるには、述語定義だけでなく各述語の呼び出し関係などの把握が必要である。このため、トップレベルウインドウには述語名・述語定義・各述語の相関関係についてのグラフ構造を管理するデータベースとこのグラフ構造をトレースする機能が備わっている。さらにこれを画面にグラフィックに表示する機能も検討中である。また、このデータベースはユーザー入力時の述語名補完機能にも利用されており、ブレークポイントの指定時の効率をさらに良くしている。

7 課題

視覚化については、表示の仕方 (配置法)、特にデータの流れに関する表示の問題があげられる。

ブレークポイントについては、その種類とその付加技法の検討が必要であろう。さらに、ソースから半自動的につけることも期待される。

ブラウザは、現在簡単な試作版が作成されているが、より機能の充実した改良版を作成する予定である。このとき、静的デバッグを支援する機能なども課題である。

トップレベルウインドウの機能としては、他にソースの簡単なエディット機能も必要となる。これは、汎用のエディタというよりは、バグの修正や小さな変更に適した機能が重視されたものとなるであろう。

また、実際に使いやすいデバッガの実現には、機能面の充実だけでなくユーザーインターフェースのさらなる改良も不可欠になってくる。

8 おわりに

現在、HyperDEBU はアプリケーションプログラマに実際に使用されており、これに基づいた評価・改良を進める予定である。

参考文献

- [1] 館村, 小池, 田中: “並列論理型言語 Fleng のマルチウインドウ・デバッガ HyperDEBU”, 第 40 回情報処理全国大会.