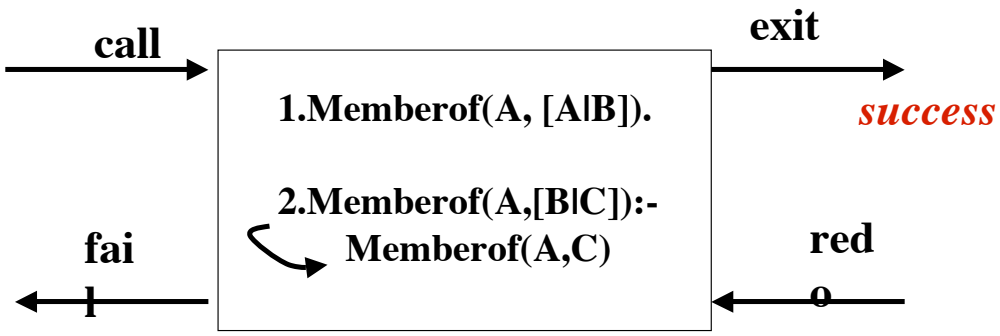
Tracing in Sicstus Prolog

- **Procedural interpretation of execution**
- **Box model of Prolog predicate rule**
- **How to follow a Prolog trace?**

Prolog Predicate - Box Model



Example

```
| ?- [mem]. %reads *.pl file into interpreter
{consulting /grad/users/ryder/prolog/programs/mem.pl...}
{Warning: [B] - singleton variables in memberof/2 in lines 1-
   2}
{Warning: [B] - singleton variables in memberof/2 in lines 2-
   3}
{/grad/users/ryder/prolog/programs/mem.pl consulted, 10 msec
   808 bytes}
yes
| ?- listing. %lists current predicate rules; can think
   %of them as procedure for this predicate
file_search_path(library, A) :- library_directory(A).
mem(A, [A|_]).                  %RULE 1
mem(A, [_|B]) :- mem(A, B).     %RULE 2
```


Example

```
| ?- trace.
{The debugger will first creep -- showing
   everything (trace)}
yes
{trace}
| ?- mem(X,[a,b,c]).
   1  1 Call: mem(_246,[a,b,c]) ?%succeeds, RULE 1
? 1  1 Exit: mem(a,[a,b,c]) ? %1 in listing
   %indicates initial copy of predicate rules being
   %used
X = a ? ; %user asks for more answers
```

```
mem(A, [A|_]).                  %RULE 1
mem(A, [_|B]) :- mem(A, B).     %RULE 2
```

Example

```
1    1 Redo: mem(a,[a,b,c]) ? %Tries RULE 2,
     %generates first recursive copy of rules;
    2  2 Call: mem(_246,[b,c]) ?
     %successfully matches RULE 1 in recursive copy
?   2  2 Exit: mem(b,[b,c]) ?
?   1  1 Exit: mem(b,[a,b,c]) ?

X = b ? ; %user asks for another answer
```

```
mem(A, [A|_]).                %RULE 1
mem(A, [_|B]) :- mem(A, B).   %RULE 2
```


Example

```
 1   1 Redo: mem(b,[a,b,c]) ? %Tries RULE 2 in
  1st              %recursive copy; results in 2nd
                   %recursive copy
     2       2 Redo: mem(b,[b,c]) ?
     3       3 Call: mem(_246,[c]) ?%matches RULE 1
              %in 2nd recursive copy
?    3       3 Exit: mem(c,[c]) ?
     2       2 Exit: mem(c,[b,c]) ?
     1       1 Exit: mem(c,[a,b,c]) ?

X = c ? ; %user asks for another answer
```

```
mem(A, [A|_]).                %RULE 1
mem(A, [_|B]) :- mem(A, B).   %RULE 2
```

Example

```
  1    1 Redo: mem(c,[a,b,c]) ?
  2    2 Redo: mem(c,[b,c]) ?
  3    3 Redo: mem(c,[c]) ?%matches RULE 2 in 2nd
             %recursive copy; results in 3rd
             %recursive copy
  4    4 Call: mem(_246,[]) ?%fails to match RULE 1
             %or RULE 2 in 3rd recursive copy, fails
  4    4 Fail: mem(_246,[]) ?
  3    3 Fail: mem(_246,[c]) ?
  2    2 Fail: mem(_246,[b,c]) ?
  1    1 Fail: mem(_246,[a,b,c]) ?
no
{trace}
```

```
mem(A, [A|_]).                %RULE 1
mem(A, [_|B]) :- mem(A, B).   %RULE 2
```