7. Rules in Production Systems

Direction of reasoning

A conditional like \( P \Rightarrow Q \) can be understood as transforming
- assertions of \( P \) to assertions of \( Q \)
- goals of \( Q \) to goals of \( P \)

Can represent the two cases explicitly:
\[
\begin{align*}
\text{assert } P & \Rightarrow \text{assert } Q \\
\text{goal } Q & \Rightarrow \text{goal } P
\end{align*}
\]

and then distinguish between

1. goal vs. data directed reasoning
   - goal: from \( Q \) towards \( P \)
   - data: from \( P \) towards \( Q \)

2. forward vs. backward-chaining
   - forward: along the \( \Rightarrow \)
   - backward: against the \( \Rightarrow \)

Possible to have
- (proc if-added (mygoal \( Q \)) ... (mygoal \( P \)))
- (proc if-needed (myassert \( P \))... (myassert \( Q \)))

How to do data-directed reasoning in Prolog

Now: a formalism with forward-chaining
Production systems

Idea: working memory + production rule set

Working memory: like DB, but volatile

Production rule: IF conditions THEN actions
  condition: tests on WM
  action: changes to WM

Basic operation: cycle of

1. recognize
   find conflict set: rules whose conditions are satisfied by current WM
2. resolve
   determine which of the rules will fire
3. act
   perform required changes to WM

Stop when no rules fire

Working memory

Set of working memory elements (WME)

Each WME is of the form (type attr\_1 val\_1 attr\_2 val\_2 ... attr\_n val\_n)

where type, attr\_i, val\_i are all atoms

Examples: (person age 27 home Toronto)
  (goal task openDoor importance 5)
  (student name JohnSmith dept CS)

Understood as \( \exists x [\text{type}(x) \land attr\_1(x)=val\_1 \land ... \land attr\_n(x)=val\_n] \)

- individual is not explicitly named
- order of attributes is not significant

Can handle n-ary relations as usual

  (myAssertion relation OlderThan firstArg John secondArg Mary)
Rule conditions

Conditions: tested conjunctively

a condition is $p$ or $\neg p$, where $p$ is a pattern of the form

\[(\text{type } \text{attr}_1 \text{spec}_1 \ldots \text{attr}_k \text{spec}_k)\]

where each specification must be one of

- an atom
- an expression within [ ]
- a variable
- a test, within {}
- the $\land$, $\lor$, $\neg$ of a specification

Examples:

\[(\text{person } \text{age } \lceil n+4 \rceil \text{ occupation } x)\]
\[\neg (\text{person } \text{age } \{< 23 \land > 6\})\]

A rule is applicable if there are values of the variables to satisfy all the conditions

- for a pattern, need WME of the correct type and for each \text{attr} in pattern, \text{val} must match \text{spec}
- for $\neg p$, there must be no WME that matches $p$  \therefore negation as failure

Rule actions

Actions: performed sequentially

An action is of the form

- ADD \text{pattern}
- REMOVE \text{index}
- MODIFY \text{index} (\text{attr spec})

where

- \text{index} \text{i} refers to the WME that matched \text{i}-th pattern (inapplicable to $\neg p$)
- variables and expressions refer to values obtained in the matching

Examples:

\[
\begin{align*}
\text{IF } \text{(Student name } x) \\
\text{THEN } \text{ADD } \text{(Person name } x) \\
\text{ordinary forward chaining}
\end{align*}
\]

\[
\begin{align*}
\text{IF } \text{(Person age } x) \text{ (Birthday)} \\
\text{THEN } \text{REMOVE 2} \\
\text{MODIFY 1 (age } [x+1]) \\
\text{database update}
\end{align*}
\]

\[
\begin{align*}
\text{IF } \text{(starting)} \\
\text{THEN } \text{REMOVE 1} \\
\text{ADD (phase val 1)} \text{ control information}
\end{align*}
\]
Example 1

Placing bricks in order of size
largest in place 1, next in place 2, etc.

Initial working memory

\[
\begin{align*}
\text{(counter index 1)} \\
\text{(brick name A size 10 place heap)} \\
\text{(brick name B size 30 place heap)} \\
\text{(brick name C size 20 place heap)}
\end{align*}
\]

Production rules:

- **IF** (brick place heap name \( n \) size \( s \))
  - (brick place heap size {> \( s \)})
  - (brick place hand)
- **THEN** MODIFY 1 (place hand)

- **IF** (brick place hand) (counter index \( i \))
- **THEN** MODIFY 1 (place \( i \))
- **MODIFY 2** (index \([i+1]\))

Trace

Only one rule can fire at a time, so no conflict resolution is required

The following modifications to WM

1. (brick name B size 30 place hand)
2. (brick name B size 30 place 1) (counter index 2)
3. (brick name C size 20 place hand)
4. (brick name C size 20 place 2) (counter index 3)
5. (brick name A size 10 place hand)
6. (brick name A size 10 place 3) (counter index 4)

So the final working memory is

\[
\begin{align*}
\text{(counter index 4)} \\
\text{(brick name A size 10 place 3)} \\
\text{(brick name B size 30 place 1)} \\
\text{(brick name C size 20 place 2)}
\end{align*}
\]
Example 2

How many days are there in a year?

Start with: (want-days year n)
End with: (has-days days m)

1. \textbf{IF} (want-days year n)
   \textbf{THEN REMOVE 1}
   \textbf{ADD} (year mod4 [n mod 4]
                             mod100 [n mod 100]
                             mod400 [n mod 400])

2. \textbf{IF} (year mod400 0)
   \textbf{THEN REMOVE 1 ADD} (has-days days 366)

3. \textbf{IF} (year mod100 0 mod400 (≠ 0))
   \textbf{THEN REMOVE 1 ADD} (has-days days 365)

4. \textbf{IF} (year mod4 0 mod100 (≠ 0))
   \textbf{THEN REMOVE 1 ADD} (has-days days 366)

5. \textbf{IF} (year mod4 (≠ 0))
   \textbf{THEN REMOVE 1 ADD} (has-days days 365)

Applications

1. Psychological modeling

   \textbf{IF} (goal is get-unit-digit)
   (minuend unit d)
   (subtrahend unit (> d))

   \textbf{THEN REMOVE 1}
   \textbf{ADD} (goal is borrow-from-tens)

2. Expert systems

   rules used by experts in a problem area to perform complex tasks
   \textit{(examples later)}

Claimed advantages:

- modularity: each rule acts independently of the others
- fine-grained control: no complex goal or control stack
- transparency: can recast rules in English to provide explanation of behaviour
MYCIN

System developed at Stanford to aid physicians in treating bacterial infections

Approximately 500 rules for recognizing about 100 causes of infection

IF

the type of $x$ is primary bacteremia
the suspected entry point of $x$ is the gastrointestinal tract
the site of the culture of $x$ is one of the sterile sites

THEN

there is evidence that $x$ is bacteroides

Certainty factors

numbers from 0 to 1 attached to conclusions to rank order alternatives

AND – take min  OR – take max

XCON

System developed at CMU (as R1) and used extensively at DEC (now owned by Compaq) to configure early Vax computers

Nearly 10,000 rules for several hundred component types

Major stimulus for commercial interest in rule-based expert systems

IF

the context is doing layout and assigning a power supply
an sbi module of any type has been put in a cabinet
the position of the sbi module is known
there is space available for the power supply
there is no available power supply
the voltage and the frequency of the components are known

THEN

add an appropriate power supply
Context switching

XCON and others use rules of the form

\[
\text{IF} \quad \text{the current context is } x \\
\text{THEN} \quad \text{deactivate } x \\
\quad \text{activate context } y
\]

organized to fire when no other rules apply

Useful for grouping rules

\[
\begin{align*}
\text{IF} \quad \text{(control phase 1) AND ...} \\
\text{THEN} \quad \text{...} \\
\text{...} \\
\text{IF} \quad \text{(control phase 1) AND ...} \\
\text{THEN} \quad \text{MODIFY 1 (phase 2) ...} & \quad \text{Allows emulation of control structures.} \\
\text{...} & \quad \text{But still difficult for complex control} \\
\text{IF} \quad \text{(control phase 2) AND ...} \\
\text{THEN} \quad \text{...} \\
\text{...} \\
\text{IF} \quad \text{(control phase 2) AND ...} \\
\text{THEN} \quad \text{MODIFY 1 (phase 3) ...}
\end{align*}
\]

Conflict resolution

Sometimes with data-directed reasoning, we want to fire \textit{all} applicable rules

With goal-directed reasoning, we may want a \textit{single} rule to fire

- arbitrary
- first rule in order of presentation (as in Prolog)
- specificity, as in
  \[
  \text{IF} \quad \text{(bird)} \quad \text{THEN ADD (can-fly)}
  \]
  \[
  \text{IF} \quad \text{(bird weight \{> 100\})} \quad \text{THEN ADD (cannot-fly)}
  \]
  \[
  \text{IF} \quad \text{(bird) (penguin)} \quad \text{THEN ADD (cannot-fly)}
  \]
- recency
  - fire on rule that uses most recent WME
  - fire on least recently used rule
- refractoriness
  - never use same rule for same value of variables (called rule instance)
  - only use a rule/WME pair once (will need a “refresh” otherwise)
Conflict combinations

OPS5:
1. discard rule instances that have already been used
2. order remaining instances in terms of recency of WME matching 1st condition (and then of 2nd condition, etc.)
3. if still no single rule, order rules by number of conditions
4. select arbitrarily among those remaining

SOAR:

system that attempts to find a way to move from a start state to a goal state by applying productions

selecting what rule to fire
≡
deciding what to do next

if unable to decide, SOAR sets up the selection as a new (meta-)goal to solve, and the process iterates

Rete procedure

Early systems spent 90% of their time matching, even with indexing and hashing.

But: • WM is modified only slightly on each cycle
  • many rules share conditions

So: • incrementally pass WME through network of tests
  • tokens that make it through satisfy all conditions and produce conflict set
  • can calculate new conflict set in terms of old one and change to WM

IF (Person father y age {< 14} name x) (Person name y occupation doctor)
THEN ...