4G−7
ヒューリスティックなカットセット解析によるPSDLコンパイラの処理ネック解決

横田 和久　橋本 正明　佐藤 正和　竹中 豊文
ATR通信システム研究所

1 はじめに

ソフトウェアの生産性と信頼性を向上するため，筆者らはプログラム仕様の再利用について研究中である。具体的には再利用対象仕様の理解性と拡張性をよくするため，ERモデルと従属性制約を適用した非手続き的なプログラム仕様記述言語PSDLと，そのコンパイラをすでに提案した[1].

そのコンパイラの特徴はJSP法[2]の構造不一致を自動的に検出し解決して，プログラム構造を決めるところにある．そのため，有向グラフ上で，デッドロックの解決に類似した閉路の矛盾を解決することによって，構造不一致を検出している．ところが，閉路数はグラフの有向枝数に対して指数関数的に増大するので，閉路解析が処理ネックになり，従来は100行強のプログラム仕様しか処理できなかった．そこ本稿では，その処理ネックを解決するため，閉路解析法に代わるヒューリスティックなカットセット解析法を提案する．この手法を用いることにより500行を越えるプログラム仕様を処理できるようになった．

2 プログラム仕様記述言語 PSDL

PSDLで記述されたプログラム仕様は，情報層，データ層，アクセス層の3階層で構成される．そのうち，アクセス層には入出力ファイルのアクセス方法を記述し，データ層には入出力データのデータ構造を記述する．

一方，情報層には，ERモデルを用いて，対象世界の情報の枠組みを記述する．対象世界の中の「もの」や「こと」を実体と呼び，実体の集合を実体型と呼ぶ．実体相互の対応づけは関連と呼び，関連の集合を関連型と呼ぶ．個々の実体の性質は属性値の組で表される．

プログラム中で起きる計算は以下の3種類の従属性制約で記述する．属性値従属性制約は実体の属性値を，その実体と関連で対応づけられた他の実体の属性値から得るための計算式を定める．関連存在従属性制約は，実体を相互に対応づける関連を，それらの実体の属性値から得るための計算式を定める．実体存在従属性制約は，実体を，他の実体の属性値から得るための計算式を定める．この場合，それらの実体を相互に対応づける関連も同時に得られる．

3 閉路解析によるプログラム生成法

閉路解析によるプログラム生成法の概略を以下に示す[1].

(1) 有向グラフ作成

グラフの節点はPSDL文に従って集める．PSDLの構成要素である実体型，関連型，従属性制約等に対応して節点を設ける．それらの節点の間には，従属性制約によって生じるデータの依存関係に従って有向枝を張る．

A Solution of PSDL Compiler Performance Neck by Heuristic Cut Set Analysis
Kazuhisa Yokota, Masaaki Hashimoto, Kazumasa Sato, Toyofumi Takenaka
ATR Communication Systems Research Laboratories

枝の方向は，情報層では実体，属性値および関連が制約から参照される方向と，制約からデータが得られる方向に合わせる．また，データ層とアクセス層ではデータが入力ファイルから流出する方向と，出力ファイルへ流入する方向に合わせる．

また，有向枝は節点に流入するデータの流れが同一節点に流入する他のデータの流れと同期しているか否かにより同期型と非同期型に分類される．

(2) 局所的解析

実体型節点に2本以上の枝が流入していれば，同じ実体が異なる順序で流入するため，それらの枝を非同期型にする．また，それらの実体型の属性節点へ流入している枝も非同期型にする．関連型節点についても同じである．

実体型節点または属性節点と，属性値従属性制約節点または実体存在従属性制約節点との間にある枝は，関連によって相互に結ばれた実体の数量関係と対応している．そこで，1つの実体に相手の実体が2つ以上結ばれていれば，同じデータが2回以上参照されるのでその枝を非同期型にする．

(3) 大局的解析

図1(1)に示すような方向混在閉路上で，同図(2)に示すように非同期型有向枝がすべて同じ方向を持っていれば，制約の実行順序に矛盾が起きる．その理由は非同期型有向枝から流入するデータ流と，同期型有向枝から流入するデータ流を同期させる処理を行なわなければならないためである．この制約実行順序の矛盾を解消するには，図1(3)に示すように反対方向の枝を少なくとも1本は非同期型に変えればよい．

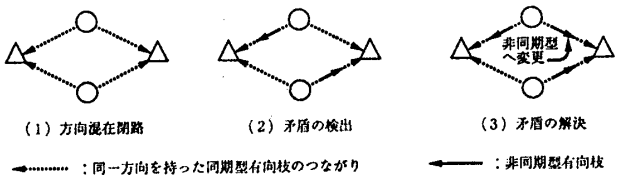

図 1: 閉路解析による矛盾の検出と解決

(4) 連結部分グラフ分割

グラフを，以下のような同期型連結部分グラフに分割する．まず，非同期型有向枝につながっている実体型と属性と関連型の節点を非同期型節点として，その他の節点は同期型節点とする．次に，非同期型節点を部分グラフの境界点に置き，同期型節点は部分グラフの内部に置く．節点を介してつながっている同期型枝よりのデータの流れは相互に同期しているので，同じ部分グラフの中に置く．そうすると，部分グラフの中ではデータの流れがすべて同期する．

(5) プログラム生成

さらに上記の部分グラフへ手続きブロックを割り当てる．ところで大局的解析によって，隣接した部分グラフの境界上の非同期型節点は，その流入枝を一方の部

分グラフ中に持ち，他方の部分グラフは流出枝のみを持つことが保証される．これは，隣接した部分グラフへ割り当てられたブロックの間に実行順序があることを示している．このため，ブロックの間に部分順序ができるので，この部分順序からブロック実行のための全順序を得る．この後，Cプログラムを生成する．

4 閉路解析による処理ネック

グラフ中の閉路の数は最悪の場合，枝の数の指数で増加する [3] ので，閉路数は爆発しやすい．実際，自己記述で作成されたPSDLコンパイラのプログラムで実測して見ると，表1に示すように，閉路数は爆発している．この閉路数の爆発によって，100行強以上のPSDLプログラム仕様はコンパイル不可能であった．

表 1: グラフ中の閉路数実測と実測時間

プログラム名	行数	有向枝数	節点数	閉路数	実測時間
BGA	60	26	23	12	0 *sec.*
AGA	71	45	29	5,773	4 *sec.*
GL1	130	72	50	97,249	4 *min.*
MLA	217	109	76	371,682	23 *hr.*
PARA	176	149	73	実測不可	> 24 *hr.*
GIC	356	167	124	実測不可	> 24 *hr.*

(注)SUN3/80で実測.

5 ヒューリスティックなカットセット解析

グラフのカットセット解析は閉路解析と似た性質を持つことが指摘されている．そこで，図2の(1)のグラフで非同期型有向枝a1を含む閉路は2つ存在するが，(2)のようにa1を含む任意のカットセットをとり，それに含まれるa2とa3とを非同期型にすれば，2つの閉路上の矛盾は解決できる．

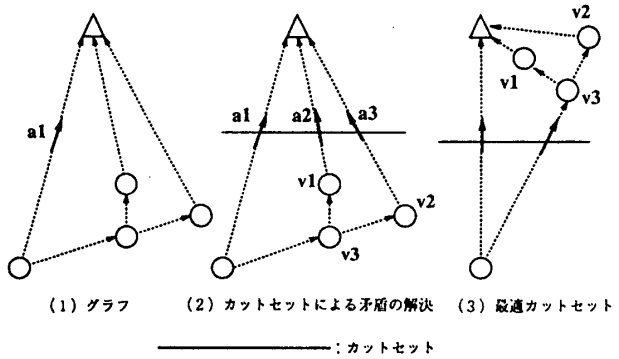

図 2: カットセット解析による矛盾の解決と最適化

しかし，矛盾解決の最適解を得るのに全てのカットセットを解析すると，カットセットの数も最悪の場合，枝の数の指数で増加する [4] ので，やはり爆発が起きる．そのため，ヒューリスティックなカットセット解析法を導入して，爆発を回避することとした．

この解析法では有向グラフを節点の半順序集合と見なし，図2のように有向枝を全て上向きに並べ直す．そこで最初は，図2の(2)の例で示すように，今着目しているa1とは順序を持たない全ての節点，たとえばv1とv2とv3とを，a1で水平に引いた線より下に置く(2(2))．この時，水平線と交わる枝の集合はカットセットになる．

次に，a1とは順序を持たない節点を1つずつ，水平線の上位に移動させる．その移動させる節点よりさらに上位に存在している節点も，上下関係を保ちながら同時に移動させる．すると(2)では非同期型に変わる枝が2本であるが，v3を移動させた(3)では1本となり，効率的な解が得られる．

ところで，水平線の上位に節点を1つずつ移動させるのではなく，同時にn(n=1,2,3,$\cdots$)対ずつ全て移動させれば，水平線上に全てのカットセットが現れる．しかしnが大きくなると，組合せのため，やはり爆発する．そこで小さい値のnについて，矛盾の解決精度を，閉路解析と比較して表2に示す．

表 2: カットセット解析と閉路解析との矛盾解決精度の比較

プログラム	有向枝数	局所解析後の非同期型枝数	大局的解析後の非同期型枝数				
			閉路解析	カットセット解析			
				n=1	n=2	n=3	n=4
BGA	26	4	6	6	6	6	6
AV	31	6	11	9	9	9	9
GIC	167	58	58	61	61	61	59
GAI	180	74	89	85	86	85	85
GII	199	77	90	90	89	90	90

(注)SUN3/80で実測した処理時間は，$n = 1 \sim 2$では$1sec.$，$n = 4$では$1 \sim 10min.$.

その結果，n=1としても，閉路解析との精度差が小さく，さらにカットセット解析の精度が悪くなる例は少ないことが分かった．一方，コンパイラの処理時間は大幅に改善される．

6 まとめ

PSDLコンパイラをヒューリスティックなカットセット解析法で改造した実験によって性能ネックが解消することを確認した．PSDLコンパイラの今後の課題としては，構造不一致の検出解決法と実時間処理プログラムで現れるタイミング不一致の検出解決法の統合や，検出解決精度向上のための解析法の改善，順序構造不一致などがあげられる．

謝辞 日頃ご指導いただくATR通信システム研究所の葉原会長，寺島社長，ならびにご討論いただいた研究室の諸氏，PSDLコンパイラの作成にご協力いただいた日本電子計算株式会社の諸氏に感謝いたします．

参考文献

[1] M. Hashimoto, K. Okamoto: A Set and Mapping-based Detection and Solution Method for Structure Clash between Program Input and Output Data, *Proc. COMPSAC'90*, pp. 629-638, 1990.

[2] M.A. Jackson: *Principles of Program Design*, Academic Press, London, 1975.

[3] P. Mateti, N. Deo: On Algorithms for Enumerating All Circuits of A Graph, *SIAM J. Comput.*, Vol. 5, No. 1, pp. 90-99, 1976.

[4] S. Tsukiyama, H. Ariyoshi, I. Shirakawa: Algorithm to Enumerate All the Cutsets in O(| V | + | E |) Time per Cutset, *Proc. ISCAS*, pp. 645-648, 1979.