ＦＯＲＴＲＡＮ開発支援ツールの開発

－ モジュール間の引用関係の静的整合性チェック方式 －

2H－7

佐藤 伸一* 1)　川口 荘太郎 1)　佐藤 輝彦 1)　伊藤 信幸 1)　磧谷 利夫 2)
1) 日立東北ソフトウェア　2) 日立製作所ソフトウェア工場

１．はじめに

プログラムのデバッグ作業において、デバッグに必要な情報は、プログラムを実行しなければ得られないものが多く、また、得られる情報も詳細な情報ではないため、その作業に多大な時間を費やしていた。

そこで、ＦＯＲＴＲＡＮプログラムのコンパイル時、モジュール間の引用関係の情報を採取し、その情報をもとにモジュール間の引用関係の整合性チェックを行い、その結果をデバッグ情報として詳細に出力することにより、従来ＦＯＲＴＲＡＮ言語の実行時がデバッグ情報を出力していたのに比べ、より効率的なデバッグ作業を可能とした。

本報告では、モジュール間の引用関係の整合性チェックを行い、デバッグ情報を出力するための処理方式と、出力するデバッグ情報について述べる。

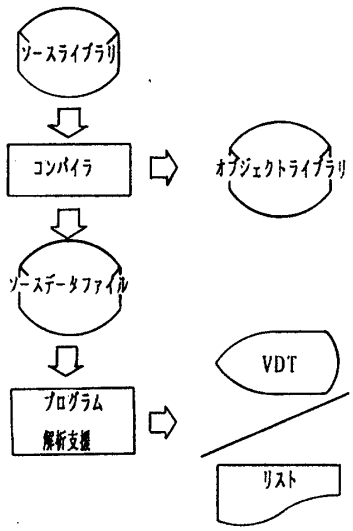

図１．プログラム解析支援システムの処理の流れ

２．モジュール間の引用関係のチェック内容

モジュール間の引用関係とは、引用する側と引用される側の関係のことであり、本機能では、モジュール間の引用関係について、下記の誤りがないかをチェックする。

（ｉ）関数がサブルーチンとして引用されている
（ii）サブルーチンが関数として引用されている
（iii）関数の型が一致していない
（iv）関数の長さが一致していない
（ｖ）引数の個数が一致していない
（vi）引数の種別が一致していない
（vii）引数の型が一致していない
（viii）引数の長さが一致していない

```
100  PROGRAM MAIN          100  SUBROUTINE SUB(C,D)
200  REAL A,B              200  REAL C
 :        :                300  INTEGER D
500  CALL SUB(A,B)          :        :
 :        :                 :        :
900  STOP                  800  RETURN
910  END                   900  END
```

図２．ソースプログラム

３．処理方式

図１に、プログラム解析支援システムの処理の流れを示し、次に処理方式を述べる。

図１において、コンパイラはソースライブラリより、図２のようなソースプログラムを入力とし、１モジュール単位に字句解析・構文解析を行い、モジュール内で手続きを引用している場合、その引用する側の情報として（図２では、モジュール’ＭＡＩＮ’での手続き’ＳＵＢ’の引用）、引用する手続き名、引用元モジュール名、引用位置、手続きの属性（種別、型、長さ）、および実引数の個数、各々の実引数の属性（種別、型、長さ）をテーブルに登録する（図３における（ａ））。また、引用される手続きの宣言側の情報として（図２では、モジュール’ＭＡＩＮ’、および’ＳＵＢ’の宣言）、モジュール名、モジュールの属性（種別、型、長さ）、および仮引数の個数、各々の仮引数の属性（種別、型、長さ）をテーブルに登録する（図３における（ｂ））。

Static check of module cross reference.

Satou S 1)　Kawaguchi S 1)　Satou T 1)　Itou N 1)　Sekiya T 2)
1) Hitachi Tohoku Software Co.,Ltd.　2) Software works,Hitachi,Ltd.

その後、その情報を外部ファイルのソースデータファイルに出力する。

(a) 引用する側の情報

引用する手続き名	引用元モジュール名	引用位置	種別	型	長さ	実引数の個数
SUB	MAIN	500	サブルーチン	−	−	2

実引数名	種別	型	長さ
A	変数	実数	4
B	変数	実数	4

(b) 引用される側の情報

モジュール名	種別	型	長さ	仮引数の個数
MAIN	主プログラム	−	−	0
SUB	サブルーチン	−	−	2

仮引数名	種別	型	長さ
C	変数	実数	4
D	変数	整数	4

図３．プログラムの引用関係の情報

次に、プログラム解析支援システムが、ソースデータファイルを入力として、モジュール間の引用関係の整合性チェックを行う。すなわち、引用される側の情報（図３における（ｂ））を基準として、引用する側の情報（図３における（ａ））との照合を行い、モジュール名と引用する手続き名が一致するものについて、相互の情報を比較し、引用関係の整合性をチェックして、その結果をＶＤＴあるいはリストに出力する。

４．出力情報

図４に、モジュール間の静的整合性チェック方式によりＶＤＴ上に出力した例を示す。

図４では、図２のプログラム中のモジュール’ＳＵＢ’について引用関係の整合性チェックを行った結果を出力した例である。

このとき、エラー情報として、モジュール’ＭＡＩＮ’の引用位置５００の’ＳＵＢ’の引用において、２番目の実引数が、モジュール’ＳＵＢ’の仮引数の型と不一致であることを、記号化して表示している。

また、プログラム解析支援システムは、プログラムエディタ（編集機能）と連動して、デバッグ情報と対応したソースプログラムを同時にＶＤＴ上に表示することができる。

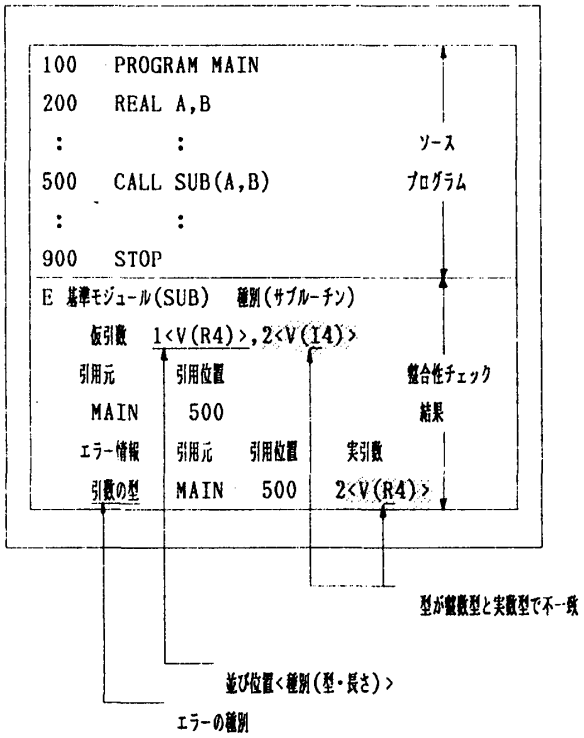

図４．モジュール間の静的整合性チェック方式によるＶＤＴへの出力例

５．効果

本方式では、プログラムを実行することなく、完成しているモジュールをコンパイルするだけで、モジュール間の引用関係のデバッグ情報を提供できるため、デバッグ時間の短縮がはかれる。

また、出力するデバッグ情報は、誤りのある個所と、そのエラーの種別を出力するだけではなく、比較した相互の情報を同時に出力し、より詳細なデバッグ情報を提供している。更に、ＶＤＴ上では、デバッグ情報と対応したソースプログラムも同時に出力できるため、効率の良いデバッグ作業を可能とする。

６．おわりに

モジュール間の引用関係の整合性チェックを、実行することなく、コンパイルするだけで行える方式を開発した。これにより効率的なプログラム開発を行うことができる。