Contents

Part I

1. Introduction ............................................................................................................. 4
2. Propositional Logic ................................................................................................. 19
3. Predicate Logic Over Finite Models ...................................................................... 173
4. Full First-Order Predicate Logic ........................................................................... 224
5. Summary of Part I .................................................................................................. 362

Part II

6. Prolog ...................................................................................................................... 375
7. A Potpourri of Subdomains .................................................................................... 411
8. SNePS ...................................................................................................................... 428
9. Belief Revision/Truth Maintenance ......................................................................... 515
10. The Situation Calculus .......................................................................................... 569
11. Summary ............................................................................................................... 588
Part III

12. Production Systems..............................................................601
13. Description Logic..............................................................610
14. Abduction.................................................................627
12 Production Systems
Architecture
Working (Short-term) Memory

Contains set (unordered, no repeats) of
Working Memory Elements (WMEs).
Each being a rather flat, ground (no variables) symbol structure.
Rule (Long-term) Memory

Contains set (unordered, no repeats) of Production Rules.
Each being a condition-action rule of form
if condition$_1$ \ldots condition$_n$ then action$_1$ \ldots action$_m$
Each condition and action being like a WME, but allowing variables (and, maybe, other expressions)
Rule Triggering

A rule if condition$_1$ ... condition$_n$ then action$_1$ ... action$_m$ is triggered if there is a substitution, $\sigma$, such that each condition$_i \sigma$ is a WME.

A single rule can be triggered in multiple ways (by multiple substitutions).
Rule Firing

A rule \textbf{if} condition$_1$ \ldots condition$_n$ \textbf{then} action$_1$ \ldots action$_m$
that is triggered in a substitution $\sigma$
fires by performing every action$_i\sigma$. 
Production System Execution Cycle

loop

Collect $\mathcal{T} = \{ r\sigma \mid r\sigma \text{ is a triggered rule} \}$

if $\mathcal{T}$ is not empty

Choose a $r\sigma \in \mathcal{T}$

Fire $r\sigma$

until $\mathcal{T}$ is empty.
Some Typical Actions

• stop
• delete a WME
• add a WME
• modify a WME
• formatted print
Conflict Resolution Strategies

Purpose: to “Choose a $r\sigma \in \mathcal{T}$”

Specificity: If the conditions of one rule are a subset of a second rule, choose the second rule. [B & L, p. 126]

Recency: Based on recency of addition or modification of WMEs, or on recency of a rule firing. [B & L, p. 126]

Refactoriness: Don’t allow the same substitution instance of a rule to fire again. [B & L, p. 127]


Page 607
The Rete Algorithm
Assumptions

Rule memory doesn’t change.
WM changes only slightly on each cycle.
WMEs are ground.
Production Systems are data-driven (use forward chaining).
Many rules share conditions.
The Rete Network

Create a network from the conditions (Like a discrimination tree) with rules at the leaves.

Create a token for each WME.

Pass each token through the network, stopping when it doesn’t satisfy a test; resuming when the WME is modified.

When tokens reach a leaf, the rule is triggered.

Kinds of branch nodes

\( \alpha \) nodes: Simple test.

\( \beta \) nodes: Constraints caused by different conditions.