Foundations of Artificial Intelligence

CS472/3

Lecture #24

Bart Selman

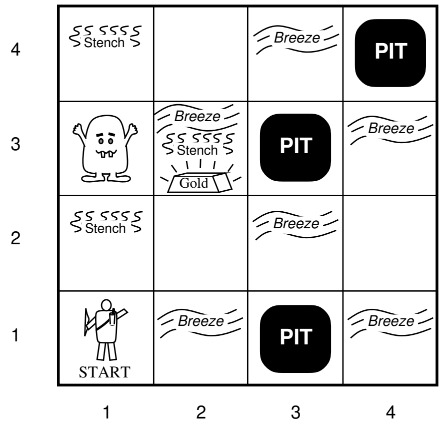
Yannis Vetsikas

Today's Lecture

Knowledge Representation & Reasoning, cont.

Chapter 6 & 7, R&N.

Modeling Agent in Grid World

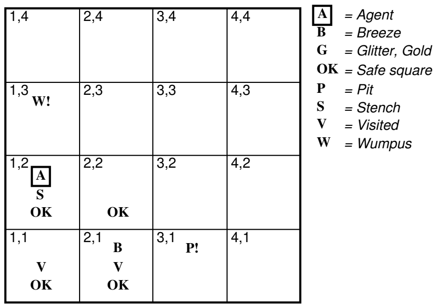


"Wumpus World" R&N, p154.

Slide CS472–3

Percept — five symbols.
E.g. [*Stench, Breeze, Glitter, None, None*]
Exercise in encoding. Important throughout AI (learning, planning etc.)

Slide CS472–4



after three moves

$\neg S_{1,1}, \neg S_{2,1}, S_{1,2}, \neg B_{1,1}, B_{2,1}, \neg B_{1,2}$

S for "stench"; B for breeze.

Slide CS472–5

Rules for "relevant" squares:

R1: $\neg S_{1,1} \Rightarrow (\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1})$ becomes

R1a: $S_{1,1} \vee \neg W_{1,1}$ (for $\neg S_{1,1} \Rightarrow \neg W_{1,1}$)

R1b: $S_{1,1} \vee \neg W_{1,2}$ R1c: $S_{1,1} \vee \neg W_{2,1}$

R2a: $S_{2,1} \vee \neg W_{1,1}$ R2b: $S_{2,1} \vee \neg W_{2,1}$

R2c: $S_{2,1} \vee \neg W_{2,2}$ R2d: $S_{2,1} \vee \neg W_{3,1}$

R3a: $S_{1,2} \vee \neg W_{1,1}$ R3b: $S_{1,2} \vee \neg W_{1,2}$

R3c: $S_{1,2} \vee \neg W_{2,2}$ R3d: $S_{2,1} \vee \neg W_{1,3}$

R4: $\neg S_{1,2} \vee W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$

R&N p. 174-175.

Slide CS472–6

Note many rule. E.g. just R3, gives up to 4 clauses **per square**.

Ex. First-order:

$\forall l_1, s, Smelly(l_1) \Rightarrow$

$(\exists l_2 At(Wumpus, l2, s) \wedge (l_2 = l_1 \vee Adjacent(l_1, l_2))$

How would you define $Adjacent$?

How about:

$\forall i, j, , s \;\; 1 \leq i, j \leq N \;.\; Smelly(i, j) \Rightarrow$

$(\exists k, l \;.\; -1 \leq k, l \leq 1 \quad At(Wumpus, i + k, j + l, s)$

What about epressiveness? What is implicitly used? Computation?

Slide CS472–7

"Finding the Wumpus"

What holds in Wumpus KB? (12 symbols — $S_{1,1}, S_{2,1}$ etc.

$2^{12} = 4096$, truth assignments

To prove: $W_{1,3}$

Use resolution.

Compare with section 6.5 R&N for inference rule approach.

Slide CS472–8

1) $\neg S_{1,1}$ (in KB / fact)
2) $\neg W_{1,1}$ (resolution with R1a)
3) $\neg S_{2,1}$ (in KB / fact)
4) $\neg W_{1,2}$ (resolution with R1b)
5) $\neg W_{2,2}$ (res with R2c)
6) $S_{1,2}$ (in KB / fact)
7) $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
8) $W_{1,3} \vee W_{1,2} \vee W_{2,2}$
9) $W_{1,3} \vee W_{1,2}$
10) $W_{1,3}$
11) $\neg W_{1,3}$ (our assumption)
12) $\emptyset$ (contradiction: it follows $W_{1,3}$ **must be True.**)

Why do things this way?

Contrast with **simulation** approach.

Basic idea: axiomatic / knowledge-based / declarative approach
allows us a very large range of **queries / conclusions.**
Procedural (e.g., standard program, "simulation") would
require separate procedure for each possible question (almost).
Of course, getting the axioms / facts right can be tricky!
And, reasoning can be computationally very hard!

Slide CS472–11

So far, most success in a restricted context:
e.g. "expert systems" knowledge encoded in basic if-then rules
(similar to Horn sentences: $(p \wedge q \wedge r) \Rightarrow s$)
use forward / backward chaining (Section 9.4. R&N).
Also, more recently in **diagnosis**. (Xerox / NASA)

And, more specialized KR languages, e.g., frame-based systems
descriptin logics. (AT&T).
Tens of thousands of facts / thousands of frames ("objects").

Slide CS472–12

Modeling actions: Simple Reflex Agent

Directly connects percepts to actions:

$\forall s, b, u, c, t \quad \mathit{Percept}([s, b, \mathit{Glitter}, u, c], t) \Rightarrow \mathit{Action}(\mathit{Grab}, t)$

Or, more indirectly:

$\forall s, b, u, c, t \quad \mathit{Percept}([s, b, \mathit{Glitter}, u, c], t) \Rightarrow \mathit{AtGold}(t)$

$\forall t \quad \mathit{AtGold}(t) \Rightarrow \mathit{Action}(\mathit{Grab}, t)$

Why more flexible? *Limitations of reflex approach?*

Representing Change / Time

How did we represent change so far? E.g., in our (propositional) Wumpus world.

Agent would move around. Captured how?

We would add sentences to the KB.

Want more general / elegant approach.

For example: We want to make statements **about** possible changes in the world. (Why?)

Situation Calculus

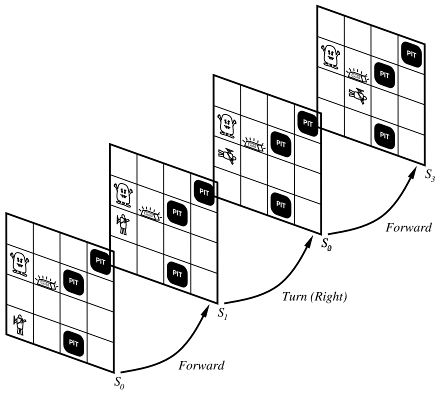
One approach: have time argument

We can be more concise, since we're only interested in when and how things **change**.

Focus on **"situations". ("snapshots")**

(John McCarthy 1963).

(Changing) World is represented by a series of *situations*.



Slide CS472–17

E.g.:

$At(Agent, [1, 1], S_0) \wedge At(Agent, [1, 2], S_1)$

"talking" about change / actions:

$Result(Forward, S_0) = S1$
$Result(Turn(Right), S_1) = S2$
$Result(Forward, S_2) = S3$

Is *Result* a relation or a function?
What about *Forward*?

Slide CS472–18

Effect Axioms

$Portable(Gold)$
$\forall s \quad AtGold(s) \Rightarrow Present(Gold, s)$
$\forall x, s \quad (Present(x, s) \land Portable(x)) \Rightarrow Holding(x, Result(Grab, s))$
$\forall x, s \quad \neg Holding(x, Result(Release, s))$

Does this work?

Frame Axioms

Need also to state what **doesn't** change!

$$\forall a, x, s \quad Holding(x, s) \land (a \neq Release)) \Rightarrow Holding(x, Result(a, x))$$

$$\forall a, x, s \quad (\neg Holding(x, s) \land (a \neq grab)) \Rightarrow \neg Holding(x, Result(a, x))$$

More compactly:

$\forall a, x, s \;\; Holding(x, Results(a, s)) \Leftrightarrow$
$\quad [(a = Grab \wedge Present(x, s) \wedge Portable(x))$
$\quad \vee (Holding(x, s) \wedge a \neq Release)]$

successor-state axioms: need to list all the ways in wich any predicate can become true / false.

Slide CS472–21

Frankly, representing and dealing with dynamic / changing is not a "strong point" of first-order logic.

- work on different logics:
  e.g. dynamic logic / nonmonotonic logic.

nonmon: long "struggle". Yale shooting problem (warn.: R):

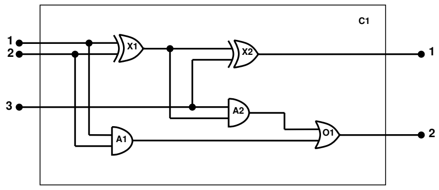
`load gun / point gun / wait 5 seconds / fire gun`

question: target dead? about 100+ research papers . . ., since 1986; still not fully resolved.

first-order logic better at "static" information.

Slide CS472–22

Another example: Electronic Circuits



Section 8.3 R&N.

Formalization

one-bit full adder: two inputs and a carry / one output and carry
 four gates: AND, OR, XOR and NOT.
Goal: analyze design to see if it matches specification.
Consider: circuits (gates and gate types), terminals, and signals.
 formalization — keep task in mind.
 e.g., for fault diagnosis: might want to specify "wires"
  could be broken... (e.g., $Wire(x, y)$)

Vocabulary

pick: functions / predicates / constants.

  constant symbols: X_1, X_2, etc.

  type gate: $Type(X_1) = XOR$, note XOR new constant.

    Alt.: $Type(X_1, XOR)$. Q. Advantage function?

  terminals: $Out(1, X_1), In(1, X_1), In(2, X_1)$.

connectivity: $Connected(Out(1, X_1), In(1, X_2))$.

  Note: we don't have to **name** the terminals explicity.

    the semantics of the function will assign some unique "object" to it.

    (Skolemization can bring back name.)

signal values: function $Signal(x)$, e.g., $Signal(In(1, X_1))$.

signal values: On and Off.

General Rules

how signals behave:

1) $\forall t_1, t_2 \;\; Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$

2a) $\forall t \;\; Signal(t) = On \vee Signal(t) = Off$

2b) $On \neq Off$

3) $\forall t_1, t_2 \;\;\; Connected(t_1, t_2) \Leftrightarrow Connected(t_2, t_1)$

Slide CS472–27

how gates behave:

4) $\forall g \;\; Type(g) = OR \Rightarrow$

$$Signal(Out(1, g)) = On \Leftrightarrow \exists n \;\; Signal(In(n, g) = On.$$

5) how AND?

6) $\forall g \;\; Type(g) = XOR \Rightarrow$

$$(Signal(Out(1, g)) = On \Leftrightarrow (Signal(In(1, g) \neq Signal(In(2, g))))$$

7) NOT, similarly.

Slide CS472–28

few rules (7): good ontology
clear rules: good vocabulary
what remains?

atomic facts — describing actual circuit under consideration.
types of gates:
$Type(X_1) = XOR, \;\; Type(X_2) = XOR, \;\; Type(A_1) = AND, \ldots$
conectivity:

$Connected(Out(1, X_1), \;\; In(1, X_2)),$
$Connected(In(1, C_1), \;\; In(1, X_1)),$
$Connected(Out(1, X_1), \;\; In(1, A_2))$
$Connected(In(1, C_1), \;\; In(1, A_1)),$

etc.

Queries

Our theory captures full behavior.

Can now ask many different queries about behavior etc.

E.g.

$\exists i_1, i_2, i_3 \;\; Signal(In(1, C_1)) = i_1 \wedge Signal(In(2, C_1)) = i_2 \wedge Signal(In_3, C_1)) = i_3$
$\wedge Signal(Out(1, C_1)) = Off \wedge Signal(Out(2, C_1)) = On$

A.: $(I_1 = On \wedge i_2 = On \wedge i_3 = Off) \vee$
$(I_1 = On \wedge i_2 = Off \wedge i_3 = On) \vee$
$(I_1 = Off \wedge i_2 = On \wedge i_3 = On)$

What is the advantage over direct simulation?

Slide CS472–31

Of course, formalization is somewhat facilitated by the "closeness" between logical formalisms and digital circuitry. Starting with Shannon, allows for very powerful design methods (but did not prevent Pentium bug...).

Slide CS472–32