Artificial Intelligence -- 15-381
Rule-Based Systems

Jaime Carbonell
17-February-2000

OUTLINE:

- Rules definions and matching
- Knowledge acquisition
  via dual-phase protocol analysis
- Real-time systems
  including agenda-based control

Rule-Based Systems:
OPS-Style Forward Chaining

RULE SYNTAX

- Unordered set of CONDITION => ACTION rules
- CONDITION is in conjunctive normal form (CNF)
  - Each condition is typically a FOL literal
  - Variables are existentially quantified
  - Variables are bound in the match process
  - Variable bindings must be consistent across conditions and actions
  - All legal matches (rule-to-WM) must be found
    (each match is a *rule instance*)
- ACTION is a sequence of atomic actions
  - Each is typically expressed as an FOL literal
  - Variables take their bindings from matched conditions

RULES [Continued]

WORKING MEMORY (WM) SYNTAX

- Unordered set of grounded literals (no variables)

EXAMPLES OF RULES

```
time(t) &
madeof(x, GLASS) &
shape(x, CONCAVE) &
NOT(polished(x)) &
fine-grinder(y) &
available(y,t)
calibrated(y)
  =>
fine-grind(x,y,t) &
NOT(available(y,t))
```

```
time(t) &
fine-grinder(y) &
available(y,t) &
not-calibrated(y)
  =>
calibrated(y,t) &
NOT(available(y,t))
```

EXAMPLES OF WM

```
madeof(primary-mirror, glass)
madeof(secondary-mirror, glass)
madeof(eyepiece, glass)
polished(eyepiece)
time(t0)
time(t1)
time(t2)
time(t3)
fine-grinder(machine1)
coarse-grinder(machine2)
polisher(machine3)
lathe(machine4)
```

```
available(t0, machine1)
available(t0, machine2)
available(t1, machine1)
available(t1, machine3)
available(t1, machine4)
available(t2, machine3)
...
not-calibrated(machine1)
not-calibrated(machine2)
```

MATCHING IN RULE-BASED SYSTEMS

ASSUMPTIONS:

- N Rules and M WM elements
- C_R conditions per rule
- V_R variables per rule

COMPLEXITY OF MATCH

- CASE 1: No variables
- CASE 2: Every variable occurs only once in C_R
- CASE 3: Any variable may occur multiple times

EXAMPLES

```
Rule:   P(x) & P(y) & P(z)  =>  Q(x,y)


WM:     P(a), P(b), P(c), P(d)
-------------------------------------------

Rule:   R(x,y,z) & Q(y,w) & S(w,x) => P(z)


WM: R(a,b,c), R(b,c,d), R(c,d,e)
    Q(c,f), Q(a,e), Q(f,b)
    S(f,c), S(f,b)
```

Reducing Match Complexity

INDEXING

- Multiple rules with common conditions
- Constraint propagation methods
  (forward pointer to future lecture)
- Condition reordering at compile time
  (Most restrictive conditions first)

EXPLOITING INCREMENTAL CHANGE

- Most of WM remains the same after executing C=>A
- Propagate change via dependency network
- RETE Algorithm (Forgy)
- MATCHBOX Algorithm (Perlin)

Two-Phase
Protocol Analysis

Objectives:

- Acquire factual and tacit knowledge from human expert
- Maximize information throughput per unit time
- Structured KA process around natural expert task performance
- Incremental process with maximal feedback

Background:

- Outgrowth of Protocol Analysis in Cognitive Psychology
- Many different variations on the theme
- Proven successful in rule-based expert systems

Two-Phase Protocol Analysis: Set-up Process

Preliminary Study:

- Read relevant material to task domain
- Observe expert(s)
- Get a feel for typical and hard tasks

Task Selection:

- Select pie-slice of domain (deep and narrow) in consultation with the expert
- Acquire typical problems (solved in the past, new ones coming in, etc.) totally contained in pie-slice)
- Rank problems according to expert's estimate of complexity

Set up Recording Equipment

- Best: video camera with sound on work-surface (CRT, table, etc.), notepad. Assistant to take additional notes.
- Minimal: tape recorder and note pad.

Two-Phase Protocol Analysis: Phase I

1. Expert solves first problem
2. Expert says (loudly, clearly) every thing he or she does, step by step.
3. KE interrupts ***only if*** expert goes silent (no questions)
4. Everything is recorded, including (especially) things pondered and not done, mistakes, and successful action sequences
5. Notes (or video) record what expert does as he or she talks
6. Repeat the process for a handful of typical problems
7. These actions become the right-hand side of rules (with minimal generalization)

Two-Phase Protocol Analysis: Phase II

1. For each step in the solution of the first task:
   a. Determine (ask) ***why*** this step was taken.
   b. Ask why other (plausible) steps were *not* taken.
2. The answers to these questions become the conditions (left-hand sides) of rules.
3. Ask *what if* [something was different] questions here {These lead to further generalization, and additional conditions]
4. For each false step off the solution path in the expert protocol ask what led to the error and how it was recognized as an error later. [These lead to discover further rules, e.g. default actions.]
5. Repeat for other tasks, and generalize rules.]

Two-Phase Protocol Analysis: Continuation

Initial Validation of Rules

- Rules should be tested on all development tasks, and refined/debugged accordingly.
- Rule firing trace should be shown to expert (if possible)
- Rules should be tested on other "typical" problems [If missing knowledge, perform mini-protocol on expert.]
- Continue refining until most or all typical problem work.

Scaling up

- Use rules for complex tasks in pie-slice.
- Perform mini-protocol when they do not work
- Retrospectively analyze rule set to simplify and generalize [If expert is willing and able, seek his or her advice.]
- Move on to next pie-slice of task domain and repeat process.

Real-Time AI

Possible Operational Definitions

AI system that runs very efficiently

Decision-cycle(AI) < Action-cycle(World)

SUM Decision-cycle(AI) < MIN Action-cycle(World)

Forall(xεEvents)[dc(x,AI) + ac(x,AI) < ac(x,w)]

Need for Real-Time AI

- Robotic application (most kinds)
- Autonomous driving
- Sensor-based warning/action systems
- Self-repairing telephone or electric networks
- ATM or Credit-card authorization

The SMOKEY System

Task Definition

- Sensor-based location, prediction, control of onboard fires in Aircraft carriers.
- **Sensors:** smoke, heat, chemical analysis
- **Knowledge:** sensor topology, ship map, loc of flammables, ...
- **Actions:** evacuate and/or seal-off section, equip and send firefighters, all-clear, ...

Objectives

- Real-time reaction
- Better performance than humans
- Robust behavior

Agenda-Based Control

Agenda Data Structure

Level-1: $T_{1,1}$, $T_{1,2}$, ..., $T_{1,j}$
Level-2: $T_{2,1}$, $T_{2,2}$, ..., $T_{2,k}$
.
.
.
Level-n: $T_{n,1}$, $T_{n,2}$, ..., $T_{n,m}$

Fields in each $T_{i,j}$

```
 Name       Domain                       Range
----------------------------------------------
TRIGGER:   pat X WM X A    ->  {F, T(bindings)}
DYNAMIC:   pat X WM X sen ->  {F, T(bindings)}
----------------------------------------------
WM-UPDATE: WM X bindings  ->  WM
A-UPDATE+: A X bindings   ->  A
A-UPDATE-: A X bindings   ->  A
A-FLUSH:   Bindings       ->  A
----------------------------------------------
ACT:       Bindings       ->  World
          (X sen X WM)
----------------------------------------------
```

Agenda-Based Control [Cont.]

Task-Execution Control:

If Active($T_{i.j}$, A)
& Match($T_{i,j}$.TRIGGER, WM)
& Match($T_{i,j}$.DYNAMIC, WM, f(sensors))

THEN Execute($T_{i.j}$.ACT, bindings)
& Update($T_{i,j}$.WM, WM, binding)
& Add($T_{i.j}$.A-UPDATE+, A)
& Delete($T_{i,j}$.A-UPDATE-, A)

ELSE-IF Match($T_{i,j}$.A-FLUSH, A)
THEN Delete($T_{i,j}$.ID, A)

Note: $T_{i,j}$.A-UPDATE+ :=
(<bindings.level, bindings.task> ...)

Agenda-Level Control

- n-level priority-queue
- global priority=queue
- pseudo-agenda (linear order)

Anytime Planning

Definitions:

***Deliberative Planning* --** Think first (full plan of action), act later.

***Reactive "Planning"* --** No thinking, reflex-action only.

***Anytime Planning* --** Think exactly as long as external world permits. Deliberative planner with interrupts that always has a best-so-far

***Probabilistic Planner* --** Accounts for uncertain consequences of actions and uncertain states of the world. plan prefix.

Properties

- Deliberation => potential for subgoaling, backtracking, weighing alternatives.
- Reactivity => potential for real-time behavior
- Any-time => [At least some of] both advantages
- Anytime probabilistic planning => Optimal, but difficult [Blythe CMU]