プログラム依存グラフ可視化ツールの開発

3N－4

蒲池 正幸, 程 京徳, 牛島 和夫
(九州大学 工学部)

1. はじめに

ソフトウェアを開発,保守する際にはプログラムの振舞いを理解することが必要となる. プログラム中の各部分の理解を積み重ねて,プログラム全体としての動作を理解しなくてはならず手間や時間,そして経験や勘を要する作業となっている. これがプログラムの開発,保守を困難にしている大きな要因の1つだと考える.プログラムの理解を助けるための良い道具があれば,ソフトウェアの開発や保守の手間や時間を大幅に減少できる可能性がある.

プログラム依存グラフはプログラムの理解を支援するのに有効だと考える. プログラム依存グラフとはプログラム中の文間に存在する依存関係を有向枝として表したグラフである.プログラム依存グラフは, デバッグ, プログラムの理解, 保守, テスト, 複雑さ評価などの幅広い応用が可能である[1]. 例えばデバッグなら, プログラム依存グラフを使うことでプログラムの中で誤りに関係する箇所だけを抽出し調べるといったことが可能になり, 作業を効率化できる. さらにプログラム依存グラフを可視化することで, プログラム中の文間の関係が一目で分かり, プログラムの構造が理解しやすくなる.

本研究ではプログラム依存グラフを可視化し, これをデバッグやプログラム理解などのソフトウェア開発の支援に応用する.

2. プログラム依存グラフについて

データ依存性, 制御依存性, プログラム依存グラフについて簡単に説明する.

- データ依存性
  ステートメントAで定義している変数の値をステートメントBで参照しているとき, BからAへデータ依存関係が存在する.
- 制御依存性
  ステートメントAの制御述語の評価がステートメントBの実行の有無に直接関わるとき,BからAへ制御依存関係が存在する.
- プログラム依存グラフ
  プログラムの各ステートメントと制御述語をノードとし, ノード間のデータ依存関係と制御依存関係を有向枝で表す有向グラフ.

Development of a Visual Tool for Program Dependence Graph
M.Kamachi,J.Cheng, and K.Ushijima
Faculty of Engineering, Kyushu University

3. プログラム依存グラフ可視化ツールについて

プログラム依存グラフ可視化ツールに要求されるものとして次の項目が挙げられる.

- ノードや枝の配置が見易いこと.
- 画面に入りきれない場合にも, 全体を見ることができるようにする仕組みが用意されていること.
- 表示したグラフとソースコードとの対応関係を明確にできること.
- 自動配置されたグラフを容易に修正できること.
- プログラム理解に役に立つ情報を付加できること.

過去に開発されたプログラム依存性の可視化ツールとしては, CAREがある[2]. CAREではノードの配置は分割統治アルゴリズムを用いている. このツールでは横にいくつかのレベルに分け, 横にノードを並べ, 異なるレベルのノード同士を枝で接続する. しかし, 枝の交錯が目立ち, ノード間の関係は見易いものとはいえない.

本研究ではプログラム依存グラフに適した配置アルゴリズムを開発し, それを用いてプログラム依存グラフを可視化し, プログラム理解やデバッグに役立つツールとして完成させることを目指す.

4. 統合的開発支援環境の構築

我々の研究室ではプログラム依存グラフを基礎としたプログラム開発支援環境を開発している. 定義使用グラフ生成ツール, プログラム依存グラフ生成ツール, 生成したプログラム依存グラフを利用してプログラム開発の支援を行うツール群から構成される. 定義使用グラフとはプログラムの処理の流れと, 各ステートメント毎の変数の値の定義と使用の関係と表すグラフである.

ソースプログラムからプログラム依存グラフ生成する過程で, 定義使用グラフを中間表現として生成している. 統合的開発環境の特徴の一つとしては, 生成する定義使用グラフを各プログラミング言語で共通のフォーマットにすることでプログラム依存グラフ生成ツール生成部自体のプログラミング言語への依存をなくしていることが挙げられる. このため未対応のプログラミング言語に対応するには, 定義使用グラフ生成部だけの開発で済む.

今回開発するプログラム依存グラフ可視化ツールはこの統合的開発支援環境のアプリケーションの一つと位置付けられる.

5. プログラム依存グラフ可視化ツールの開発

5.1 ノードの配置アルゴリズム

プログラム依存グラフの配置を行うアルゴリズムについて検討する.

本ツールにおけるプログラム依存グラフのノードの配置を行うアルゴリズムは次のことを目指している.

- ノード同士が重なることがない.
- 各枝がどのノードからどこへと延びているのかについて見やすい.
- 大規模なグラフであっても, 待ち通しくない時間で処理できる.

プログラム依存グラフにおけるノード間の関係はソースプログラムにおける文の前後関係に依存しない. しかし, ソースプログラムとプログラム依存グラフの対応を見易くすることを考え, 依存枝で接続された2ノード間の距離をできるだけ短くなるようにし, これに反しない限りはノード番号に従って上から下へと並べるようにした.

5.2 全体の見渡し

プログラム依存グラフがある程度以上大きくなると, ウィンドウの中に収まりきれなくなる. グラフ全体を眺めるための機能を付加することにした.

1つの実現法としてはスクロールバーがある. しかし, スクロールバーは画面の縦方向の移動と, 横方向の移動の別々に配置する. このため意図した通り上下左右にするためには2つのスクロールバー間を移動する必要があり, 操作が繁雑となる. い. また, 全体の構造が分からない状態ではどこを指せば目的のノードがあるのか見当がつかない.

これに対応するために本ツールでは, 画面中に全体の構成図を用意する. ウィンドウ内に全体図を縮小して表示する. 全体図の中でマウスカーソルがある状態でマウスのボタンをクリックすると, クリックした地点を中心として再描画を行う. こうすることでプログラム依存グラフ全体の中で現在ウィンドウ上に表示されている部分がどういう位置を占めているのかが常に把握できる.

5.3 ソースの表示

プログラム理解やデバッグへの利用を考えるとソースプログラムの対応を明確にする必要がある. ノードに対応するソースリストの先頭の一部を表示することで行う. ソースリストとの対応はプログラム依存グラフ生成ツールの出力にノードに対応するソースの行番号が記されていることから, これを利用して行単位で行う. しかし, 常にノードと共に表示するようにしたのでは, ソース表示だけでかなりの表示面積を消費し, 枝の識別が困難になる. このことから, ソースは一時的に表示することにする. メインのウィンドウでマウスのボタンをクリックしている間だけソースを表示している.

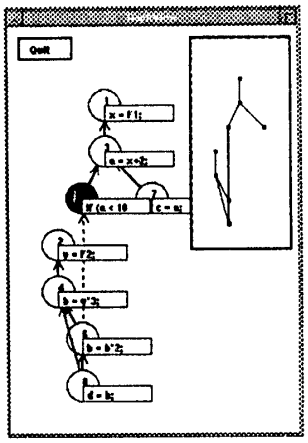

図 1: 今回開発したツールの出力例

5.4 出力例

図1に今回開発したプログラム依存グラフ可視化ツールの出力例を示す. プログラム依存グラフ生成ツールにより, ソースプログラムからプログラム依存グラフを生成し, これをプログラム依存グラフ可視化ツールの入力として用いる. データ依存枝は実線で, 制御依存枝は破線で表示している.

また, ソースプログラムの表示, 非表示はマウスのボタンによって制御するが, 図1ではソースプログラムを表示させた状態となっている.

6. おわりに

今後はこのツールを使ってプログラム理解やデバッグへ実際に応用していき, 目的にかなうようにツール自体に改良を加えていく.

今回の開発したグラフ表示部はプログラム依存グラフ以外にも, 応用が可能と思われる. どういう用途に適しているか, また適さないかについても検討していきたい.

今回開発した配置アルゴリズムは改良の余地が残されている. ノードの上を通過する枝を減少させるには, ノードの場所以外に枝の通過する場所を確保するといった手法が考えられる.

高速性を失わずに, いかにして配置の見易さを向上させるかを検討していきたい.

参考文献

[1] J. Cheng, "Process Dependence Net of Distributed Programs and Its Applications in Development of Distributed Systems," Proc. IEEE-CS 17th Annual COMPSAC, pp. 231-240, Nov., 1993.

[2] P.K.Linos, "Visualizing Program Dependencies: An Experimental Study", Software–Practice and Experience, vol. 24, No. 4, pp. 387-403, 1994.