Foundations of Artificial Intelligence

CS472/3
Lecture #27

Bart Selman

Today's Lecture

Knowl. Repr., cont.
Inference, cont.
Chapter 8 & 9, R&N.

Last time: Formal Representation of Circuit

- circuit diagram captured in
  general rules about gates, signals, and connecting terminals
    E.g. $\forall t_1, t_2 \quad Connected(t_1, t_2) \Rightarrow$
      $Signal(t_1) = Signal(t_2)$
  with atomic facts given the actual circuit
    E.g., $Type(A2) = AND$.

Slide CS472–3

Allows us to **reason** about overall behavior.
  E.g. What inputs give a particular output.
Used in analysis of circuits/systems.
  Contrast with Truth table method.
And querying (What logically follows
  from specification.).

Slide CS472–4

Diagnosis

So far, discussed checking specification.

How about diagnosing circuit?

I.e., given inputs and outputs, find faulty gates.

Slide CS472–5

Why bother?

Don't you normally just throw the chip out??

A. Still need **good** testing patterns.

I.e., ones with large coverage,

Method applicable at higher level, i.e., in terms of larger components.

Also other domains:

- Cassini mission to Saturn (on its way...)
  real-time diagnosis / **25 msec decisions!**
  2,000 vars and 10,000 clauses SAT problem
- Software diagnosis / synthesis
  Components are subroutines / classes.

Slide CS472–6

Many different possibilities...

In practice, rank according to likelihood.

E.g. single-fault more likely than double fault.

Also, incorportate failure rates of components.

(reasoning with uncertainty)

Slide CS472–7

For example, what happens if we assert with our initial KB that:

$Signal(In(1, C1)) = On, \quad Signal(In(2, C1)) = On$

$Signal(In(3, C1)) = On$

$Signal(Out(1, C1)) = On, \quad Signal(Out(2, c1)) = Off$?

Slide CS472–8

Need to capture possibly faulty behavior.

$4'$) $\forall g \;\; ((Type(g) = OR) \wedge Functioning(g)) \Rightarrow$
$(Signal(Out(1, g)) = On \Leftrightarrow$
$\exists n \;\; Signal(In(n, g) = On)$

Slide CS472–9

KB with $\forall x \quad Functioning(x)$ with

$Signal(In(1, C1)) = On, \quad Signal(In(2, C1)) = On$
$Signal(In(3, C1)) = On$

implies

$Signal(Out(1, C1)) = On, \quad Signal(Out(2, c1)) = On.$

Slide CS472–10

So, if we **observe**
$Signal(Out(1, C1)) = On, \quad Signal(Out(2, c1)) = Off.$

It follows that one or more of the $Functioning()$ predicates must be **false**.

Finding the smallest set of non-functioning components consistent with the observations is called **abductive** or **diagnostic** reasoning.

Slide CS472–11

Which component(s) is (are) non-functioning?

Aside: abductive reasoning is generally computationally harder than deductive reasoning.

Still, currently abductive reasoning more widely applied because of the need for constant monitoring of complex systems.

Slide CS472–12

General Logical Inference

Relatively straightforward KB can quickly
overwhelm general resolution methods.
Resolution strategies reduce the problem somewhat,
but not completely.
As a consequence, many **practical** Knowledge Represenation
formalisms in AI use a **restricted form**
and **specialized inference.**
Can often understand them in terms of standard
first-order logic! (clear **syntax & semantics**)

KR Systems (Chapt. 10, R&N)

- **Theorem provers / logic programming**
- **Production systems**
  forward chaining / if-then-rules / **expert** systems
- **Frame systems and semantic networks**
- **Description logics**

Theorem provers / logic programming

Theorem provers: generally based on resolution
many different strategies to improve efficiency
Logic programming: program statements directly in
restricted FOL (Horn clauses).
Execution: search for proof of goal/query
using backward chaining with depth first-search.
In certain cases too inefficient.

Production systems

- rich history in AI
- **"expert system"** boom in 70's / 80's

Basic idea:
capture knowledge of human expert in a
large set of "if-then" rules
(really, logical implication $\Rightarrow$)
"production rules"

Example: Car Diagnosis

```
Rule 1
   If   the engine is getting gas, and
        the engine will turn over,
   then
        the problem is spark plugs.

Rule 2
   If   the engine does not turn over, and
        the lights do not go on
   then
        the problem is battery or cables.
```

Slide CS472–17

```
Rule 3
   If   the engine does not turn over, and
        the lights do come on
   then
        the problem is the starter motor.

Rule 4
   If   there is gas in the fuel tank, and
        there is gas in carburetor
   then
         the engine is getting gas.
```

Slide CS472–18

A goal-driven or backward chaining expert system.
Start with goal-expression e.g. "$\exists X \;\; Problem(X)$"
    Match with one of the rules (1, 2, or 3)
    Depth-first backward, so pick 1.
    Place premises in working memory.
        Try to prove premises.
        E.g. use rule 4.
        Until base facts can be observed.

Slide CS472–19

E.g.:
    Problem (X)
    Problem (spark plugs) [rule 1]
    engine getting gas & engine turns over (OK)
    gas in fuel tank (OK) & gas in carb. (OK) &
        engine turns over (OK) [rule 4]

bottom facts confirmed; problem "spark plugs" found.

What if one or more bottom facts were not confirmed?

Slide CS472–20

Follows roughly the process of hypothesis testing in human problem solving.

Could it be this? Could it be that? Is so, then this must hold. Otherwise, that must be true. Etc. Etc.

Chaining through rules; either backwards or forwards.
Goes back to early cognitive studies of human problem solving behavior (Newell and Simon 1961; 1972).

Slide CS472–21

Rich history. Led to quite successful **expert systems** (concrete examples later)
Bottleneck: Getting the rules!
The **Knowledge Acquisition Problem.**
(all the knowledge is in the rules)

Slide CS472–22

How does this form of diagnosis differ from the
earlier circuit diagnosis example?
[logical specification circuit
I/O facts
minimal fault set to restore consistency]

Slide CS472–23

Circuit example: diagnosis from **first-principles**
use full description of circuit — **deep knowledge**.

Here **"shallow knowledge".**
no full model of car.
direct encoding of **diagnostic rules**.
[**weak methods** use weak info about domain]
See also chapter 1, R&N.

Slide CS472–24

Deep Blue vs. Kasparove

Which one from "First Principles"?

Slide CS472–25

Success in two opposite directions

1) Computers appear **better at** "reasoning from first principles." (possibly because of raw speed)
   Approach reduces the Knowledge Acquisition Problem.

2) Rule-based approach to the extreme:
   Case-Based Reasoning Systems
   If-then rules, but no chaining; match current situation to a known previous case & do what you did then (single rule).
   E.g. Store 50,000+ car repair scenarios
   use **closest match**

Slide CS472–26

Successes in Rule-Based Reasoning

Expert systems

- DENDRAL (Buchanan *et al.*, 1969)
- MYCIN (Feigenbaum, Buchanan, Shortliffe ca. 1972)
- PROSPECTOR (Duda *et al.*, 1979)
- R1 (McDermott, 1982)

page 22 & 23 R&N.

- DENDRAL (Buchanan *et al.*, 1969)
  - infers molecular structure from the information provided by a mass spectrometer
  - ```
    if  there are peaks at x_1 and x_2 s.t.
       x_1 + x_2 = M + 28
       x_1 - 28 is a high peak
       x_2 - 28 is a high peak
       At least one of x_1 and x_2 is high
    then there is a ketone subgroup
    ```
    ```

- MYCIN (Feigenbaum, Buchanan, Shortliffe)
  - diagnosis of blood infections
  - 450 rules; performs as well as experts
  - incorporated **certainty factors**

```
If: (1) the stain of the organism is
        gram-positive, and
    (2) the morphology of the organism is
        coccus, and
    (3) the growth conformation of the organism
        is clumps,
then there is suggestive evidence (0.7) that the
  identity of the organism is staphylococcus.
```

Slide CS472–29

- PROSPECTOR (Duda *et al.*, 1979)
  - correctly recommended exploratory drilling at a geological site
  - rule-based system founded on probability theory
- R1 (McDermott, 1982)
  - designs configurations of computer components (DEC)
  - about 10,000 rules
  - by 1986, saving company about $40 million a year

Slide CS472–30

Cognitive Modeling with Rule-Based Systems

SOAR is a general architecture for building intelligent systems.

- Long term memory consists of rules.
- Working memory describes current state.
- All problem solving, including deciding what rule to execute, is state space search.
- Successful rule sequences are *chunked* into new rules.
- Control strategy embodied in terms of meta-rules.

What's the difference between rule-based systems and
general theorem proving systems?
(**Horn clause** sentences only.)

Also, forward-chaining production systems
no queries; rules are constinuously fired based
on current state of the knowledge-base.
good for modeling actions / agents.

Advantages / disadvantages of rule-base approach?

What happened to expert systems?