目次

1. はじめに.

人間やロボットのモデルとして知的主体（agent）というものを考える。agentは基本的事実のいくつかを知っており，推論能力を有している。また他のagentとメッセージを交換することにより知識を増やすことができる。その時他のagentのモデルを自分の中に取り込み，他のagentの推論までも利用して推論する。このようなagent達が登場する例として3賢人問題がある。これは良く知られたクイズである。詳しくは別項で説明する。

自然言語理解や知的インタフェースでは相手の意図を知るという要素が多分にあると思われる。相手の意図を把んでおれば文や発話の理解は正確になる。たとえ不十分な文や発話であってもその意味を正しく理解することを可能にする。また相手の持つ知識や能力に対するメタな知識により，文や発話の表面には出ていない情報を導くこともできるようになる。実際の人間のコミュニケーションの場でもそのような場面が多いように思われる。相手の立場にたって考えられる人間のモデルの研究は重要であると思われる。

知的主体のモデルを様相論理の体系で記述するという考えが文献［SAT 77］,［KON 83］等で提案されている。本Xモはそれらの考えを実験して知識の論理モデルのヒントを得ようとするものである。

実験のために［SAT 77］で展開されている様相体係 GT3, GT4, GT5 から時間パラメタを除いたものを実現した。それらは人があることを知っているシステムの論理的側面を形式化した体系である。3賢人問題にのみ関して言えば，それは一番弱い体係 GT3 ですでに解ける。

実現した様相命題計算は LK［松本 71］である。LKは理論的な体系なので小さな問題しか実際上は解けないと思われる。これは定理証明プログラム一般の限界と共通の問題点と思われる。今回扱ったのは動詞 know だけであり，他にも see, believe, say --- など重要な動詞がいくつもある。それらを全部扱うためには，本システムをどの

ように拡張して行けば良いか？　これは興味深い今後の課題であり，様相論理の土俵からの知識表現言語およびその処理系をもたらすかも知れない。

一方様相論理にもとずかない3賢人問題の解も比較しておもしろいと思い，付録3に示した。それは3賢人問題に対するad-hocな分析にもとずいた解である。このように問題の個別性を重んじて知識表現処理を行なおうという考えも成り立つ[NIL 83]。

なお文献[XIW & WEI 83]では，様相論理に基ずく知識の公理化を与え「Mr. P & Mr. S 問題」を扱っている。同問題は3賢人問題より難しい。

2 3賢人問題

2.1 賢人のモデル (1)

賢人は基本文の集合と推論規則から成る [KOND 93]. 賢人に対して質問 α を与えるとき

$$\Gamma \Rightarrow \alpha$$

が彼の推論規則で証明できるとき「yes」と答えさもなければ「no」と答える. ここで Γ は賢人の基本文の集合である。また, 賢人は基本文の集合を増やすことができる. 一般には知識を変更することも許してよいが 3賢人問題に直接必要ではない。知識を増やすきっかけは見たり聞いたりすることであるがここでは扱わない。

2.2 賢人のモデル (2)

モデル(1)と同様であるが次の部分が若干違う。質問 α が賢人に与えられたとする。賢人は彼の知っている限りの情報を用いて α の可能な値を評価する。その結果 α の値がユニークに決まれば「yes」と答えさもなければ「no」と答える。

この2つの賢人モデルに従って作った2つのプログラムを付録に示す。モデル(1)は様相命題論理の provable を用いて実現している。モデル(2)では Prolog 述語を用いて質問式の可能な値を計算するものであり様相論理は用いない。但し命題論理の簡単な計算は行なう。

2.3 3賢人問題

まずそで扱かっている3賢人問題を述べる。

3人の賢人（賢人1，賢人2，賢人3と呼ぶ）がいる。
(1) 全員，額に白いマークが付けられている。
(2) 各賢人は他人の白いマークは見て知っているが自分自身に白いマークが付いているかどうかは知らない。
(3) 各賢人は「3人のうち少なくとも一人は白いマークを持っていることを全員が知っている」ということを知っている。

ある日そこにある人（王様）が賢人1に彼の額に白いマークがあるかどうかをたずねたところ賢人1は分からないと答えた。それを聞いた賢人2もやはり分からないと答えた。賢人1と賢人2の答えぶりを一部始終見ていた賢人3は自分の色が分かったと答えた。

3賢人の問題を様相論理のはで定式化する。

(1) 命題記号 P_i を「賢人iが額に白いマークを持っている」ことを意味するものとする。ただし $i=1,2,3$
(2) $[K_i]X$ は賢人iが X を知っていることを意味する様相論理式である。（K_i は know（i，X）と書かれる）
(3) $[S]X$ は $[1]X \vee [2]X \vee [3]X$ の省略とする。
(4) 合理的な知的主体（agent）の「考え」で O が知っていないことは無知であるとする：

$$[O]X \rightarrow [I]X$$

これらの仮定を用いて各マップにおける状況を記述したものが次の図1である。特に賢人3の知識の増加に注目したものである。賢人3の答を次の様相論理式の正しさに帰着させることにより3賢人問題が定式化できる。

$$K_1 \wedge K_2 \wedge K_3 \rightarrow [S_3]P_3$$

（K_1，K_2，K_3 は図1を参照のこと）

$$\begin{cases} W_2 = [0](P_1 \vee P_2 \vee P_3) \\ W_3 = [0](\mathbb{I}S_1\mathbb{I}P_2 \wedge \mathbb{I}S_1\mathbb{I}P_3 \wedge \mathbb{I}S_2\mathbb{I}P_3 \wedge \mathbb{I}S_2\mathbb{I}P_1 \wedge \mathbb{I}S_3\mathbb{I}P_1 \wedge \mathbb{I}S_3\mathbb{I}P_2) \end{cases}$$

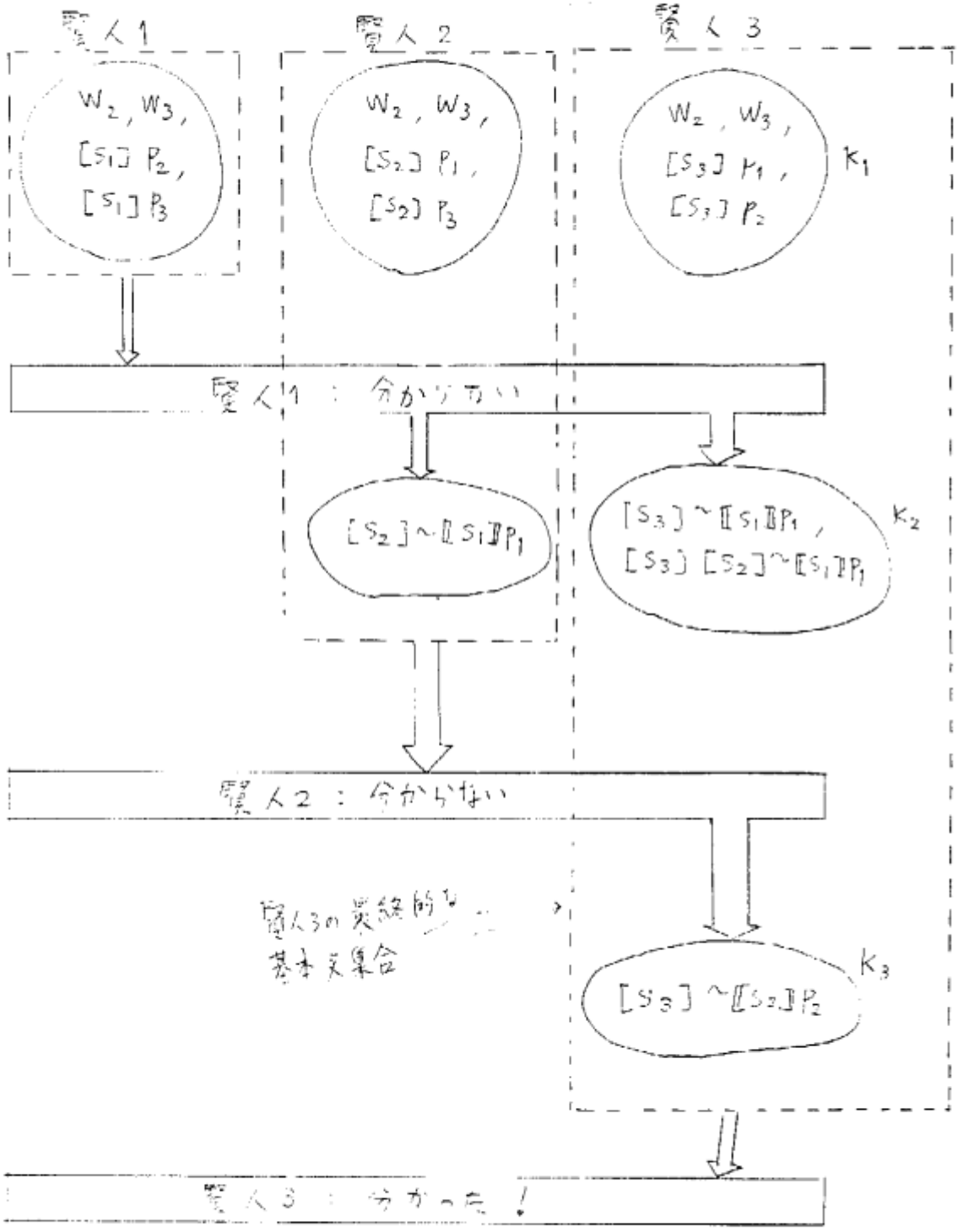

図1　3賢人の知識の増加の推移

3 Knowの形式化.

実現した様相命題計算は参考文献 [SAT 77] に従がっている。ここでは実現した内容を簡単に説明する。

Knowの公理として次のものが与えられている [SAT 77].

(K1) $know(x,y) \rightarrow y$
(K2) $know(0,y) \rightarrow know(0, know(x,y))$
(K3) $know(x, y \rightarrow z) \longrightarrow (know(x,y) \rightarrow know(x,z))$
(K4) $know(x,y) \rightarrow know(x, know(x,y))$
(K5) $\neg know(x,y) \rightarrow know(x, \neg know(x,y))$

実現した命題計算は次のものたちである：

GT3 --- K1, K2, K3
GT4 --- K1, K2, K3, K4
GT5 --- K1, K2, K3, K4, K5

公理の意味は次のとうりである.

K1 : 知られていることは正しい.
K2 : 仮想的なagent 0が知っていることは他の皆なが知っている.
K3 : 各agentは3段論法を使える(賢人である).
K4 : 知っているという自覚を持っている.
K5 : 知っていないという自覚を持っている.

3賢人の問題は実際はGT3の中で既に解ける.

なおknowの形式的な意味論は[SAT 77]を参照のこと.

4　命題計算プログラムの説明

proved (X, Y, Z, U, V, W, G)

式 $\Gamma \Rightarrow \Delta$ が証明可能であることを意味する述語である

但し　$\Gamma = U \cup V \cup W$

$\Delta = X \cup Y \cup Z$

である.

各辺を3つの成分に分けたのは 証明図上の枝分かれをなるべく遅らせるためである。 枝分かれを遅らせた方が同じ計算をする確率は少なくなるからである. 例としてトートロジー(1)を考える.

$$(a \wedge b) \vee \sim(a \wedge b) \qquad \cdots (1)$$

下図はその証明図である.

$$\dfrac{\dfrac{\dfrac{\dfrac{a, b \Rightarrow a, b}{a \wedge b \Rightarrow a, b}}{a \wedge b \Rightarrow a \wedge b}}{\Rightarrow (a \wedge b), \sim(a \wedge b)}}{\Rightarrow (a \wedge b) \vee \sim(a \wedge b)} \qquad (2)$$

$$\dfrac{\dfrac{\dfrac{a, b \Rightarrow a}{a \wedge b \Rightarrow a}}{\Rightarrow a, \sim(a \wedge b)} \quad \dfrac{\dfrac{a, b \Rightarrow b}{a \wedge b \Rightarrow b}}{\Rightarrow b, \sim(a \wedge b)}}{\Rightarrow (a \wedge b), \sim(a \wedge b)}}{\Rightarrow (a \wedge b) \vee \sim(a \wedge b)} \qquad (3)$$

このふたつの証明図を比較する限り 枝分かれを遅らせた方が計算の手間が少なく済むと考えられる。

枝分かれするときに 論理記号をキー記号として持つ論理式はスタックしておく。それ以上分解できない論理式(素論理式)を見つけるための入れものを用意する。したがって シーケントの各辺に つき一つの引き数が必要である。

最後の引数 G は スイッチである。 純粋命題計算, GT3, GT4, GT5 のいずれかが指定できる。

5 今後の課題

本メモでは3賢人問題のみを扱ったが様相論理の枠組が知的主体からなる世界の推論体系としてひとつの有力な道具となることが示せたと思われる。自然言語の中には know 以外にも種々多様の様相 (modality) があることが知られている。様相論理の体系を自然言語理解の枠組とうまく適合させることの検討が次のステップとなる。あまり良く分かっていない自然言語理解モデルを明確な体系と方法を有する様相論理の方から接近して見ることも意義があると考えられる。

3賢人問題は Prolog では setof と述語命題計算を組み合わせることにより簡単にプログラムを書くことができた。同じことは知識表現言語と呼ばれる言語 KRL, Mandala, ---, etc ではどうなるか? また最近の Situation Semantics ではどう記述されるか? それらの記述を比較することは各手法間の関係を明らかにするよう役立つと思われる。

REFERENCE

[SAT 77] M. Sato: A Study of Kripke-type Models for Some Modal Logics by Gentzen's Sequential Method, Publications of the Research Institute for Mathematical Sciences Kyoto University, Vol. 13, No. 2, 1977

[NIL 83] N. Nilsson: A Logical Model of Knowledge, in the Proceeding of IJCAI-83.

[XIW & WEI 83] M. Yiwen & G. Weide: M-JS: A Modal Logic of Knowledge, in the Proceeding of IJCAI-83.

[KON 83] K. Konolige: A Deductive Model of Belief, in the Peoceeding of IJCAI-83.

松本和夫 : 数理論理学, 共立出版, 昭和46年

Appendix 1 Modal Calculus Program

1. Program List

The following is a listing of the modal propositional calculus based on the sequential calculus GCi (i=3,4,5) [Sato 77].

```
% logical operators:
:-op(900,fy,~).
:-op(910,xfy,and).
:-op(910,xfy,or).
:-op(920,xfy,'->').
:-op(920,xfy,'<-').
:-op(930,xfy,'<->').

:- public vt/2,vt/1.
:- public valid/2,valid/1.
:- public proved/7.
:- public seqcal/3,seqcal/2.

runtime:-
        statistics(runtime,[_,T]),
        display('runtime='),
        display(T),
        display(' ms').

vt(X):-seqcal([],[X]).
vt(X,C):-seqcal([],[X],C).

seqcal(X,Y):-
        type(C),
        !,
        seqcal(X,Y,C).
seqcal(X,Y):-seqcal(X,Y,0).
seqcal(X,Y,C):-
        nl,
        statistics(runtime,_),
        proved(Y,[],[],X,[],[],C),
        !,
        display('---proved---'),
        nl,
        runtime.
seqcal(_,_,_):-
        display('NOT valid'),
        nl,
        runtime.

valid(Y):-vt(Y,0).
valid(Y,C):-
        proved(Y,[],[],[],[],[],C).

:-mode proved(-,?,?,?,?,?,?).
proved([],[],_,_,_,_,_).
proved([X|Y],Z,U,V,W,S,C):-
        member(X,Z),!,
        proved(Y,Z,U,V,W,S,C).
proved([X|Y],Z,U,V,W,S,C):-
        X=..[and,X1,X2],!,
        proved([X1,X2|Y],Z,U,V,W,S,C).
proved([X|Y],Z,U,V,W,S,C):-
        X=..[or,X1,X2],!,
        proved([X1|Y],Z,U,V,W,S,C),
        proved([X2|Y],Z,U,V,W,S,C).
proved([X->Y|Z],U,V,W,S,T,C):-!,
        proved([~X,Y|Z],U,V,W,S,T,C).
proved([X<->Y|Z],U,V,W,S,T,C):-!,
        proved([X->Y,Y->X|Z],U,V,W,S,T,C).
```

```
prove([A<->B|R],X,Y,Z,U,V,Ct):-!,
        prove([A,~B|R],X,Y,Z,U,V,Ct).
prove([A<->B|R],X,Y,Z,U,V,Ct):-!,
        prove(P,[(A or ~B) and (B or ~A)|X],Y,Z,U,V,Ct).
prove([A and B|R],X,Y,Z,U,V,Ct):-!,
        prove(P,[A and B|X],Y,Z,U,V,Ct).
prove([~A|R],X,Y,Z,U,V,Ct):-!,
        prove(P,X,Y,[A|Z],U,V,Ct).
prove([A|R],X,Y,Z,U,V,Ct):-
        prove(P,X,[A|Y],Z,U,V,Ct).

:-mode prove1(?,?,?,?,?,?,?).
prove1(Y,Z,[A|R],U,V,Ct):-
        member(A,Z),!.
prove1(Y,Z,[A|R],U,V,Ct):-
        member(A,V),!,
        prove1(Y,Z,R,U,V,Ct).
prove1(Y,Z,[],U,V,Ct):-!,
        succedent_reduction(Y,Z,U,V,Ct).
prove1(Y,Z,[A or B|R],U,V,Ct):-!,
        prove1(Y,Z,R,[A or B|U],V,Ct).
prove1(Y,Z,[A and B|R],U,V,Ct):-!,
        prove1(Y,Z,[A,B|R],U,V,Ct).
prove1(Y,Z,[A->B|P],U,V,Ct):-!,
        prove1(Y,Z,R,[~A or B|R],[A->B|V],Ct).
prove1(Y,Z,[A<->B|R],U,V,Ct):-!,
        prove1(Y,Z,R,[A or ~B|U],V,Ct).
prove1(Y,Z,[A<->B|R],U,V,Ct):-!,
        prove1(Y,Z,R,[(A or ~B),(B or ~A)|U],V,Ct).
prove1(Y,Z,[~A|R],U,V,Ct):-!,
        prove([A],Y,Z,R,U,V,Ct).
prove1(Y,Z,[know(S,A)|R],U,V,Ct):-!,
        prove1(Y,Z,[A|R],U,[know(S,A)|V],Ct).
prove1(Y,Z,[A|R],U,V,Ct):-
        prove1(Y,Z,R,U,[A|V],Ct).

:-mode succedent_reduction(?,?,?,?,?).
succedent_reduction([],Y,Y,Z,Ct):-!,
        antecedent_reduction(Z,Z,Z,Ct).
succedent_reduction([A and B|R],X,Y,Z,Ct):-!,
        prove([A],Y,Z,[],X,Z,Ct),
        prove([B],X,Y,[],Y,Z,Ct).

:-mode antecedent_reduction(?,?,?,?).
antecedent_reduction([],[],Y,Ct):-!,
        deduce_know(Y,Y,Ct).
antecedent_reduction(Y,[A or B|R],Y,Ct):-!,
        prove1([],Y,[A],R,Y,Ct),
        prove1([],Y,[B],R,Y,Ct).

:-mode deduce_know(?,?,?).
deduce_know(Y,Y,Ct):-!,
        ct3(Y,Z).
deduce_know(X,Y,Ct):-!,
        ct4(X,Y).
deduce_know(X,Y,Ct):-
        know(S,A),member(know(S,A),X),!,
        delete(X1,X2,know(S,A)),
        collect(A,X2,X3),
        collect(A,Y,Y1),
        prove([know(S,A)|X3],[],[],Y1,[],[],Ct).

:-mode ct3(?,?).
ct3([know(S,A)|R],Y):-
```

```
        collect1(S,Y,K),
        proved([A],[],[],K,[],[],gt3),!.
gt3([_|X],Y):-
        gt3(X,Y).

:-mode gt4(+,-).
gt4([know(S,A)|R],Y):-
        collect(S,Y,K),
        proved([A],[],[],K,[],[],gt4),!.
gt4([_|X],Y):-
        gt4(X,Y).

:-mode collect(-,+,-).
collect(A,[],[]).
collect(A,[know(0,X)|Y],[know(0,X)|Z]):-!,
        collect(A,Y,Z).
collect(A,[know(A,X)|Y],[know(A,X)|Z]):-!,
        collect(A,Y,Z).
collect(S,[Y|R],Y):-
        collect(S,R,Y).

:-mode collect1(+,+,-).
collect1(A,[],[]).
collect1(A,[know(0,X)|Y],[know(0,X)|Z]):-!,
        collect1(A,Y,Z).
collect1(A,[know(A,X)|Y],[X|Z]):-!,
        collect1(A,Y,Z).
collect1(S,[X|R],Y):-
        collect1(S,R,Y).

:-mode knowledge(+,-).
knowledge([],[]):-!.
knowledge([know(Y,Y)|Z],[know(Y,Y)|U]):-!,
        knowledge(Z,U).
knowledge([_|X],Y):-
        knowledge(X,Y).

:-mode delete(-,-,?).
delete([X|Y],X,Y).
delete([X|Y],[X|Z],U):-
        delete(Y,Z,U).

:-mode append(+,-,-).
append([],Y,Y):-!.
append([X|Y],Z,[X|U]):-append(Y,Z,U).

:-mode member(+,+).
member(Y,[Y|_]):-Y==Y,!.
member(Y,[_|Y]):-member(Y,Y).
```

2. Run Test of the Calculus

2.1 Examples of Proofs in [7].

```
know(a,a)->a
---proved---
runtime=0 ms

know(0,a)->know(0,know(a,a))
---proved---
runtime=11 ms
```

```
know(a,a->b)->know(a,a)->know(a,b)
---proved---
runtime=16 ms

know(a,a)->know(a,know(a,a))
###fail###
runtime=11 ms

~know(a,a)->know(a,~know(a,a))
###fail###
runtime=9 ms
```

2.2 Examples of Proofs in GTM

```
know(a,a)->a
---proved---
runtime=8 ms

know(a,a)->know(a,know(a,a))
---proved---
runtime=10 ms

know(a,a->b)->know(a,a)->know(a,b)
---proved---
runtime=19 ms

know(a,a)->know(a,know(a,a))
---proved---
runtime=11 ms

~know(a,a)->know(a,~know(a,a))
###fail###
runtime=9 ms
```

2.3 Examples of Proofs in GTS

```
know(a,a)->a
---proved---
runtime=9 ms

know(a,a)->know(a,know(a,a))
---proved---
runtime=14 ms

know(a,a->b)->know(a,a)->know(a,b)
---proved---
runtime=20 ms

know(a,a)->know(a,know(a,a))
---proved---
runtime=11 ms

~know(a,a)->know(a,~know(a,a))
---proved---
runtime=17 ms
```

Appendix-2 Three wisemen Problem

1. Description of the Three Wisemen Problem

```
:-op(900,fy,~).
:-op(910,xfy,and).
:-op(910,xfy,or).


w2(know(0,p1 or p2 or p3)).

w3(know(0,(know(s1,p2) or  know(s1,~p2))
          and (know(s1,p3) or  know(s1,~p3))
          and (know(s2,p3) or  know(s2,~p3))
          and (know(s2,p1) or  know(s2,~p1))
          and (know(s3,p1) or  know(s3,~p1))
          and (know(s3,p2) or  know(s3,~p2)))).

w5(know(s2,~(know(s1,p1) or know(s1,~p1)))).

w7(know(s3,know(s2,~(know(s1,p1) or know(s1,~p1))))).

w8(know(s3,~(know(s2,p2) or know(s2,~p2)))).


wise1:-w2(W2),w3(W3),
       seqeal([W2,W3,know(s1,p2),know(s1,p3)],
       [know(s1,p1),know(s1,~p1)],wt3).

wise2:-w2(W2),w3(W3),w5(W5),
       seqeal([W2,W3,W5,know(s2,p1),know(s2,p3)],
              [know(s2,p2),know(s2,~p2)],wt3).

wise3:-w2(W2),w3(W3),w5(W5),w7(W7),w8(W8),
       seqeal([W2,W3,W5,W7,W8,know(s3,p1),know(s3,p2)],
              [know(s3,p3),know(s3,~p3)],wt3).

qa1(X,G):-w2(W2),w3(W3),
       seqeal([W2,W3,know(s1,p2),know(s1,p3)],
       [X],G).
qa2(X,G):-w2(W2),w3(W3),w5(W5),
       seqeal([W2,W3,W5,know(s2,p1),know(s2,p3)],
              [X],G).

qa3(X,G):-w2(W2),w3(W3),w5(W5),w7(W7),w8(W8),
       seqeal([W2,W3,W5,W7,W8,know(s3,p1),know(s3,p2)],
              [X],G).
```

2. Run Test under the Model System CW2

```
| ?- wise1.

no*fail***
runtime=67 ms
yes
```

```
| ?- wise2.

---Failed---
runtime=207 ms
yes
| ?- wise3.

---proved---
runtime=4163 ms
yes
```

3. Run test under each Local System

```
| ?- qa3(know(s3,p3),t3).

---proved---
runtime=3333 ms
yes
| ?- qa3(know(s3,p3),t4).

---proved---
runtime=4030 ms
yes
| ?- qa3(know(s3,p3),t5).

---proved---
runtime=9660 ms
yes
```

Appendix-3 Another solution of the three wisemen Problem

1. Source List

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  A solution of the three wisemen problem by direct use of  %
%  Prolog. In this formulation, the meaning of "x knows y"   %
%  is that x can deduce the unique existence of y using x's  %
%  other knowledges.                                          %
%                                                             %
%  K. Mukai October 1983                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

consistent(X):- \+ valid( ~ X).

q1(p1).
q1(~p1).

q2(p2).
q2(~p2).

q3(p3).
q3(~p3).

wiseman1_say_at_first(Common,[A,B,C],Ans):-
        setof(Q1,
              (q1(Q1),
              consistent(Common and Q1 and B and C)),
              T),
        length(T,N),
        !,(N=1,Ans='Yes, I know'; N>1,Ans='No, I do not know').
wiseman1_say_at_first(_,_,'No, I do not know').

wiseman2_say_at_second(Common,[A,B,C],Ans1,Ans):-
        setof(Q2,
        (q2(Q2),
        consistent(Common and A and Q2 and C),
        wiseman1_say_at_first(Common,[A,Q2,C],Ans1)),
        T),
        length(T,N),
        !,(N=1,Ans='Yes, I know';N>1,Ans='No, I do not know').
wiseman2_say_at_second(_,_,_,'No, I do not know').

wiseman3_say_at_last(Common,[A,B,C],Ans1,Ans2,Ans):-
        setof(Q3,
        (q3(Q3),
        consistent(Common and A and B and Q3),
        wiseman1_say_at_first(Common,[A,B,Q3],Ans1),
        wiseman2_say_at_second(Common,[A,B,Q3],Ans1,Ans2)),
        T),
        length(T,N),
        !,(N=1,Ans='Yes, I know';N>1,Ans='No, I do not know').
wiseman3_say_at_last(_,_,_,_,'No, I do not know').

three(Common,Situ,[X,Y,Z]):-
        wiseman1_say_at_first(Common,Situ,X),
        wiseman2_say_at_second(Common,Situ,X,Y),
        wiseman3_say_at_last(Common,Situ,X,Y,Z).
```

```
wiseman:-
        statistics(runtime,_),
        [q1(Q1),q2(Q2),q3(Q3)],
        Con=(p1 or p2 or p3),
        consistent(Con and Q1 and Q2 and Q3),
        minc(Con,[Q1,Q2,Q3],[_,_,_]),
        display('Then the situation is '),
        print(Q1 and Q2 and Q3),
        display(':'),nl,
        display('wiseman1 answers: '),display(Q1),nl,
        display('wiseman2 answers: '),display(Q2),nl,
        display('wiseman3 answers: '),display(Q3),nl,nl,
        fail;
        statistics(runtime,[_,T]),
        display('runtime = '),display(T),display(ms)).


% Run Time of an Alternative Solution


        | ?- wiseman.
        Then the situation is p1 and p2 and p3:
        wiseman1 answers: No, I do not know
        wiseman2 answers: No, I do not know
        wiseman3 answers: Yes, I know

        Then the situation is p1 and p2 and~p3:
        wiseman1 answers: No, I do not know
        wiseman2 answers: Yes, I know
        wiseman3 answers: No, I know

        Then the situation is p1 and~p2 and p3:
        wiseman1 answers: No, I do not know
        wiseman2 answers: No, I do not know
        wiseman3 answers: Yes, I know

        Then the situation is p1 and~p2 and~p3:
        wiseman1 answers: Yes, I know
        wiseman2 answers: Yes, I know
        wiseman3 answers: No, I know

        Then the situation is ~p1 and p2 and p3:
        wiseman1 answers: No, I do not know
        wiseman2 answers: No, I do not know
        wiseman3 answers: Yes, I know

        Then the situation is ~p1 and p2 and~p3:
        wiseman1 answers: No, I do not know
        wiseman2 answers: No, I do not know
        wiseman3 answers: Yes, I know

        Then the situation is ~p1 and~p2 and p3:
        wiseman1 answers: No, I do not know
        wiseman2 answers: No, I do not know
        wiseman3 answers: Yes, I know

        Then the situation is ~p1 and~p2 and~p3:
        wiseman1 answers: No, I do not know
        wiseman2 answers: No, I do not know
        wiseman3 answers: No, I know
```

```
runtime= 2085ms
yes
```