照合を用いるSLD導出の実用性に関する検討

7Q-6

周 能法　高木 利久　牛島 和夫
九州大学

1．まえがき

照合を用いるSLD導出は，ユニフィケーションを用いる通常のSLD導出より，効率の面に関しては優れている．これは，照合を用いるSLD導出ではファクトとルールの検索などに対して効率の良い実現技法が存在するからである[3]．しかし，照合を用いるSLD導出で証明不可能なPrologプログラムが存在する．我々は，既に照合を用いるSLD導出で証明可能なPrologプログラムに対する十分条件を挙げた[3]．本稿では，実用性の観点からこの条件の妥当性を検討する．また，証明できないプログラムに対しては，条件を満足しない原因を挙げる．

次の節では，実用性の検討に必要であるモードとモード抽出プログラムという概念について述べる．3節では，照合を用いるSLD導出を定義し，証明可能なPrologプログラムに対する十分条件を挙げる．4節では，実用的と考えられる応用プログラムを対象に検討した結果を示す．5節では，その結果を分析して条件を満足しない理由を考察する．

2．モードとモード抽出

一般的に論理プログラムの引数には方向性がない．例えば，述語に対して入力引数と出力引数という概念がない．しかし，実際のプログラムにおいて述語の一部分の引数が常に入力引数として使われ，一部分の引数が常に出力引数として使われることが多い．このような情報を引数のモード情報と言う．本稿では，二種類のモード情報を取り上げる．第一は，述語の引数の方向を指定するためのモード情報で，述語の呼び出しパターン（Call Pattern）という．これは，次の三つのモードで指定する：in,out,var．inは，対応する引数が常に入力引数として使われることを意味する．outは，対応する引数が常に出力引数として使われることを意味する．varは，対応する引数が入力引数であるか出力引数であるか分からないことを意味する．この三つのモードからなる集合上で次のような半順序関係≦を定義する：{X≦Y| X=Y,X=inかつY=var,またはX=outかつY=var}．第二は，述語の引数が属する項の集合を指定するためのモード情報で，述語の戻りパターン（Return Pattern）という．これは，次の三つのモードで指定する：in-or-ground,free,don't-know．in-or-groundは，対応する引数が入力引数に現れる変数，または定数で構成される項の集合に属することを意味する．freeは，対応する引数が変数からなる集合に属することを意味する．don't-knowは，対応する引数がどんな項であるか分からない，つまりすべての項からなる集合に属することを意味する．この三つのモードからなる集合上で次のような半順序関係≦を定義する：
{X≦Y|X=Y,X=in-or-groundかつY=don't-know,またはX=freeかつY=don't-know}．

与えられたプログラムとトップレベル述語の呼び出しパターンとを入力とし，プログラムに現れるすべての述語の呼び出しパターンと戻りパターンとを抽出するプログラムをモード抽出プログラムと呼ぶ．

p(C1,...,Cn)を述語pの一つの呼び出しパターンとする．プログラムの実行中に現れる述語pの任意の呼び出しパターンp(C1',...,Cn')に対して，もし，C1'≦C1かつ...かつCn'≦Cnが成り立つならば，p(C1,...,Cn)をpの安全な呼び出しパターンと呼ぶ．p(R1,...,Rn)を述語pの一つの戻りパターンとする．プログラムの実行中に現れる述語pの任意の戻りパターンp(R1',...,Rn')に対して，もし，R1'≦R1かつ...かつRn'≦Rnが成り立つならば，p(R1,...,Rn)をpの安全な戻りパターンと呼ぶ．安全な呼び出しパターンと安全な戻りパターンしか抽出しないプログラムを安全なモード抽出プログラムと呼ぶ．

3．照合を用いるSLD導出

Fをファクトの集合，Qをルールの集合，Gを“<-A1,...,Ak”という形のゴールとする．照合を用いるSLD導出では，Gが空となるまで，次の過程を繰り返す．まず，計算規則によって，Gからサブゴールを選択する．そして，次の二つの導出規則によってGから新しいゴールを導出する．

規則1：FからAiθ=Fjとなるファクトを検索して，Gから<-(A1,...,Ai-1,Ai+1,...,Ak)θを導出する．

規則2：QからAi=Hθとなるルール“H<-B1,...,Bm”を検索して，Gから“<-A1,...,Ai-1,B1θ,...,Bmθ,Ai+1,...,Ak”を導出する．

照合でサブゴールとルール頭部とのmgu(most general

Applicability of SLD-resolution That Uses Matching Instead of Unification
Nen-Fa ZHOU,Toshihisa TAKAGI,Kazuo USHIJIMA
KYUSHU UNIVERSITY

unifier)を求めるために，ルール頭部に次の二つの制限を加えなければならない：

1）モードがout或はvarである引数は変数でなければならない．

2）モードがout或はvarである引数に現れる変数が頭部に二回以上出現してはならない．

この二つの制限を満足しないルールがあれば，二つの述語setqとmatchとを用いて制限を満足するようなルールに変換することができる[3]．setq(X,Y)は，変数Xに項Yを束縛する述語である．match(X,Y)は，$X\theta=Y$となる代入θを求める述語である．求めたθがXとYのmguであるためには，以下の条件が満たされなければならない．言い換えると，以下の条件が満たされれば，プログラムが照合を用いるSLD導出で証明可能である．

1）ユーザが与えたトップレベル述語の呼び出しパターンが安全である．

2）モード抽出プログラムが安全である．

3）すべてのルールr="H<-B1,...,Bm"に対して，対応するモードがvarであるHの引数に現れるすべての変数の集合をWとし，対応するモードがinであるHの引数に現れる全ての変数の集合をV0とし，対応するモードがin-or-groundであるBiの引数に現れる全ての変数の集合をViとする．すると，

$$W \subseteq \cup_{i=0,m} Vi.$$

4．実用性の評価

"The Art of Prolog"[2]にある6本の応用プログラムについて前節で述べた条件を検討した．結果は表1に示す．モード抽出には，Debray[1]の安全なプログラムを用いた．C1はプログラムを構成するファクトとルールの数で，C2は条件3)を満たさないルールの数である．結果はプログラムの証明可能性を示す．

表1．評価結果

	応用プログラム	トップレベル述語の呼び出しパターン	C1	C2	結果
1	game (mastermind)	mastermind(out)	23	0	可
2	game (nim)	play(in)	36	0	可
3	game (kalah)	play(in)	73	1	可
4	credit evaluation expert system	credit(in,out)	54	0	可
5	equation solver	solve_equation (in,in,out)	123	52	否
6	compiler	compile(in,out)	72	23	否

5．条件を満足しない原因

表1に示す評価結果を分析することによって，ルールを満たさない原因には二つあることが分かった．一つの原因は，モード抽出プログラムが正確なモードを抽出できないことによる．例えば，次に示す応用プログラム3のルールは条件を満たさないルールである．

```
alpha_beta(0,Position,Alpha,Beta,Move,Value):-
        value(Position,Value).
```

alpha_betaの正確な呼び出しパターンは"alpha_beta(in,in,in,in,out,out)"であるのに，抽出したパターンは"alpha_beta(in,in,in,in,var,var)"である．

もう一つの原因は，プログラムに不完備なデータ構造が使われていることによる．例えば，応用プログラム5には図1(a)のように差分リストが使われている．また，応用プログラム6には図1(b)のように不完備辞書が使われている．

```
factorize(A*B,X,Factors¥Rest):-
        factorize(A,X,Factors¥Factors1),
        factorize(B,X,Factors1¥Rest).
```

(a)

```
lookup(Key,dict(Key,X,Left,Right),Value):-
        X=Value.
lookup(Key,dict(Key1,X,Left,Right),Value):-
        Key<Key1,
        lookup(Key,Left,Value).
lookup(Key,dict(Key1,X,Left,Right),Value):-
        Key>Key1,
        lookup(Key,Right,Value).
```

(b)

図1．不完備データ構造

6．今後の課題

条件を満たさないルールをどのように対処するかが一つの課題である．一番目の原因に対しては，モード抽出をもっと精確に行うプログラムが必要である．二番目の原因に対しては，条件を満さないプログラムを，条件を満たすプログラムに自動的に変換する方法が存在するかどうかを検討する必要がある．

参考文献

[1] Debray, D.K., Warren, D.S.: Automatic Mode Inference for Logic Programs, J. Logic Programming, Vol.5, PP.207-229, 1988.

[2] Sterling, L., Shapiro, E.: The Art of Prolog, The MIT press, 1986.

[3] 周，高木，牛島: Improving the Efficiency of Prolog Programs by Using Matching Instead of Unification, 人工知能学会人工知能基礎論研究会-2, PP.1-10, 1988.