B_023

プログラムの自動生成における入出力条件について
Input and Output Conditions of Automatic Program Generation

恐神　正博†
Masahiro Osogami

1．まえがき

１９８０年代の中ごろから構造化プログラミング設計法やオブジェクト指向の手法等が提唱され，プログラミングの生産性向上の研究及びそれらの実用化が進められている．一方，従来より手作業によって行われてきたプログラミングという作業を機械化する試み，すなわちプログラムの自動生成に関する技術についても研究が進められている．

ここではモジュールの機能や処理対象のデータの型などに関する属性項目を記述したモジュールの見出し表を作成し，そこから仕様に対して適用可能なモジュールを検索し，C 言語のプログラムを合成する一つの方法と，その中でモジュールの自動リンクの際行われる入出力条件のチェック法において，従来行ってきた[1][2]Prolog のリスト処理の他に，ペトリネットを用いた方法について検討を行う．

2．モジュールの構成

我々は，従来より Prolog を用いた自動合成システムの一つの構成法について，その実験システム MAPP(Module Aided Programming system by Prolog)を通し報告を行なってきているため[1,2]，ここでは Prolog のリスト処理を用いたモジュールの構成及びその検索法について説明を行う．一般に MAPP 内においてモジュールは，次のような構成を持つ．

module(Head,Tail). (2-1)

ここで Head や Tail の引数部はリスト形式を取る．Head はモジュールの機能の種別を表す呼び出し文項目や処理対象のデータ構造の項目などからなる頭部であり，Tail はリスト形式により不定長・不定数の属性項目からなる尾部である．尾部の項目としては，モジュールの関数の型と機能を日本語文で説明するコメント項目，この関数を C 言語プログラムでコールするときの関数表現の項目，引き数部の処理対象の変数の型や，関数の戻り値の型を記述する対応項目，このモジュールを適用するのに必要な入力データやその条件を記述する入力項目，適用後，出力するデータの名前や性質を記述する出力項目，このモジュールを収納するファイル名項目などである．これらの属性項目の記述は，リストを用いることにより不定長でよく，また属性項目の種類はモジュールを通じて同じである必要はない．

†福井工業大学　経営情報学科

3．ソースコードの生成

概略仕様は処理関連対象の型やデータ構造を指定するデータ構造仕様と，処理方法を指定する処理仕様に大別し，データ構造仕様を，

obj_list(spec(NUM)):-obj(x1), obj(x2),･･･,obj(xn). (3-1)

の形で与える．

次に処理仕様を，それぞれの処理項目を process_1, process_2,･･･,process_n とした時，

```
process(spec(NUM)):-proc_list([ process_1,
                                process_2,
                                 ･･･  ,
                                process_n ]).
```
(3-2)

で与える．[3]

概略仕様として上の式(3-1)および式(3-2)がそろった後，次の式(3-3)を実行することで C 言語のソースコードが生成される．[4]

```
prog(spec(NUM)):-
    lang(c),obj_list(spec(NUM)),
    main_head,process(spec(NUM)),
    main_end.
```
(3-3)

式(2-1)で与えるモジュールには，モジュールの適用条件である入力項目 in(IN)，処理後の状態や出力データを記述する出力項目 out(OUT)を設けている．従って，これらの知識を用いることにより，モジュールを組み合わせて作った

プログラムの実行可能性に関するチェックや，逆に入出力条件を与えこれを満たすプログラムをいくつかのモジュールをリンクして生成することができる．その概念図をしたの図１に示す．

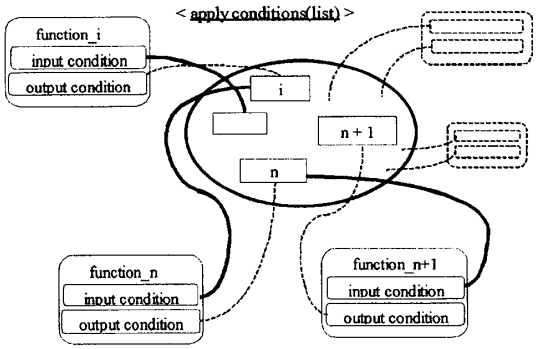

図１．入出力条件のチェック概念図

図１では，まず function_n がリンクされる際に入力条件を蓄えているリスト(apply_conditions)に自身の入力条件（input condition）が存在しているかどうかを調べ，存在した場合自身の出力条件(output condition)をリスト上に追加しリンクされる．次に function_n+1 がリンクされる際，同様に自身の入力条件が既に存在しているかどうかを調べる．例えば，この入力条件が function_n の出力条件であった場合，すでに apply conditions のリストに存在しているので function_n+1 がリンクできることになる．すなわち，finction_n+1 がリンクされるための前提条件が function_n であったことになる．

このように，設計文に記述されているすべてのモジュールに対しそれらの入力条件をチェックしていくが，１つでも条件を満たしていない場合，このリンクは失敗に終わってしまう．そこで，もし自分の入力条件が存在しなかった場合，モジュールを蓄えている辞書の中から必要な前提条件をもつモジュールを探索し，自動的に自分の前にリンクしていく自動補填の考え方を取り入れる．

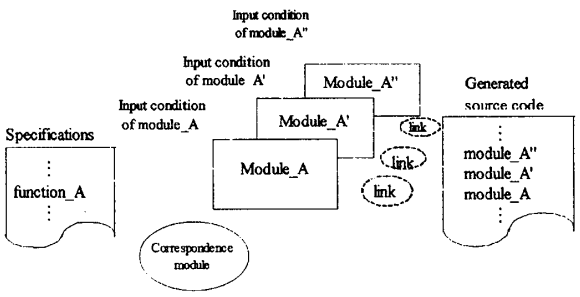

図２．設計文における欠落部分の自動補填

ここでは，最初の処理仕様においては process_A のみが記述されていたにもかかわらず，最終的にその入力条件を満たすための process_A'および process_A"が補填されており，最終的にすべての入力条件が満たされた形でプログラムが生成されていることがわかる．

４．入出力条件のチェックにおける検討

上で述べたように，入出力条件のチェックはおのおののモジュールごとに入出力条件を設け，モジュールがリンクする際，前提としてその入力条件がすでに存在するかどうかでチェックをおこなっていたが，このチェックにペトリネットを導入できれば，システムの安全性や可動性について検証を行うことができる．

function_n と function_n+1 がリンクされる際, function_n+1 の入力条件が function_n の出力条件である場合は図３のよ

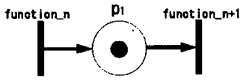

図３．function_n と function_n+1 がリンクされる際のペトリネット

うにペトリネットに書くことができる．自動補填も終了した段階の最終的に生成されるプログラムについてペトリネットでそれらの入出力条件を含めた検証を行えばそのシステムとしての安全性や実行可能性を検証できる．

参考文献

[1]恐神,西田:プロログによるモジュールの検索とプログラムの合成, 福井工業大学研究紀要第 23 号, pp.313-320(1993.3).

[2]恐神,西田:入出力条件によるプログラムの合成, 情報処理学会第５２回全国大会, pp.5-47-48(1996.3)

[3] Osogami, Nishida: "A Method of Automatic Program Desgining and Code Generation using Informal Procedure Call Sentences", Proc. of IASTED -ASC'98, pp.161-164, May 1998.

[4]恐神:プログラム生成におけるリスト処理を用いた入出力条件のチェック, 福井工業大学研究紀要第 29 号, pp.289-296(1999.3).