プログラム解析による宣言的仕様の生成と読解支援への応用

6L－5

地引尚史　原田賢一
慶應義塾大学理工学部

1　はじめに

プログラム診断や、ソースコードからのリバースエンジニアリングを目的として、プログラムの処理内容を解析する**プログラム認識** [4. 5. 7] についての研究がいくつか行われている。これは、人が持っていると考えられるプログラミング知識をプログラム断片の形で表現し、主にパターン照合によって処理内容を解析しようとするものである。しかし、パターン照合を主体とする解析であるために、一般のプログラムへ適用するには柔軟性に欠ける面がある。

本論文では、与えられた手続き的プログラムから、その処理内容を表す宣言的仕様を生成する手法について述べる。この手法は、できる限り機械的な変換を行うことを目指しており、パターン照合はその変換過程で必要となる部分でしか行わない。また、この手法をソースコードの読解支援に応用した例についても述べる。

2　プログラム認識の問題点と本研究の方針

人は様々なプログラムを柔軟に理解することができる。プログラム認識においても、様々なプログラムに対応できるように努力がなされているが、人の理解ほどに柔軟な処理ができているとは言い難い [5]。その理由として、無数に存在し得るプログラム断片に、パターン照合が対応しきれないことが挙げられる。このパターン照合主体の解析ははたして妥当な方法なのだろうか。

人はプログラムを書くとき、いくつかの事例について頭の中で実行してみることにより、その動作を確認することが多い [1]。すると、プログラムの処理内容を理解するときにも、頭の中である程度実行しているのではないだろうか。そして、プログラムを一つの関数に見立て、その入力と出力の間に存在する関係を抽出することで、プログラムの機能を理解していると考えられる [2]。

本研究では以上の考えに基づき、プログラムの処理内容の解析を、入力と出力の関係の抽出、という問題に置き換えて考える。そして、この抽出した入出力間の関係を、**宣言的仕様**として表す。この解析は、**記号実行**を用いて、できるだけ機械的な変換によって行う。

3　宣言的仕様の生成

次に本論文で提案する、プログラムの処理内容の解析手法について述べる。問題を簡単にするため、対象とする手続き的プログラムに対し、以下のような制限を加える。

- 型は整数型とその配列だけとする。
- プログラムは構造化されているものとする。
- 関数や手続き呼出しは考えない。

構造化されたプログラムに対し、代入文並び、条件分岐、ループの三つの構造について、それぞれの解析手順を定義する。

3.1　代入文並びの解析

各変数の値 (計算式) を格納する**値表**を用意し、記号実行を行ってゆく。このとき計算できる部分は計算してしまう。式中に残っている変数は、その変数の初期値を表す。最終的に得られた値表が、プログラムの入力と出力の間の関係を表わしたものである。

```
a = 0;
b = a + c;  ⇒
a = 4 + c;
```

a	4 + c
b	c

3.2　条件分岐の解析

then 部と else 部それぞれを別々に解析した後、得られた二つの値表を統合する。

```
if (x > 0)
    y = 1;
else if (x < 0)
    y = -1;
else
    z = 0;
```

⇒

y	1	if	$x > 0$
	−1	if	$x < 0$
	y	if	$x = 0$
z	z	if	$x \neq 0$
	0	if	$x = 0$

A Method for Generation of Declarative Specification from a Program, and Application to Source Code Reading Support.
Hisashi JIBIKI, Ken'ichi HARADA
Keio University
3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223, JAPAN

3.3 ループの解析

ループに対しては、ループ内部を記号実行によって解析した後、全体を抽象化する。このとき、必要ならば変換規則を使って抽象化する。

以下では、次のプログラムを例にとり、具体的な解析の流れを示す。

```
for (i=0; i<10; i++)
    a[i] = i;
```

3.3.1 ループ内部の解析

まずループ内部を記号実行によって解析する。同時に、ループの終了条件も抽出しておく。この結果、ループを一度回ることよって起こる、各変数値の変化がわかる。

値表：

i	$i+1$
a	$a[i]=i$

終了条件： $i \geq 10$

3.3.2 得られた値表の解析

次に、値表の情報をもとに、各変数に対して値の不変式を求める。この不変式は、各変数のループ先頭における値を、ループの実行回数を示す仮想的な変数 LC を使って表現したものである。この変換は、パターンが用意されていればそれを使用し、そうでなければ機械的な変換を試みる。

解析の順番は、値表の計算式の依存関係で決める。この例の場合、変数 a の値の計算式に変数 i が含まれているため、まず変数 i の不変式を求める。変換パターンを用いて、次のように不変式を得る。

値表: $i : i+1$

不変式: $i : LC + i\_$ ($i\_$ は i の初期値)

次に変数 a の不変式を求める。a の値の計算式に変数 i の不変式を代入した後、式中の LC を $LC-1$ で置き換えることで不変式を得る。

値表: $a : a[i] = i$

不変式: $a : a[LC + i\_ - 1] = LC + i\_ - 1$

3.3.3 終了条件の解析

次に終了条件を元に、最終的なループの実行回数 LN を計算する。この変換も、パターン照合を用いて行なう。

ループ実行回数： $LN = 10$

3.3.4 ループ後の値の計算

各変数値の不変式から、各変数のループを抜けた後の値を計算する。

i	$10 + i\_$
a	$a[j + i\_1] = j + i\_1,\ 1 \leq j \leq 10$

3.3.5 ループ前の値表との統合

ループの解析で得られた値表と、ループ直前までの値表とを統合する。

i	10
a	$a[j-1] = j-1,\ 1 \leq j \leq 10$

これが、例題プログラムから得られた最終的な結果である。プログラムの実行によって得られる入力と出力の関係を示している。

4 読解支援への応用

上で示した宣言的仕様生成の手法を用いて、簡単な読解支援システムを試作した。C 言語を対象とし、Mule 上においてリージョンで指定したプログラム断片に対して、入力と出力の関係を示すものである。

ただし、上の方法によって生成した宣言的仕様は、記法が形式的なので、人にとって読み易いものであるとは言えない。そこで、いくつかの表現については、日本語に書き換えて提示するようにしている。

5 おわりに

パターン照合を用いる場面を少なくして、プログラムの処理内容を解析する手法について述べた。従来のプログラム認識に比べ、一般のプログラムに対してより柔軟に解析を行なうことが期待できる。

今後の課題として、以下の点が挙げられる。

- 構造体やポインタなど、複雑なデータ構造への対応。
- 読解支援以外への応用の検討。プログラム診断 [3]、仕様記述言語への変換 [6] など。

参考文献

[1] Gries,D. 著, 筧捷彦 訳：プログラミングの科学, 培風館.

[2] Hausler, P.A., Pleszkoch, M.G., Linger, R.C., Hevner, A.R. : Using Function Abstraction to Understand Program Behavior, IEEE Software, Vol.7, No.1 (Jan. 1990), 55-63.

[3] Johnson, W.L., and Soloway, E.: PROUST: Knowledge-Based Program Understanding, IEEE Transaction on Software Engineering, Vol.11, No.3 (Mar. 1985), pp. 267-275.

[4] 海尻賢二：プログラム認識 / 理解とその応用, 信学技報, KBSE93-11, pp. 51-58.

[5] Ourston, D. : Program Recognition, IEEE Expert, Winter, 1989, pp. 36-49.

[6] 関本理佳, 海尻賢二：プログラム認識による仕様記述の生成について, 信学技報, KBSE93-10, pp. 43-50.

[7] Wills, L.M. : Automated Program Recognition: A Feasibility Demonstration, Artificial Intelligence, No.45 (1990), pp. 113-171.