知識工学と人工知能 46－8
（1986. 5. 9）

論理型言語におけるCircumscription

森 辰則　中川 裕志
横浜国立大学 工学部 電子情報工学科

人間の常識による推論を表現するためのアプローチとしてJ.McCarthyの提唱したCircumscriptionと呼ばれる推論方法は高名である。V.Lifshitzの研究によりある限定された論理式の範囲においてCircumscriptionが機械的に計算可能なことが解明されたことで、その応用の道が開けてきた。本稿では、V.Lifshitzの研究を踏まえて、知識を実際に計算機上に表現し推論を行う手段であるPrologを始めとした論理型言語を対象とするCircumscriptionについて考察し、そのアルゴリズムについて述べる。また、本研究で使用した言語では「否定」として“Negation as failure”の他に、陽に否定知識を表現するための論理的否定を導入しているので、この二つの否定がCircumscriptionと、どの様な関係にあるかも考察する。

WG AI 46-8
"Computable Circumscription in Logic Programing" (in Japanese)

Tatsunori MORI and Hiroshi NAKAGAWA

Department of Electrical and Computer Engineering, Yokohama National University, Yokohama, 240, Japan.

Circumscription, proposed by John McCarthy , is an approach to formalize a commonsense reasoning. Vladimir Lifshitz showed the computable circumscription for "separable formulas" which is a subset of first order formulas.

In this paper , we present implementation of the computable circumscription in logic programing languages as Prolog. In order to implement the computable circumscription,in addition to "negation as failure", we intorduce a "logical negation" into Prolog. We found that a relation between "negation as failure" and "logical negation" is an essential part of a circumscription in Prolog.

1．　はじめに

人間の常識による推論を表現するために、知識の完全性を要求される古典論理を越えて不完全な知識を取り扱う様々なアプローチがなされてきている。その中でもJ.McCarthyの提唱したCircumscription[1]と呼ばれる推論方法は高名である。

それは、古典論理の枠組みで、人間が思考の過程で無意識に行っている推論の対象範囲の限定をシミュレートしている。V.Lifshitzの研究[2]により、ある限定された論理式の範囲においては機械的に計算可能なことが解明され、Circumscriptionの応用への道が開けてきた。しかし、Circumscriptionは純粋な一階述語論理を対象としているので実際のプログラム言語との対応は研究されていない。

本稿では、V.Lifshitzの研究を踏まえて知識を実際に計算機上に表現し推論を行う手段であるPrologを始めとした論理型言語を対象とするCircumscriptionについて考察する。本研究で使用した Uranus をベースとする言語では「否定」としてProlog等で通常利用されているNegation as failure(THNOT)の他に、陽に否定知識を表現するためのNOTを導入している[3]。この二つの否定がCircumscriptionとどの様な関係にあるかを考えることが重要なポイントとなる。

2．　Circumscription

Circumscriptionは、パズルを解く場合等に見受けられる「述べられていること以外は考慮しない。」という人間の思考を一階述語論理で表現された世界に適用する方法を定式化したものである。それは与えられた論理式の下で注目する述語（などの論理式）のextensionを最小化して求める事によって成される。通常はそれぞれの述語に関しての最小化は並列に行うので、これをParallel Circumscriptionと呼ぶ。ABを述語定数の組、ZをABに含まれない関数や述語定数の組とするとき、論理式A(AB,Z)の下でZの変化を許した場合のABのCircumscriptionをCircum[A(AB,Z);AB;Z] と表し、その定義を次式とする。

$$\mathrm{Circum}[A(AB,Z);AB;Z] \stackrel{def}{=} A(AB,Z) \wedge \neg\, \exists ab\, \exists z.[A(ab,z) \wedge (ab < AB)] \quad (2.1)$$

ここで、記号≤はコ（含意）に等しく、＜は同値の場合を除いた関係である。また束縛変数ab,zはそれぞれAB,Zと同じ形の述語の記号を値とする述語変数である。なお、Zが空の場合はZを省略して表記する。

McCarthyはCircumscriptionの対象を統一して表現すべく例外を表すabnormal述語を導入した。abnormal述語を用いると一般的な事象を表現し、かつ、例外をも許す表現が可能となる。例えば、「鳥は一般に飛ぶ。」という知識は次のように表現される。ここで、abnormal述語abは鳥の“飛ぶ”という性質についてのabnormality（異常性）を表現している。

$$\forall x.\,[bird(x) \wedge \neg ab(x) \supset fly(x)]$$

2．1　ComputableなCircumscription

CircumscriptionはSeparableと呼ばれる特別な形をした論理式に対しては機械的に計算可能であることがV.Lifshitzにより示された。separable formulasの特別な場合であるsolitary formulasは次のように定義される。

ABを述語定数の組とする。論理式A(AB)がABに関してsolitaryであるとは次のよ

うな形をしている場合である。

$$N(AB)\wedge(U\leq AB) \qquad (2.1.1)$$

ここで、N(AB)は ABが正のリテラルとして出現しない論理式であり、UはABを含まない述語の組である。このとき

$$Circum[N(AB)\wedge(U\leq AB);AB] \equiv N(U)\wedge(U=AB) \qquad (2.1.2)$$

が成り立つ。Zが空でない場合には次式が成立して、空な場合に帰着できる。

$$Circum[A(AB,Z);AB;Z] \equiv A(AB,Z)\wedge Circum[\exists z.A(AB,z);AB] \qquad (2.1.3)$$

ここで再び二階の変数が現れるがA(AB,z)がzに関してsolitaryな場合、すなわち次式によって表される場合、

$$A(AB,z) \equiv Nn(AB,z)\wedge(Uu(AB)\leq z) \qquad (2.1.4)$$

次式が成立して、二階の限量子記号は消去できる。

$$\exists z.A(AB,z) \equiv Nn(AB,Uu(AB)) \qquad (2.1.5)$$

2．2 Prioritized circumscription

それぞれの述語の最小化の結果が互いに矛盾する場合などabnormal述語のいくつかに最小化に関しての優先順位を設けたいことがある。これには関係≤の代わりに次の関係≤を考える。

$$ab\preceq AB \equiv \bigwedge_{i=1}^{k}[\bigwedge_{j=1}^{i-1}\{(ab^j=AB^j)\supset(ab^i\leq AB^i)\}] \qquad (2.2.1)$$

これを用いると添字の小さいものが先に考慮されるので、二つ以上の違ったabnormalityに優先順位を設けることができる。関係≤に関するCircumscription $Circum_{\preceq}[A;P;Z]$を$Circum[A;AB^1>\cdots>AB^k;Z]$と書きprioritized circumscriptionと呼ぶ。prioritized circumscriptionは次のようにparallel circumscriptionの連言によって表すことができる。

$$Circum[A;AB^1>\cdots>AB^k;Z] \equiv \bigwedge_{i=1}^{k} Circum[A;AB^i;AB^{i+1},\cdots,AB^k,Z] \qquad (2.2.2)$$

3． 論理型言語におけるCircumscription

Horn節の集合をプログラムと見なすProlog等のような言語は、基本的には一階述語論理式の部分系なのでCircumscriptionの枠組みはそのまま当てはまる。問題となるのは二つの否定の扱いについてであるが、これは後半で詳細を述べる。

3．1 アルゴリズム

次のような簡単な例を考えCircumscriptionのアルゴリズムの概略を述べる。

((FLY *X) (BIRD *X)(THNOT (AB1 *X)))
((NOT (FLY *X)) (OSTRICH *X)(THNOT (AB2 *X)))

↓

A= [(BIRD *X)∧(THNOT (AB1 *X)) ≤ (FLY *X)] ∧ (3.1.1)
[(OSTRICH *X)∧(THNOT (AB2 *X)) ≤ (NOT (FLY *X))] (3.1.2)

においてCircum[A; AB2>AB1; FLY]を求める。

① prioritized circumscriptionを(2.2.2)を用いてparallel circumscriptionの連言に変換する。（②以降はparallel circumscriptionの手続きを示す。）

Circum[A; AB2>AB1; FLY]≡Circum[A; AB2; AB1,FLY]∧Circum[A; AB1; FLY]
(3.1.3) (3.1.4)

②Circumscriptionの過程で変化を許す述語を(2.1.3)(2.1.5)を用いて消去する。

(3.1.3) $\xrightarrow{(2.1.3)}$ Circum[A; AB2; AB1,FLY] ≡ A∧Circum[∃ab1 ∃fly.A; AB2]

述語FLYは(3.1.1)で正のリテラルとして(3.1.2)では負のリテラルとしてのみ現れているので論理式Aは述語FLYに関してsolitaryとなっている。(2.1.5)を利用して述語FLYを消去するが、その準備として論理式における含意と標準形の関係を用いて次の変形を施しておく。消去しようとしているリテラルが
正のリテラルの場合はclauseのheadとなるように変形
負のリテラルの場合はclauseのbodyに来るように変形
この変形の詳細は後に述べることとしてとりあえずは変形の結果のみを示す。

(3.1.1)→(BIRD *X)∧(THNOT (AB1 *X)) ≤ (FLY *X) (3.1.1)
(3.1.2)→(OSTRICH *X)∧(THNOT (AB2 *X)))∧(THNOT(NOT(FLY *X))) ≤(FALSE)
→(OSTRICH *X)∧(THNOT (AB2 *X)))∧(FLY *X) ≤(FALSE) (3.1.5)

(2.1.5)を対応させて書くと

A(AB,fly) ≡ Nn(AB,fly)∧(Uu(AB)≤fly) (3.1.6)
∃fly.A(AB,fly) ≡ Nn(AB,Uu(AB)) (3.1.7)

(3.1.1)がUu(AB)≤flyに対応し、(3.1.5)がNn(AB,fly)に対応する。(3.1.7)はこれらの式からflyを消去する方法を示している。すなわち(3.1.1)から得られたFLYのextension（clauseのbody）を(3.1.5)のFLYの出現している部分に代入するのである。これは、Prologにおけるプログラム変換に対応させると、いわゆるunfoldにより(3.1.5)のFLYを(3.1.1)のclauseを用いて部分実行させることにより正当性を失わずにFLYを消去していることにあたる。

(BIRD *X)∧(THNOT (AB1 *X)) ≤(FLY *X) (3.1.1)
↓ unfold
(OSTRICH *X)∧(THNOT (AB2 *X))∧(FLY *X)≤(FALSE) (3.1.5)
↓

(OSTRICH *X)∧(THNOT (AB2 *X))∧(BIRD *X)∧(THNOT (AB1 *X)) ≤(FALSE)
(3.1.8)

次に、FLYの場合と同じ手続きによりAB1を消去するがFLYを消去した後の(3.1.8)はAB1を正のリテラルとしてのみ含む。このような場合は(3.1.9)のようなAB1を負のリテラルとして含む恒真式を補ってやればよい。

(3.1.8)→(OSTRICH *X)∧(THNOT (AB2 *X))∧(BIRD *X) ≤(AB1 *X)
↓unfold
(AB1 *X)≤(TRUE)
↓ (3.1.9)
(OSTRICH *X)∧(THNOT (AB2 *X))∧(BIRD *X) ≤(TRUE) (3.1.10)

③(2.1.5)におけるHorn節の連言Nn(AB,Uu(AB))を変形して現在注目して最小化する述語を含意する形の式にする。最後に次式を適用する。

Circum[N(AB)∧(U≤AB); AB] ≡ N(U)∧(U=AB) (2.1.2)

(2.1.2)において含意(≤)を同値(=)にすることが、述語ABをcircumscribeしたことになる。(3.1.10)より変形して

(OSTRICH *X)∧(BIRD *X)∧(THNOT (TRUE)) ≤(AB2 *X) (3.1.11)
↓
(FALSE)≤(AB2 *X)
↓ (2.1.2)によりcircumscribe
(FALSE)=(AB2 *X)
∴ Circum[A; AB2; AB1,FLY] ≡ A∧Circum[∃ab1 ∃fly.A; AB2]
≡ A∧[(FALSE)=(AB2 *X)]

これで(3.1.3)のCircumscriptionは完了した。同様にして(3.1.4)を行う。

(3.1.4)→Circum[A; AB1; FLY] ≡ A∧Circum[∃fly.A; AB1]

∃fly.A≡(OSTRICH *X)∧(THNOT (AB2 *X))∧(BIRD *X)∧(THNOT (AB1 *X))
≤(FALSE)
≡(OSTRICH *X)∧(THNOT (AB2 *X))∧(BIRD *X) ≤ (AB1 *X)
∴ Circum[∃fly.A; AB1] ≡
(OSTRICH *X)∧(THNOT (AB2 *X))∧(BIRD *X) = (AB1 *X)

以上のようにして求めるべき次式を得る。

Circum[A; AB2>AB1; FLY]≡Circum[A; AB2; AB1,FLY]∧Circum[A; AB1; FLY]
≡ [(BIRD *X)∧(THNOT (AB1 *X)) ≤(FLY *X)] ∧
[(OSTRICH *X)∧(THNOT (AB2 *X)) ≤(NOT (FLY *X))] ∧
[(FALSE)=(AB2 *X)] ∧
[(OSTRICH *X)∧(THNOT (AB2 *X))∧(BIRD *X) =(AB1 *X)]

この式において第３項、第４項はCircumscriptionにより新たに生成された公理であり、それぞれ述語AB2とAB1の定義を与えている。これより与えられた元の公理からAB1,AB2を消去することができる。すなわち得られた定義を元の公理に代入すればよい。

```
[(BIRD *X)∧(THNOT (OSTRICH *X)∧(BIRD *X)) ≤(FLY *X)] ∧
[(OSTRICH *X)∧(THNOT (FALSE)) ≤(NOT (FLY *X))]
                ↓ program化
((FLY *X) (BIRD *X)(THNOT (OSTRICH *X)))
((NOT (FLY *X)) (OSTRICH *X))
```

このようにCircumscriptionによりabnormal述語を消去した、より直感的な知識表現にすることができる。

３．２　二つの否定の扱い

我々が使用しているUranusシステムでは論理的否定として否定知識を明示するNOTと証明論的否定THNOTを明確に区別しており、それらは本質的に別個のものである。この否定の問題はCircumscriptionの過程でしばしば登場した含意の関係を用いた変形に関わってくる。

述語論理における含意には標準形との間に次のような関係が成り立つ。

$$\forall x.[a(x) \wedge b(x) \supset p(x)] \equiv \forall x.[\neg a(x) \vee \neg b(x) \vee p(x)] \qquad (3.2.1)$$

Prologのclauseは基本的にはHorn節と同じものなので意味的には同様の変形が可能である。しかしNOTとTHNOTの二つの否定を考慮した場合には(3.2.1)における否定をどちらに解釈するかによって変形の方法が二通りに考えられる。

```
((P *X) (A *X)(B *X))→(P *X)∨(NOT (A *X))∨(NOT (B *X))   ? (3.2.2)
((P *X) (A *X)(B *X))→(P *X)∨(THNOT(A *X))∨(THNOT(B *X))? (3.2.3)
```

Ｐの証明過程と(NOT Ｐ)の証明過程の間に相互関係はないので(3.2.2)の場合(A *X)、(B *X)の証明状況には言及していないことになる。これに対して、(3.2.3)は(A *X)と(B *X)の証明状況に触れ、両者が証明された場合に(P *X)が成立することを述べている。以上のことから(3.2.3)の変形を採用する。

また、知識が不完全な場合、例外は証明されないかぎり成り立たないと考えるのが妥当であるので、abnormal述語に付加されて現れる否定はTHNOTと考えるべきであろう。abnormal述語＋THNOT がnon-monotonic logicやdefault logicにおける様相記号Mと同じ意味を持つと考えることもできる。

さて、(3.2.3)の変形を妥当と考えるときCircumscriptionの過程に現れる含意記号を越えてのリテラルの移動は、移動するリテラルにTHNOTを施してやれば良いことになる。

```
(A *X)  ≤ (C *X)                 (THNOT (AB *X)) ≤ (P *X)
     ↑THNOT┘                           ↑THNOT┘
     ↓                                 ↓
(A *X)∧(THNOT (C *X)) ≤(FALSE)    (THNOT (P *X)) ≤ (AB *X)
```

この移動はCircumscriptionの過程で変化を許す述語を消去するunfoldの準備として行うものであるが、次の様な状況は節3.1でも現れたように頻繁に発生する。

(OSTRICH *X)∧(THNOT (AB2 *X)) ≤ (NOT (FLY *X))
↑THNOT┘
↓
(OSTRICH *X)∧(THNOT (AB2 *X))∧(THNOT (NOT (FLY *X))) ≤(FALSE)

(THNOT (NOT …)))が現れるのである。一般に

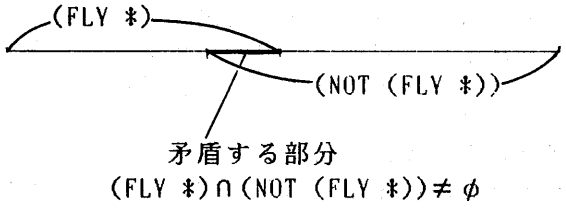
(FLY *X) ≠ (THNOT (NOT (FLY *X)))

であるから、このような状況における消去を考える必要がある。なお、(THNOT (NOT (FLY *X)))は「(NOT (FLY *X))が証明されなければ真」であるから無矛盾性を表す様相記号Mをgoal(FLY *)に適用した表現とほぼ等しい。

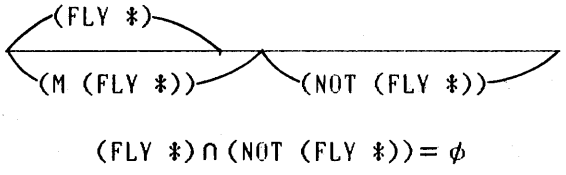
先ほどの例で述語FLYを消去する場合を考え、その意味を考察してみる。

(OSTRICH *)≤(BIRD *) (TRUE)≤(OSTRICH JILL)
(BIRD *)∧(THNOT (AB1 *))≤(FLY *) (3.2.4)
(THNOT (NOT (FLY *)))∧(OSTRICH *)≤(FALSE) (3.2.5)
↓ ↑
(M (FLY *))

(3.2.4)のabnormal述語AB1を調節して矛盾を無くすことがCircumscriptionの目的となる。この場合、矛盾するとは次図のように(FLY *)と(NOT (FLY *))が共に証明されるような変数*の値が存在することである。



逆に矛盾しないとは次図のように(FLY *)と(NOT (FLY *))が両方とも成立するような変数*の値が存在しないことである。



この図で解る通り、述語FLYに関して矛盾を生じさせないためには変数*における(M (FLY *))の領域の中に(FLY *)の領域がすべて含まれるようにしなければな

らない。すなわち述語FLYに関する無矛盾性を保つためには、

$$(FLY\ *) \leq (M\ (FLY\ *)) \qquad (3.2.6)$$

が成立しなければならない。これを、公理に加えて考えるが(M (FLY *))に関する知識が他にはないのでこれを用いて(3.2.5)のgoal(M (FLY *))をunfoldすると、

$$(FLY\ *) \leq \underline{(M\ (FLY\ *))} \qquad (3.2.6)$$

$$\downarrow \text{unfold}$$

$$\underline{(M\ (FLY\ *))} \wedge (OSTRICH\ *) \leq (FALSE) \qquad (3.2.5)$$

$$\downarrow$$

$$(FLY\ *) \wedge (OSTRICH\ *) \leq (FALSE)$$

結局は

$$(FLY\ *) = (M\ (FLY\ *)) \qquad (3.2.7)$$

とおいた場合と同じになる。すなわち、この操作は（Mを述語と見なしたとすると）Mについて(3.2.6)をcircumscribeしていることに等しい。

以上の例で解るように(FLY *)と(THNOT (NOT (FLY *)))を同等に扱ってunfoldすることにより述語FLYに関しての無矛盾性が保たれるのである。

4．　むすび

本稿は、V.Lifshitzの提案したcomputableなCircumscriptionを否定知識を陽に表現した論理型言語に応用する方法について考察した。この方法では対象とするプログラムをseparable formulasに限定しているためにrecursive callを含むような場合には対処する事ができない。この点については今後の課題となろう。

しかし、矛盾を解消する方向でabnormal述語を消去することにより例外知識をルール型の知識に取り込むことができるので、矛盾の発見を知識獲得の契機とする学習にとっては強力なツールとなるであろう。現在、このCircumscriptionを利用して階層構造を持つ概念を学習するシステムを検討中である。このシステムはUranusの多重世界機構により表現された概念知識の上におけるCircumscriptionを考えることにより多重世界内に記述された知識のみならず世界間の構造も学習してしまおうというものである。

abnormal述語の導入の契機を矛盾の発見に置き、いくつかのabnormal述語が導入された時点でこれらにpriorityを与えてcircumscribeするのである。このときのpriorityは概念世界の階層の上下関係から決定する事を考えている。

【参考文献】

(1) J.McCarthy: "Applications of Circumscription to Formalizing Commonsense Knowledge", AAAI Workshop on Non-Monotonic Reasoning (1984)

(2) V.Lifschitz: "COMPUTING CIRCUMSCRIPTION" , Proceedings Tenth International Joint Conferece on Artificial Intelligence(1985)

(3) 中川裕志：「論理型言語におけるexplicitな否定知識」
第27回プログラミングシンポジウム(1986)