多重階層表現が有効なプログラム構造とそのデバッグ環境

3K－8

藤堂 秀仁　関西大学大学院
江澤 義典　関西大学総合情報学部
植村 知正　関西大学工学部

1　はじめに

デバッギングにおいて、ユーザはプログラムの大域的な流れから局所的な参照関係まで様々な要因を把握する必要がある。そこで、プログラム理解を容易にする補助的手段として、プログラムの構造を可視化することが有用だと考えられる[1]。しかしながら、制御の流れだけを図示するフローチャートだけでは十分な表現にはならない。本論文では、プログラムの静的構造の可視化として多重階層表現を導入した。次に、プログラム構造を反映し易い多重階層表現の特徴について考察した。最後に、デバッグ環境への応用について述べる。

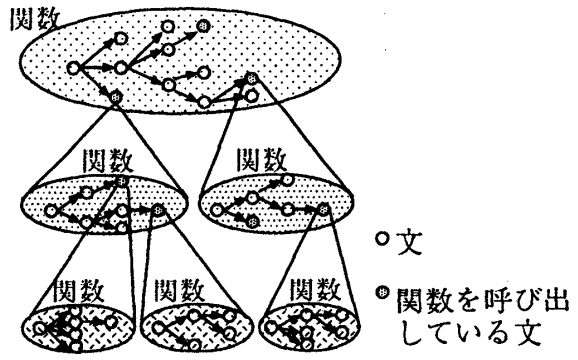

図 1: 多重階層表現

2　多重階層表現

プログラムは一般に複数の関数から構成されている。多重階層表現 (図 1) とは、プログラムを関数ごとで区切り、関数の呼び出し関係を階層構造で表現し、そのノードとして関数内の文の依存関係を明示したPDG(Program Dependence Graphs)[2] を用いたものである。これによって、データ依存関係による階層表現、制御依存関係による階層表現、アドレス依存関係による階層表現などの多様な階層構造を多重に表現することができる。

多重階層表現は、どの関数を呼び出しているのか、あるいはどこから呼び出されている関数なのかを図示できる。このことから対象プログラムが次の３つの特徴を有する場合に、多重階層表現が有効であると考えられる。

1. 再帰関数を含むプログラム
2. 多くの関数を含むプログラム
3. 関数間の呼び出し関係が複雑なプログラム

例えば、文献 [3] のプログラムで各関数間にどのような呼び出し関係 (図 2) があるのかを把握したいとき、多重階層表現 (図 3) をすることによって、把握がより容易になると考えられる。

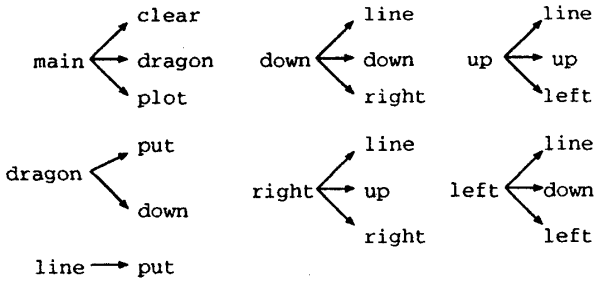

図 2: 関数間の呼び出し関係

3　多重階層表現を用いたデバッグ環境

プログラムの階層を多重表現した場合、階層が上位ほど大域的で下位ほど局所的であることが多い。トップダウン的デバッグ作業の効率を高めるためには、デバッグの視野を大域的視点から局所的視点に順に狭めて階層的理解を前提とすることが有用である。このとき多重階層表現が効果的である。例えば、関数自体には誤りがないのに不正値を返すときは、現在の関数の階層からさらに１つ下の階層にある関数を調べていくことになる。関数内については制御依存関係やデータ

Multiple Representation of Hierarchies of the Program Structure and the Debugging Environment
Hidehito Todo, Yoshinori Ezawa, Tomomasa Uemura
Kansai University Graduate School
Faculty of Informatics, Kansai University
Faculty of Engineering, Kansai University
3-3-35 Yamatecho, Suita, Osaka 564, Japan

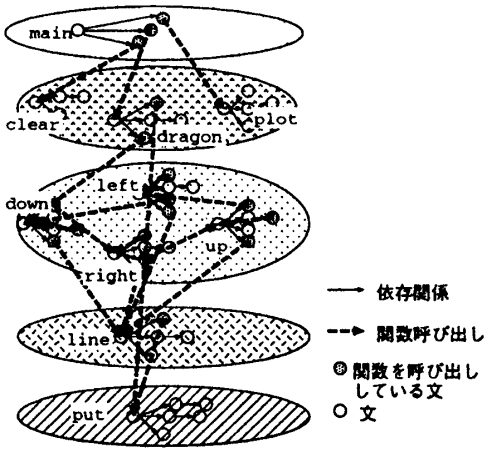

図 3: 多重階層表現が有効なプログラム構造の例

依存関係が明示されていることを利用して、それらの依存関係をたどることによってバグを探索すればよい。そこで、以下の機能が有用であると考えた。

- 関数の呼び出し関係の描画機能
  - 着目している関数はどんな関数を呼んでいるのか
  - 着目している関数はどの関数から呼ばれているのか
- 関数内での文の依存関係 (PDG) の描画機能 (図 4)
- 関数内において着目した文に影響のある文の抽出 (Slicing[4]) 機能 (図 5)

これらの機能はデバッグに有用な環境を提供することができる。また、ユーザがプログラムの静的構造を把握することや関数内の依存関係に着目することによってどのような作用をし得るかの把握が可能となった。現在、これらの機能を備えたシステムが UNIX の X-Window 上に実装されている。

4 おわりに

多重階層表現を提案し、関数が多い、関係が複雑なプログラムに有効であることを述べた。また、多重階層表現を用いることによって、ユーザのプログラムの静的構造の把握とデバッグが効果的にできることを示した。

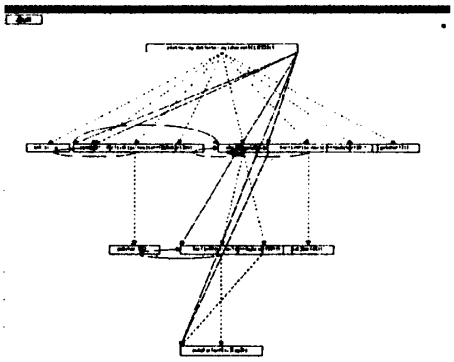

図 4: plot 関数の依存関係 (PDG)

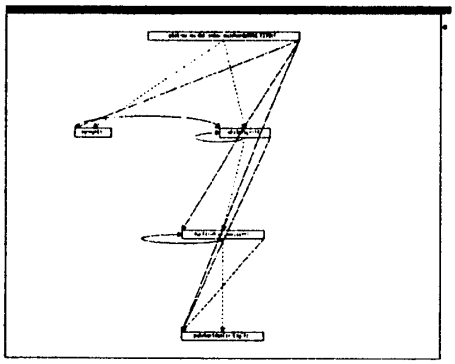

図 5: Slicing の結果

参考文献

[1] 藤堂、松田、柏原、平嶋、江澤、豊田: 操作可能な PDG によるプログラミング支援について、情報処理学会第 47 回全国大会講演論文集 (5), pp. 5-293〜5-294(1993)

[2] S. Horwitz, T. Reps and D. Binkley: Interprocedual Slicing Using Dependence Graphs, ACM Trans. Programming Languages and Systems, Vol. 12, No. 1, pp. 35-46(Jan. 1990)

[3] 石田 晴久: C プログラミング, pp. 131-132, 共立出版 (1991)

[4] Mark Weiser: Program Slicing, IEEE Transactions on Software Engineering Vol. SE-10, No. 4, pp. 352-357(July 1984)