OPS5 Language Introduction

Michael Mauldin
October, 1992

This document contains a sketchy description of OPS5 language features, syntax and semantics of conditions and actions. For more information, consult the OPS5 manual.

1 Production Memory

create rules with **p** (production) or **build** (later)

an OPS5 production-rule definition is a list containing

- a function call to **p**
- LHS = one or more condition elements (first not negated), each in Lisp list format.
- a separator = `-->`
- RHS = one or more actions, each in Lisp list format.

2 Sample Rule

```
;; IF    the key is on AND the engine is not turning
;; THEN  conclude that the problem is in the starting system
(p bad-starting-system
    (task ^goal diagnose)
    (fact ^name |key is off| ^value no)
    (fact ^name |engine is turning| ^value no)
    -->
    (bind <x> |problem is in starting system|)
    (make fact ^name <x> ^value yes)
    (write (crlf) Concluding <x> (crlf)))
```

3 Left-Hand Side

LHS is collection of patterns to be matched against working memory. Each pattern contains an element-class name followed by some number of LHS terms. Each term consists of an `^attribute-name` followed by a LHS-value. The LHS-value can be a

constant	in pattern `^on couch`, "couch" is a constant; in pattern `^GRE 100`, "100" is a constant;
variable	in pattern, `^Status` <n1>, "<n1>" is variable that will be bound during matching to an actual value for some element in working memory;
predicate operator	one of seven operators may precede a constant or variable: =, <>, <=>, <, <=, >=, >; the = is assumed if no operator is present;
disjunction	in the pattern `^weight << light medium` >>, "<< light medium >>" specifies that only one of the set of values, light and medium, must match; any LHS-values may be contained in the disjunction; *warning* leave spaces between values and angle brackets to avoid confusing them with variable brackets;
conjunction	in pattern `^GRE { > 600 < 800 }`, "{ > 600 < 800 }" specifies a set of value

restrictions all of which must match; any LHS-values may be contained in the conjunction;

Restrictions to predicate operators:

- <, <=, >= and > used only with numbers and with variables bound to numbers. <=> means same type, and <> means not equal.
- first occurrence of a variable cannot be preceded by any predicate other than = (first occurrence establishes binding)

A condition pattern in LHS (other than first) may be negated by putting a “-” in front of the normal pattern

Ordering of condition elements is significant in variable binding, for conflict resolution and for match efficiency

4 RHS of OPS5 Rules

- The RHS of the OPS5 rule consists of an ordered sequence of actions.
- The primitive actions that affect working memory are **make**, **modify**, and **remove**.
- The **write** action is used to output information.
- The **halt** action provides a way of explicitly stopping the firing of production rules.
- RHS can also contain functions that return values within the actions. For example, the **compute** function allows OPS5 to do arithmetic. It provides for infix evaluation of +,-,*, //, and \\ (respectively addition, subtraction, multiplication, division, and modulus). Operations are performed from right to left.
- These and other actions and functions will be demonstrated by example.

5 Specific Commands

The WATCH Command

no argument	Print current watch level (initialized to 1) unchanged
`(watch 0)`	No report of firings or changes to working memory
`(watch 1)`	Report rule name and time tags of each working memory element for each instantiation fired
`(watch 2)`	In addition to level 1 reports, give each change (add or delete) to working memory

The RUN Command

`(run)`	run until a break or halt or no rules in conflict set
`(run N)`	run N steps unless early stop as above
`(run 1)`	for single stepping

[The WM and PPWM Commands

(wm) -- list the contents of working memory, optional arguments specify time tags; if no time tags are given, shows all elements.

`(ppwm <pat>)` -- <pat> is pattern (in LHS condition form), prints all wme’s that match <pat>. No variables, predicates or special characters are allowed in in <pat>. If pattern is null, all elements are printed.

use with **cs** and **matches** to determine why a rule failed to be instantiated at the right time.

The PM Command

`(pm <args>)` -- <args> any number of rule names

The CS Command

`(cs)` -- lists each instantiated rule in conflict set, one to a line, followed by currently dominant instantiation (that is, the one to be fired on next cycle)

The MATCHES Command

`(matches <rules>)` -- prints partial matches for rules whose names are arguments. For each condition element of specified rules, time tags of matching wme's are listed, as well as intersections of partial matches.

```
(literalize number value)

(p example-rule
   (number ^value { <number-1> > 100 } )
   (number ^value { <number-2> <> <number-1> } )
   (number ^value { <number-3> < 50 } )
   -->
   (write (crlf) <number-1> <number-2> <number-3> ) )

(make number ^value 101)  ; given time-tag 1

(make number ^value 102)  ; given time-tag 2

(make number ^value  11)  ; given time-tag 3

=>(matches example-rule)

example-rule
 ** matches for (1) **
 2
 1
 ** matches for (2) **
 3
 2
 1
 ** matches for (2 1) **
 3  1
 3  2
 1  2
 2  1
 ** matches for (3)
 3
nil
```

The final intersection, which in this example would be `matches for (3 2 1)`, is not included.

Uses:

- a given condition element is never matched,
- the intersection of two or more condition elements, each of which is matched, fails to be satisfied, or
- a negated condition element is matched.

The PBREAK Command

- `(pbreak <rules>)` -- toggles break/nobreak status of rules
- `(pbreak)` -- says which rules are broken
- breaks after rule fires

The BACK Command

- `(back <n>)` undoes the effects of up to 32 rule firings, provided there are no external references (user-defined functions) in any RHS

The MAKE and REMOVE Commands

- `(remove *)` deletes everything from working memory.
- `(remove <args>)` deletes working memory elements with time tags in <args>

The EXCISE Command

`(excise <rules>)` -- prevents rules from firing (still in network), reload to recall, but won't be current on wm.