HANOI UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY
Spring 2009, Artificial Intelligence

TUTORIAL 6: LOGICAL AGENTS

Tutor: Tran Quang Anh

1. General information

- Server: 192.168.6.251
- Username: YOUR NAME, for example: haintv_1c05
- Password: YOUR NAME, for example: haintv
- aima code (read only): /usr/local/acl81_express/aima_code/
- Common Lisp: /usr/local/acl81_express/allegro-express

2. Run Common Lisp and load the code:

- Run CommonLisp:
/usr/local/acl81_express/allegro-express
- Load AIMA code:
(load “/home/ait/aima_code/aima.lisp”)
- Load aima’s logic part
(aima-load ‘logic)
- Compile (you need to do this one time and when the code is changed):
(aima-compile)

3. Logic in general

3.1 Print a Truth Table

- Print a truth table of $P \Rightarrow Q$
(truth-table "P => Q")

- Print a truth table of $\neg P \vee Q$
(truth-table "~P | Q")

- Print a truth table of $(P \vee Q) \wedge \neg P \Rightarrow Q$
(truth-table "(P | Q)^~P=>Q")

3.2 Validity and satisfiability

- Validity: A sentence is valid if it is true in all models.

(validity "P=>Q <=> ~Q=>~P")

(validity "ToBe or not ToBe")

```
(validity "((S => W1|W2|W3|W4) ^ S ^ (~W1^~W2^~W3)) => W4")
```

- Satisfiability: A sentence is satisfiability if it is true in some models.

```
(validity "SillyQuestion")
```

```
(validity "P=>Q")
```

```
(validity "P^Q")
```

4. Knowledge Base

- Create a knowledge base name mykb

```
(setf mykb (make-prop-kb))
```

- Tell the knowledge base some sentences

```
(tell mykb "P")
```

```
(tell mykb "P => Q")
```

Now these sentences (“P” and “P=>Q”) are in the knowledge base.

- Ask the knowledge base some sentences:

```
(ask mykb "P")
```

P is in the knowledge base.

```
(ask mykb "Q")
```

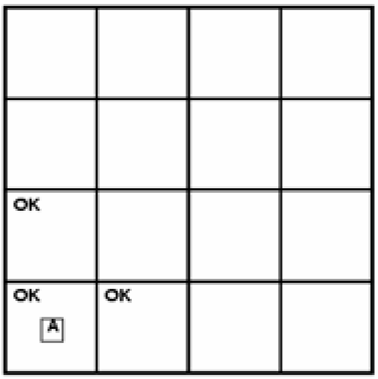
Although Q is not in the knowledge base, but the knowledge base is able to entail that Q is true.

5. The Wumpus problem

- Create a knowledge base name kb1

```
(setf kb1 (make-prop-kb))
```

- When the agent is in square [1,1], there is no Stench and no Breeze:



We tell the KB that information:

```
(tell kb1 "~S11")
```

```
(tell kb1 "~B11")
```

We also tell the KB the related Wumpus rules:

```
(tell kb1 "~S11 => ~W11 ^ ~W12 ^ ~W21")
```

```
(tell kb1 "~B11 => ~P11 ^ ~P12 ^ ~P21")
```

Now we want to know whether we can move to square [2,1].

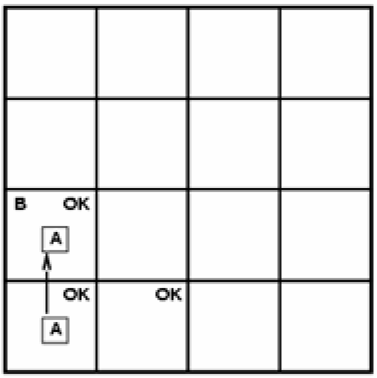
We ask the KB: “Is there not Wumpus in [2,1]”

```
(ask kb1 "~W21")
```

```
(ask kb1 "~P21")
```

The answers should be all yes.

- When the agent moves to square [2,1], there is a Breeze and no Stench.



We tell the KB that information:

(tell kb1 "~S21")

(tell kb1 "B21")

We also tell the KB the related Wumpus rules:

(tell kb1 "~S21 => ~W21 ^ ~W31 ^ ~W22 ^ ~W11")

(tell kb1 "B21 => P31 | P22 | P11")

Now we want to know whether we can move to square [3,1], and [2,2].

We ask the KB: “Is there not Wumpus in [3,1] or [2,2]”

(ask kb1 "~W31")

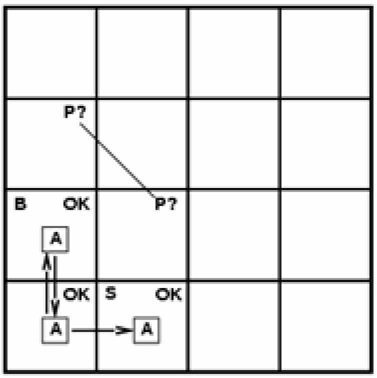
(ask kb1 "~P31")

(ask kb1 "~W22")

(ask kb1 "~P22")

The answers should not be all yes, so that we can not move to square [3,1] or [2,2], therefore we move back to squre [1,1] and then [1,2].

When we are at square [1,2], there is a Stench, no Breeze:



We tell the KB that information:

(tell kb1 "S12")

(tell kb1 "~B12")

We also tell the KB the related Wumpus rules:

(tell kb1 "S12 => W11 | W22 | W13")

(tell kb1 "~B12 => ~P11 ^ ~P22 ^ ~P13")

Now we want to know whether we can move to square [2,2] or [1,3].

CG-USER(21): (ask kb1 "~W22")

CG-USER(22): (ask kb1 "~P22")

(ask kb1 "~W13")