知識工学と人工知能 38－3
（１９８５．１．１８）

Ｔ述語を使用したホーン節による否定の表現と開世界推論

西原典孝　森田憲一
（阪大・基礎工）

１．はじめに

知的な内容を持つデータベースの処理（知識ベース処理）を行なうための手段として、データベースを論理式で表現する方法がある。普通、このような知識ベース処理の基本記述言語には、論理型プログラム言語ＰＲＯＬＯＧが用いられる。ＰＲＯＬＯＧのプログラムは、一階述語論理式（今後、ＬＰＣ式と呼ぶ）のsubsetであるホーン節集合で表現される。一方、ホーン節に対しては効率的な推論手続きが存在し、ＰＲＯＬＯＧの動作自体がその推論手続きの実行に他ならない。これが、ＰＲＯＬＯＧが知識ベース処理に用いられる所以である。

ところが、ＰＲＯＬＯＧでは基本的には否定の表現ができないという欠点がある。従来の方法では、否定をnegation as failure としてとらえ、その解決策としてきた。すなわち、“知らない（知識として持っていない）”事柄はすべて“そうではない”という閉世界の立場で考えるわけである。閉世界に相対する考え方として開世界がある。その現実的な有用性についての議論は別にして、人間はこの開世界推論というものを効率的に行なっているように思われる。

本研究の目的は、ホーン節で否定の表現を可能にさせ、かつ開世界推論を可能とさせることにある。ここでいう開世界推論とは、簡単にいうと、次のような推論をすることである。

知識Ｋに対して、質問Ｑが、

Ｋ⊢Ｑ（証明可能）	ならば	yes
Ｋ⊢～Ｑ（矛盾）	ならば	no
else　（独立）	ならば	unknown （知らない）

さて、ＩＣＯＴの国藤らは、知識ベースの整合性をチェックする方法を提案している［１］，［２］。これは、知識ベースに新たな知識を付加える際、その証明可能性、矛盾性、冗長性をチェックし、このどれでもない場合、その新たな知識を独立であるとして知識ベースに付加え、冗長性がある場合には、知識ベースから冗長なものを取除いた後、その新たな知識を付加えるというものである。ここでの証明可能性、矛盾性チェックの方法をそのまま開世界推論に採用できないかという疑問がある。しかし、これには次のような問題がある。

- 証明可能性ないし矛盾性チェックのどちらかで、知識ベースに対する全数検査を必要とするために効率が悪い。
- 知識を肯定的および否定的知識という２つに分割して考えなければならない。
- 知識中の述語の直前に残っている否定記号については negation as failure という閉世界の立場でとらえ、また、全称限量化変数についても閉世界の立場でとらえるため、完全な開世界推論はできない。

本論文では、まず、ａ）述語の引数に肯定ないし否定の情報を埋込んだＴ述語（Ｔruth value embedded predicate ）というものを提案し、ｂ）Ｔ述語を使用したホーン節による知識の表現と、それに対する開世界推論手続き（Ｔ方式開世界推論）について述べ、さらに、ｃ）プログラム言語としてのホーン節の特徴を生かせるために与えた制限Ｔ方式開世界推論について述べる。

２．Ｔ述語とＴ式

２．１　Ｔ述語

従来、“John doesn´t love mary.” および“John loves mary. ”の文に対する論理式は、それぞれ、～love（john, mary）およびlove（john, mary）とされてきた。しかし、前文が表わしている事実は、後文が表わしている事実に否定記号が付いたものととらえるより、“not love”自体を一つの意味を表わしている述語ととらえる方が自然であると思われる。Ｔ述語を提案するそもそもの理由はここにあった。Ｔ述語とは簡単にいえば肯定ないし否定を表明するフラグを述語の引数に埋込んだ述語である。例えば、最初にあげた英文の前者は love（－，john, mary）、後者は love（＋，john, mary）と表わすことにする。一般にＴ述語とは、

p（s , x1,..,xn)　　n ≧ 0, s ＝＋ないし－

の形式をした述語で、p（＋，x1,..,xn）は、（x1,..,xn）には“p ”という関係が成り立ち、p（－，x1,..,xn）は、（x1,..,xn）には“p でない”という関係が成り立つことを意味している。なお、p（＋，x1,..,xn）を正述語、p（－，x1,..,xn）を負述語、第

1引数を真理値フラグと呼ぶことにする。

2．2　T式の定義と一階述語論理式との関係

T式は、T述語を元に、論理記号　∨，∧，→，∀，∃　を用いて構成される式である（否定記号～を用いない点がＬＰＣ式と異なる）。

まず、任意のT式αに対して、その相補式α* を次のように定義する（α* はαの否定に対応している）。

1)　α＝ｐ(＋，x1,..,xn)　ならば
　　α* ＝ｐ(－，x1,..,xn)
2)　α＝ｐ(－，x1,..,xn)　ならば
　　α* ＝ｐ(＋，x1,..,xn)
3)　α＝(β∨γ)　ならば　α* ＝β* ∧γ*
4)　α＝(β∧γ)　〃　α* ＝β* ∨γ*
5)　α＝(β→γ)　〃　α* ＝β　∧γ*
6)　α＝∀x (β)　〃　α* ＝∃x (β*)
7)　α＝∃x (β)　〃　α* ＝∀x (β*)

次に、任意のＬＰＣ式αに対し、それに対応するT式　α´を次のように定める。

(i)　α＝ｐ(x1,..,xn)　ならば
　　α´＝ｐ(＋，x1,..,xn)
(ii)　α＝～β　ならば　α´＝(β´)*
(iii)　α＝(β∨γ)，(β∧γ)，(β→γ)，
　　∀x (β)，∃x (β)
　ならば、それぞれ、
　α´＝(β´∨γ´)，(β´∧γ´)，(β´
　　→γ´)，∀x (β´)，∃x (β´)

最後に、ＬＰＣに対する任意の解釈のもとでのＬＰＣ式αの真理値を元に、αに対応するT式α´の真理値を次のように定める。

　αが真　iff　α´が真

T式の真理値をこのように定めると、T式の中で使われている論理記号（∨，∧，→，∀，∃）がＬＰＣにおけるそれらの記号と、また、相補式を表わす記号 * がＬＰＣの否定記号と、それぞれ同じ論理的性質を持つことが容易に確かめられる。そこで、今後、T式の変形過程を示すために、混乱の生じない限り、否定記号を便宜的に使用することにする（例えば、～Ｌ＝Ｌ* ）。

3．　T述語を使用したホーン節による推論手続き

3．1　知識の表現法

T式はＬＰＣ式と同様に節表現に直せる。そこで、今後はT式の節表現について考えていく。

T式による節表現、

$H_1 ; H_2 ; .. ; H_m \leftarrow B_1 , B_2 , .. , B_n .$
$m,n \geq 0$

ここで、“；”は論理和、“，”は論理積を表わす。

に対して、

$H_1 \leftarrow H_2 ^* , H_3 ^* , .. , H_m ^* , B_1 , .. , B_n .$
$H_2 \leftarrow H_1 ^* , H_3 ^* , .. , H_m ^* , B_1 , .. , B_n .$
　：
$H_m \leftarrow H_1 ^* , .. , H_{m-1} ^* , B_1 , .. , B_n .$
$B_1 ^* \leftarrow H_1 ^* , .. , H_m ^* , B_2 , .. , B_n .$
　：
$B_n ^* \leftarrow H_1 ^* , .. , H_m ^* , B_1 , .. , B_{n-1} .$

このように、各T述語を頭部とするm ＋n 個のホーン節（ただし、右辺のT述語の順序は保存される）を生成し、列挙する。

これらのホーン節は元の節と同値である。このことは、例えば次のようにして示せる。

$H_1 ; H_2 ; .. ; H_m \leftarrow B_1 , .. , B_n .$
$= (H_1 \vee H_2 \vee .. \vee H_m) \leftarrow (B_1 \wedge .. \wedge B_n)$
$= H_1 \leftarrow (\sim H_2 \wedge .. \wedge \sim H_m \wedge B_1 \wedge .. \wedge B_n)$
$= H_1 \leftarrow (H_2 ^* \wedge .. \wedge H_m ^* \wedge B_1 \wedge .. \wedge B_n)$
$= H_1 \leftarrow H_2 ^* , .. , H_m ^* , B_1 , .. , B_n .$

例 3.11

ここで、小文字は定数、大文字は変数を表わし、かっこ内は対応するＬＰＣ式である。

(i)　ｐ(－，ａ)．　　{～ｐ(ａ)．}

ｐ(－，ａ)．

(ii)　ｐ1(－)；ｐ2(＋)←ｐ3(＋)．
　　{～ｐ1；ｐ2←ｐ3．}
ｐ1(－)←ｐ2(－)，ｐ3(＋)．
ｐ2(＋)←ｐ1(＋)，ｐ3(＋)．
ｐ3(－)←ｐ1(＋)，ｐ2(－)．

(iii) $\leftarrow q1(+, X), q2(+, X).$

$\{\leftarrow q1(X), q2(X). = \sim(q1(X) \wedge q2(X))\}$

$q1(-, X) \leftarrow q2(+, X).$

$q2(-, X) \leftarrow q1(+, X).$

このように、T述語を使用することにより、ホーン節での否定の表現が可能になるばかりでなく、非ホーン節もホーン節で記述できるようになる。

3．2　推論手続き（T方式推論）

まず、知識を表わすT式による節集合に対して、各節ごとに前節で述べたようなホーン節を生成し、列挙しておく。そして、質問をT述語によるゴール節、

$\leftarrow G_1, \ldots, G_n.$

で表わし、これらをPROLOGのプログラムとして実行させればよい。

これまでに述べてきた方法は、本質的にはInput resolutionと等価である。Input resolution とは、演繹木の各ノードのどちらか一方が最初に用意された節であるような推論戦略である［3］（図1）。ここで提案した方法は、いわばInput resolution をT述語を使用してホーン節で実現したともいえる。また、それゆえ、推論能力もInput resolution と同等である。だから、例えば次のような推論は成功しない。

$P \leftarrow Q.$

$P \leftarrow Q^*.$

$\leftarrow P.$

4．　開世界推論手続き（T方式開世界推論）

4．1　質問の表現

質問としては、次の2タイプを許す。

タイプ1　PL：　$Q_1, \ldots, Q_n.$

$\{=PL: Q_1 \wedge .. \wedge Q_n\}$

タイプ2　PL：　$Q_1 ; .. ; Q_m \leftarrow P_1, .., P_n.$

$\{=PL: Q_1 \vee .. \vee Q_m \vee P_1^* \vee .. \vee P_n^*\}$

ここで　$PL=[(q1X1), .., (qrXr)]$

$qi=\forall$ないし$\exists$

1) $\sim P \vee Q$

2) $P \vee R$

3) $\sim Q$

4) $\sim R$

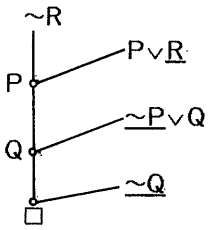

図1　Input resolution の演繹例

このように質問として、すべてのT述語が論理積ないし論理和で結ばれたものを考え、変数の限量化は全称、存在のどちらも使用できるものとする。なお今後、PL* はPL中の∀を∃に、∃を∀に置き変えたものとする。

4．2　タイプ1の質問の取扱い

PL：　$Q_1, \ldots, Q_n.$

a) 証明可能性

導出原理は背理法に基づいているので、質問の否定をとってその矛盾性を証明する。よって、

$\sim\{PL: Q_1, \ldots, Q_n.\}$

$=PL^*: \sim(Q_1 \wedge .. \wedge Q_n)$

により、

$PL^* \leftarrow Q_1, \ldots, Q_n.$

となる。そして、PL* 中の各（∃xi）をスコーレム関数に変換し、PL* を取除いてこれをゴール節として実行する。もし、このゴール節が成功すれば質問は証明可能であり、答は yesとなる。

b) 矛盾性

$PL: Q_1, \ldots, Q_n.$

$=PL: \sim\sim Q_i \wedge Q_1 \wedge .. \wedge Q_r \wedge .. \wedge Q_n$　（$r \neq i$）

$=PL: \sim Q_i^* \wedge Q_1 \wedge .. \wedge Q_r \wedge .. \wedge Q_n$

$$=PL\left[\begin{array}{l} Q_1 . \\ : \\ Q_r . \\ : \\ Q_n . \\ \leftarrow Q_i\ * . \end{array}\right.$$

と変形できるので、よって、

各Q_i に対して、

$$PL\left[\begin{array}{ll} Q_1 . & \\ : & \\ Q_r . & r \neq i \\ : & \\ Q_n . & \\ \leftarrow Q_i\ * . & \end{array}\right.$$

をチェックする。その1つでも成功すれば、質問は矛盾となり、答は no となる。もちろん、一度仮定として知識に付け加えられる$Q_1 . . . Q_r . . . Q_n .$ は、各チェックが終了するごとに除去されるものとする。

例 4.21

ゼウスは神である。

god (+, zeus) .

人間は神でない。

god (−, X) ← man (+, X) .

《生成されるホーン節》

god (+, zeus) .
god (−, X) ← man (+, X) .
man (−, X) ← god (+, X) .

質問：ゼウスは人間ですか。

man (+, zeus) .
証明可能性　← man (+, zeus) . ・・失敗
矛盾性　← man (−, zeus) . ・・成功

よって答は no 。

例 4.22

(∀X) p1 (X) ←p2 (X) ∧p3 (X) . {LPC式}
p1 (+, X) ←p2 (+, X) , p3 (+, X) .

《生成されるホーン節》

p1 (+, X) ←p2 (+, X) , p3 (+, X) .
p2 (−, X) ←p1 (−, X) , p3 (+, X) .
p3 (−, X) ←p1 (−, X) , p2 (+, X) .

質問：(∃X) ～p1 (X) ∧p2 (X) ∧p3 (X)
[(∃X)] : p1 (−, X) , p2 (+, X) ,
p3 (+, X) .

証明可能性

←p1 (−, X) , p2 (+, X) , p3 (+, X) .
・・失敗

矛盾性

p2 (+, c1) .　(c1はスコーレム定数)
p3 (+, c1) .
←p1 (+, c1) .　・・成功

よって答は no 。

例 4.23

(∀X) ～p2 (X)　{LPC式}
p2 (−, X) .

質問：例 4.22 と同様

証明可能性・・失敗

矛盾性

p2 (+, c1) .
p3 (+, c1) .
←p1 (+, c1) .
・・失敗

p1 (−, c1) .
p3 (+, c1) .
←p2 (−, c1) .
・・成功

よって答は no 。

4．3　タイプ2の質問の取扱い

PL :　Q_1 ;..; $Q_m \leftarrow P_1$, .., P_n .

a) 証明可能性

各Q_i と各P_j に対して、

$$PL^*\left[\begin{array}{l} P_1 . \\ : \\ P_n . \\ Q_1\ * . \\ : \\ Q_r\ * .\ r \neq i \\ : \\ Q_m\ * . \\ \leftarrow Q_i . \end{array}\right.$$

ないし

$$PL^*\left[\begin{array}{ll} P_1 . & \\ : & \\ P_r . & r \neq j \\ : & \\ P_n & \\ Q_1\ * . & \\ : & \\ Q_m\ * . & \\ \leftarrow P_j\ * . & \end{array}\right.$$

をチェックする。その1つでも成功すればよい。

b) 矛盾性

$PL \leftarrow Q_1^*, .., Q_m^*, P_1, .., P_n.$

をチェックする。

例 4.31

知識：例 4.21 と同様。

質問：神は人間ではないですか。

[(∀X)] : man (−, X) ← god (+, X).

証明可能性 ⌈ god (+, c1).
⌊ ← man (−, c1). ・・成功

よって答は yes。

例 4.32

p1 (+, X) ; p2 (+, X) ← q (+, X).
q (+, a).

《生成されるホーン節》

p1 (+, X) ←p2 (−, X), q (+, X).
p2 (+, X) ←p1 (−, X), q (+, X).
q (−, X) ←p1 (−, X), p2 (−, X).
q (+, a).

質問： p1 (+, a) ; p2 (+, a).

証明可能性 ⌈ p2 (−, a).
⌊ ←p1 (+, a). ・・成功

よって答は yes。

4．4 メモリ効率

今、同値なホーン節を除外した場合の、知識ベースを構成する節の平均述語数をNとすると、T方式推論では実際の知識ベースの大きさはそのN倍となる。すなわち、論理的にみてN倍の冗長な情報を含んでいることになる。しかし、本来、知識ベースというものの大部分が宣言節から成ることを考えれば、Nの値は十分小さく、どのような知識ベースを構成するかにもよるが、おそらくはその値は1に近くなると思われる。

5． T方式開世界推論の制限系

5．1 制限系の目的

T方式推論では、知識を構成する節はその各述語を頭部とするホーン節に直される。このホーン節の生成を制限することで、T方式推論の能力を制限することができる。本章では、この制限系の1つを提案する。

ここで、制限系を提案する理由は2つある。その理由の1つは、ホーン節生成の際、現実的には無用なホーン節まで作ってしまうおそれがあるということである。例えば、“ジョンは動物が好きである。”という知識に対する節は、

like (+, john, X) ←animal (+, X).

となるが、T方式では、この対偶である

animal (−, X) ←like (−, john, X).

というホーン節をも知識に付加えてしまうことになる。確かに論理的には2つのホーン節は同値であるが、“ジョンが好きでないものは動物でない”という意味の後者のホーン節が、実際に使用されることはないかもしれない。そこで、同値なホーン節生成の際、知識ベース作成者の意図が加えられるようにしたいわけである。

一方、PROLOGの特色の1つは、ホーン節が、意味を記述している論理式であるとともに、ある手続きを行なうプログラムともなりうることであった。この手続き的なホーン節をT方式にも導入したいというのがもう1つの理由である。

5．2 制限系における知識の表現

T式による節表現、

$H_1 ; .. ; H_m \leftarrow B_1, .., B_n.$

$m \geq 1, n \geq 0$

に対して、

$H_1 \leftarrow H_2^*, .., H_m^*, B_1, .., B_n.$

:

$H_m \leftarrow H_1^*, .., H_{m-1}^*, B_1, .., B_n.$

このように、各H_i を頭部とするm 個の同値なホーン節を生成し、列挙する。ただし、m =0 場合、すなわち、

$\leftarrow B_1, .., B_n.$

に対しては、

$B_1^* \leftarrow B_2, .., B_n.$

:

$B_n^* \leftarrow B_1, .., B_{n-1}.$

このように各B_i を頭部とするn 個のホーン節を生成する。

この制限系において、例えば、“人間は神でない”という知識を、その対偶の意味も込めて知識に入れたい場合は、

god (−, X) ← man (+, X).
man (−, X) ← god (+, X).

という2つの節を別々に入力するか、または、

← god (+, X), man (+, X).

ないし、

god (−, X) ; man (−, X) .

として入力しなければならない。

5．3　制限系における質問の取扱い

タイプ1、およびタイプ2の矛盾性チェックについては前と同様に行ない、タイプ2の証明可能性チェックを以下のように変更する。

タイプ2の証明可能性

PL:　Q_1 ;..;Q_m ←P_1 , .., P_n .

m ≧ 1, n ≧0 の場合、各Q_i に対して、

PL*
P_1 .
:
P_n .
Q_1 * .
:
Q_r * .　　r ≠i
:
Q_m * .
←Q_i .

m =0 の場合、各P_i に対して、

PL*
P_1 .
:
P_r .　　r ≠i
:
P_n .
←P_i * .

をチェックする。その1つでも成功すればよい。

5．4　手続き的ホーン節の導入例

次の例題は、文献［1］中の例題を修正したものである。これは、親と子の血液型の結合性制約を表わす知識で、例えば、則夫さん（血液型a ）と由美子さん（血液型o ）夫妻に、太郎（血液型b ）という子供がいる場合、則夫さんまたは由美子さんが（実の）父親または母親ではないことを示している。

```
father (−, X, F) ;mather (−, X, M)
  ←blood-type (+, F, FT),
    blood-type (+, M, MT),
    genes-match (+, FT, MT, CBT),
    blood-type (+, X, BT),
    member (−, BT, CBT).

member (+, X, [X | Y] ).
member (S, X, [A | Y] ) ←member (S, X, Y).
member (−, X, [] ).
```

今、“mather (+, taro, yumiko) ”という知識を既に持っているとすると、質問、

father (+, taro, norio) .

に対する答は no となる。

ここでは、述語memberは手続き的な定義で与えられている。さらに1番目の節に対して、制限系においては、T述語fatherおよびmatherを頭部とするホーン節のみが生成され、それ以外は生成されない。この場合、定義したい知識はfather (−, X, F) とmather (−, X, M) なので、実際にこれら以外のホーン節は無用である。

5．5　解の探索

制限T方式においては、いくつかの組込み述語を用意することで、PROLOGと同様に、質問を満足させる解を求めたり、強制バックトラックの手法で全数解の探索が行なえる。

```
write (+, X) ←write (X) .
write (−, X) ←fail.
p (+, a ) .
p (+, b ) .
```

ここで、 write (X) {Xを出力} 、fail {常に失敗} は組込み述語で、その機能は全くPROLOGの場合と同じである。

```
質問：[ (∃X) ] : p (+, X),
                 write (+, X), fail.
```

まず、証明可能性チェックで、 p (+, X) を満足させる解、a , b が出力される。そして、最終的にfailにより失敗に終わり、次に矛盾性チェックに入る。ここで、もしfailを恒偽な述語と解釈すれば、この質問はfailにより矛盾式となる。すなわち矛盾性チェック、

```
p (+, c1) .
write (+, c1) .
←fail* .
```

において、fail* は恒真な述語なので、これは成功し、

結局、答は no となる。しかし、次の例題を見れば分かるように、むしろ p（+，X）自体の矛盾性をチェックしたいわけである。そこで、failを恒真でも恒偽でもない述語と解釈する（開世界の立場ではこの解釈は妥当である）。すなわち、failもfail* も常に失敗する述語であるとする。

例 5.55

```
知識： p（+，a）.
       p（+，b）.
       q（−，X）.

質問１：[（∃X）]：p（+，X），
                  write（+，X），fail.
  答    a   b
        unknown
質問２：[（∃X）]：r（+，X），
                  write（+，X），fail.
  答    unknown   {r（+，X）であるようなXは
                  知識にない}
質問３：[（∃X）]：q（+，X），
                  write（+，X），fail.
  答    no        {q（+，X）であるようなXは
                  存在しえない}
```

6. まとめ

T述語を導入することにより、ホーン節において、否定の表現と開世界推論が行なえるようになった。さらに、非ホーン節もホーン節で取扱えるようになった。また、制限T方式というものを提案することで、ホーン節の、手続き的処理を行なうプログラムとしての側面をも行かせるようになった。T方式推論の有用性はまだ十分に検討してはいないが、おそらくは、その制限系である制限T方式は、知識ベース処理の基本言語となりえるものであると考えている。

なお、T方式開世界推論およびその制限系は、現在、PC9800上のPROLOGでインプリメントされている。付録にその実行例を載せる。

［参考文献］

1）国藤進，麻生盛敏,et al.：PROLOGによる対象知識とメタ知識の融合とその応用，情報処理学会研究会資料「知識工学と人工知能」，30-1（1983）.

2）北上始，麻生盛敏,et al.：（2）知識同化機構の一実現法，情報処理学会研究会資料「知識工学と人工知能」，30-2（1983）.

3）Chang, C. C. &Lee, R. C. : Symbolic Logic and Mechanical Theorem Proving, Academic Press, pp. 130-162 (1973)

［付録１］T方式開世界推論の実行例

ここで、プロンプト“>”のモードは知識入力モード、“?”は質問入力モードであり、そのモード変換は、それぞれ、q.、k.を入力して行なう。なお、“←”は“：−”、“（∀X）”は“X”、“（∃X）”は“#X”で表わしている。

```
>
>god(+,zeus).
>god(-,X):-man(+,X).
>q.

?man(+,zeus).

no

?[X]:man(-,X):-god(+,X).

yes

?[X]:god(+,X):-man(-,X).

unknown

?[#X]:man(-,X),god(+,X).

yes

?k.
>p1(+,X):-p2(+,X),p3(+,X).
>k1(+,X);k2(+,X):-q(+,X).
>q(+,a).
>q.

?[#X]:p1(-,X),p2(+,X),p3(+,X).

no

?k1(+,a);k2(+,a).

yes
```

[付録２] 制限Ｔ方式開世界推論の実行例

```
>
>write(+,X):-write(X),write(' ').
>write(-,X):-fail.
>member(+,X,[X|Y]).
>member(+,X,[A|Y]):-member(+,X,Y).
>member(-,X,L):-member(+,X,L),!,fail.
>member(-,_,_).
>
>p(+,X):-q(+,X),member(+,X,[a,b]).
>p(-,X):-q(+,X),member(-,X,[a,b]).
>q(+,a).
>q(+,b).
>q(+,c).
>q(+,d).
>q.

?[#X]:p(+,X),write(+,X),fail.
a b
unknown

?[#X]:p(-,X),write(+,X),fail.
c d
unknown

?[#X]:q(-,X),write(+,X),fail.

unknown

?[#X]:p(+,X),p(-,X),write(+,X),fail.

no
```