Foundations of Artificial Intelligence

CS472/3 — Fall 1999

Lecture #18 & 19

Bart Selman

Slide CS472–1

Today's Lecture

K&R / Knowledge-Based Systems

Chapter 7 & 8, R&N.

Slide CS472–2

Example: N-Queens, first-order

- **1) no row with two queens**

  $$\forall i,j,k \;\; (1 \leq i,j,k \leq N) \;\; (Queen(i,j) \wedge Queen(i,k)) \Rightarrow (j = k)$$

- **2) no column with two queens**

  $$\forall i,j,k \;\; (1 \leq i,j,k \leq N) \;\; (Queen(j,i) \wedge Queen(k,i)) \Rightarrow (j = k)$$

Slide CS472–3

- **3) no diagonal with two queens**

  $$\forall i,j,k,l \;\; (1 \leq i,j,k,l \leq N) \;\; [(Queen(i,j) \wedge Queen(k,l) \wedge leftdiagonal(i,j,k,l)] \Rightarrow ((i = j) \wedge (k = l))$$

similarly for right diagonal.

Slide CS472–4

$$\forall i,j,k,l \ \ (1 \le i,j,k,l \le N)$$
$$leftdiagonal(i,j,k,l) \Leftrightarrow [(i-j)=(k-l)]$$

Similarly, for $rightdiagonal$

Complete?

- **4) at least one queen per row**

  $\forall i \ \exists j \ \ (1 \le i,j \le N) \ \ Queen(i,j)$

Note **finite domain:** finite propositional encoding exists.
Often be best for computer to reason with!

Interesting (first-order) consequence:

- at least one queen per column:
  $\forall i \ \exists j (1 \leq i, j \leq N) \ \ Queen(j, i)$

How do we know? How can we derive it automatically?

Simple Reflex Agent

Directly connects percepts to actions:
$\forall s, b, u, c, t \quad Percept([s, b, Glitter, u, c], t) \Rightarrow Action(Grab, t)$

Or, more indirectly:

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

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

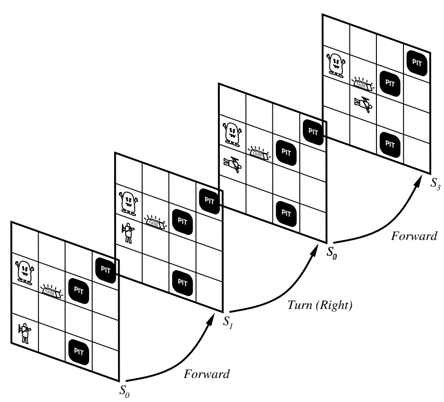
We can be more concise, since we're only interested in when and how things **change**.

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

(John McCarthy 1963).

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

Slide CS472–11



Slide CS472–12

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 CS472–13

Effect Axioms

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

Does this work?

Slide CS472–14

Frame Axioms

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

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

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

Slide CS472–15

More compactly:

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

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

Slide CS472–16

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):

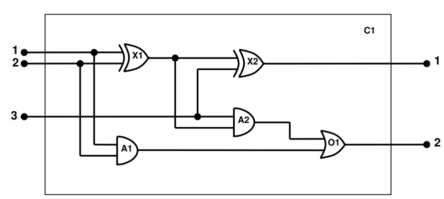
`load gun / point gun / wait 5 seconds / fire gun`

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

first-order logic better at "static" information.

Slide CS472–17

Another example: Electronic Circuits



Section 8.3 R&N.

Slide CS472–18

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., $Wire(x, y)$)

Slide CS472–19

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

Slide CS472–20

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.

Slide CS472–21

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) $\forall t_1, t_2 \quad Connected(t_1, t_2) \Leftrightarrow Connected(t_2, t_1)$

Slide CS472–22

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 CS472–23

few rules (7): good ontology

clear rules: good vocabulary

what remains?

Slide CS472–24

atomic facts — describing actual circuit under consideration.

types of gates:

$Type(X_1) = XOR$, $Type(X_2) = XOR$, $Type(A_1) = AND, ..$

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.

Slide CS472–25

Queries

Our theory captures full behavior.

Can now ask many different queries about behavior etc.

E.g.

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

A.: $(I_1 = On \land i_2 = On \land i_3 = Off) \lor$
$(I_1 = On \land i_2 = Off \land i_3 = On) \lor$
$(I_1 = Off \land i_2 = On \land i_3 = On)$

What is the advantage over direct simulation?

Slide CS472–26

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...).