21-2 構造化と最適化の2段階プログラミング法における等価性証明について

中所武司（日立製作所 システム開発研究所）

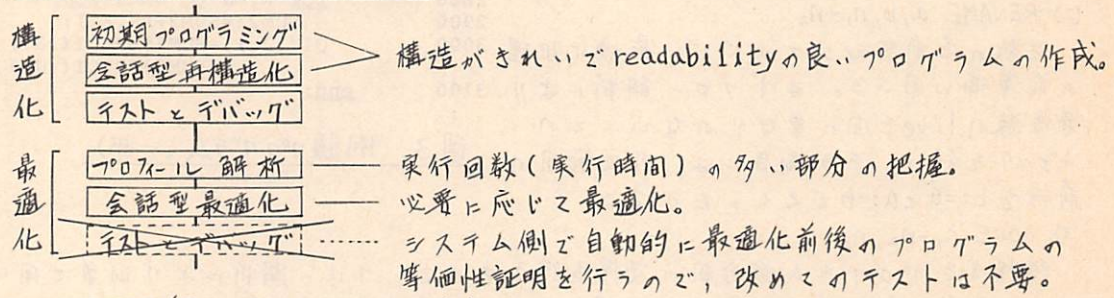
1. はじめに

構造化プログラミングをオブジェクト効率の厳しい分野に適用する方法として最初に構造化，その後に最適化を行うような2段階プログラミング法が有効である。これは，E.W. Dijkstraの「1度に1つの決定を」という段階的詳細化法を，「1つの評価基準で」行うように拡張したものである。

そこで，本方式を実用化するためには，構造化と最適化を会話型で処理するシステムが有効であるが，この場合プログラムの信頼性重視の面から，最適化前後のプログラムの機能的等価性をシステム側で自動証明することが重要である。

会話型最適化システムの研究は幾つか行なわれているが[1]，我々は，先に開発した構造化プログラミング用言語SPL[2]を対象に検討した会話型システムCROPS/SPL[3]の経験を基に，CROPS/Pascal1の基本設計を行った。

2. 2段階プログラミング過程



3. CROPS/Pascal1のシステム構成

本システムの構成を図1に示す。コマンドは以下の4種類から成る。

- 共通コマンド ------ モード変更，退避，終了等。
- 最適化コマンド ----- 主機能（後述）。
- 検証コマンド ------ 最適化コマンド処理用補助
- 編集コマンド ------ コマンドをユーザに解放。

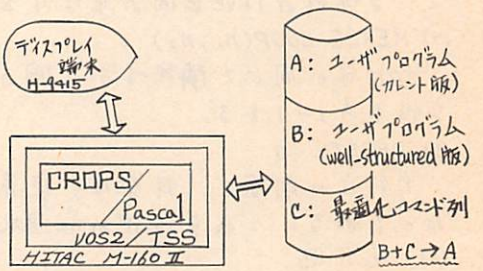


図1. CROPS/Pascalのシステム構成

4. 例題プログラミングによる実験

最適化コマンド抽出のため，机上実験を行った。例題は，あるページングアルゴリズムの解析で必要となった，次の漸化式の解法プログラムを選んだ。（L_jはLRUスタックの参照確率で既知，Q_{ij}はその存在確率，fは参照回数。）

$$
\begin{cases}
Q_{ij} = \begin{cases} 1, & j = 1 \\ 0, & 2 \le j \le j_{max} \end{cases} \\
Q_{ij} = \begin{cases} \sum_{k=1}^{j_{max}} L_k \cdot Q_{i-1,k} & , 2 \le i \text{ かつ } j = 1 \\ \left(\sum_{k=1}^{j-1} L_k\right) Q_{i-1,j} + \left(\sum_{k=j}^{j_{max}} L_k\right) Q_{i-1,j-1} & , 2 \le i \text{ かつ } 2 \le j \le j_{max} \end{cases} \\
f(n) = \prod_{i=1}^{n} \left(\sum_{j=1}^{j_{max}} L_j \cdot Q_{ij}\right)
\end{cases}
$$

まず，readabilityを重視して，本式を忠実にプログラム化したところ，$n = 40$で約30分のCPU時間を費した。また，Qを2次元配列としたため，そのストレージ

がnに比例して大きくなる。

そこで，図1の最適化コマンド列を机上適用したところ，CPU時間は27'8"から1'19"と1/20に減少した。最初の4コマンドは，2個のループを1個に統合，次の9コマンドは2次元配列を1次元に縮小，最後の4コマンドは，図3の3重ループの2400～2900部分を外2ループと共に外1ループの外に移動するものである。

5. 等価性証明

最適化コマンドはすべてシステム側で，等価性の検証を行う。例えば，図2で用いたコマンドでは以下のように行う。

(1) EXPAND LOOP(n),FIRST

for ループの初回の実行をループの直前に展開するもので，実行が保証されない時は展開部分を実行条件による if 文で包む。

(2) RENAME a,b,n_1-n_2

変数の名前替えコマンドで，最適化処理の前準備に用いる。まずフロー解析により各変数のlive区間に重なりのないことのチェックを行い，ある場合には，指定範囲の前後をb:=aとa:=bでくくったりする。

(3) MOVE n_1-n_2,n_3

隣接するプログラム部分間の交換を行うもので，フロー解析により両者で用いている変数のlive区間の重なりを調べるなど，局所的独立性をチェックする。

(4) MERGE LOOP(n_1,n_2)

for 句が同じで隣接する2個の for ループを統合するもので，両者の局所的独立性をチェックする。

(5) DELETE n

n行目の削除で，対象は，定義が参照されない代入文，実行不可能な文，a:=aなどである。これらは他の最適化コマンドによって生じるものが多い。

(6) その他

ROTATE LOOP は for ループ本体の前半と後半の交換，EXECUTE S は記号実行による置換，SPLIT LOOP は for ループを2個に分離するコマンドである。

```
EXPAND LOOP(3400),FIRST
RENAME SUM,SUM3,3300-4100
MOVE   3300-3350,1500
MERGE  LOOP(1600,3400)
RENAME Q(.I,*.),QI(.*.),1800-4000
RENAME Q(.I-1,*.),PREQI(.*.),1800-4000
DELETE 1710
DELETE 4020
ROTATE LOOP(1600),1720,BACK
EXECUTE S,4010-4030
DELETE 4010
RENAME Q(.1,*.),PREQI(.*.),1300-1570
DELETE 1570
RENAME SUM1,SUM1L(.J.),2400-3000
RENAME SUM2,SUM2L(.J.),2400-3000
SPLIT  LOOP(2200),3000
MOVE   2200-2900,1560
```

図2. 最適化コマンド列

```
      :
1600  for I:=2 to IMAX do
      :
2200    for J:=2 to JMAX do
2300      begin
2400        SUM1:=0;
2500        for K:=1 to J-1 do
2600          SUM1:=SUM1+L(.K.);
2700        SUM2:=0;
2800        for K:=J to JMAX do
2900          SUM2:=SUM2+L(.K.);
3000        QI(.J.):=SUM1*PREQI(.J.)
                    +SUM2*PREQI(.J-1.);
3100      end;
      :
```

図3. 例題プログラム(一部)

6. おわりに

本実験の中で，例えば，$\sum_{j=1}^{jmax} L_j = 1$ を利用した最適化など，未だ等価性証明の方法が解決していないものがあり，今後の課題である。CROPS/Pascal は，Pascal で記述し，完成後，まず本システム自身に適用してみる予定である。

REFERENCES

1) Knuth,D. Structured programming with goto statements. Computing Surveys 6,4 (Dec.1974) 261-301.

2) Chusho,T.and Hayashi,T. Two-stage programming: interactive optimization after structured programming. Proc. 3rd UJCC(Oct.1978) 171-175.

(注) CROPS: Conversational Restructuring,Optimizing and Partitioning System