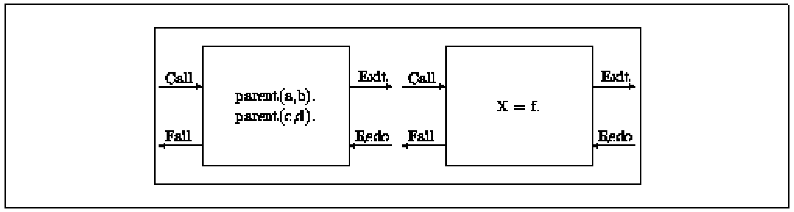
The Box model of computation

CALL, EXIT, REDO, FAIL

Example: `parent(a,b).  parent(c,d).`

1. `?- parent(X,d).`

2. `?- parent(X,Y), X=f.`



Extralogical features I: Cut

Example 1:

```
p :- a,b.
p :- c.
a. c.
?- p.
```

Example 2:

```
p :- a,!,b.
p :- c.
a. c.
?- p.
```

in logic = $p \leftarrow (a \wedge b) \vee (\neg a \wedge c)$

Extralogical features II: fail

```
all_solutions(A,B) :-
   call(A),
   write(X), nl,
   fail.
all_solutions(_,_).

p(a). p(b). p(c).

?- all_solutions(p(X),X).
```


Extralogical features III: negation

Closed World Assumption

```
tutor(m, mojmir).
?- tutor(w,X).

tutor(f,ivana).
?- tutor(w,X).

?- tutor(X,Y), not(X=m).

not(X) :- call(X),!,fail.
not(X).
```

Metainterpreters I

The simplest metainterpreter

```
prove(true).
prove((A,B))     :- prove(A), prove(B).
prove(A)         :- A.
prove(A)         :- clause(A,B), prove(B).
```


The metainterpreter that can interpret itself

```
prove(true).
prove((A,B))     :- prove(A), prove(B).
prove(A)         :- built_in(A),A.
prove(A)         :- clause(A,B), prove(B).

built_in(clause(_,_)).
```


Metainterpreter saving the trace

```
solve(',' (A,B),In,Out) :-
      solve(A,In,Out1),solve(B,Out1,Out).
solve(A,In,Out) :-
      clause(A,B),
      append(In,[A],Out1),
      solve(B,Out1,Out).
solve(true,In,In).
solve(A,In,Out) :-
      A,append(In,[A],Out).

r:-     solve(fib(3,F),[],L),write(L),
        L=[H|B],assert(':-'(H,B)).
```


Depth-Bounded Interpreter

```
solve(P,X) :- solve(P,25,X),(X \== true, !; true).

solve(true,D,true) :- !.
solve(A,0,(overflow,[])) :- !.
solve((A,B),D,S) :- !,
   solve(A,D,Sa),
   ifthenelse(Sa=true,(solve(B,D,Sb),S=Sb),S=Sa).
solve(A,D,Sa) :-
   ifthenelse(system(A),(A, Sa = true),
      (  D1 is D - 1,
        clause(A,B), solve(B,D1,Sb),
        (Sb=true -> Sa = true;
        Sb=(overflow,S) -> Sa=(overflow,[(A:-B)|S])
```

Handling cuts

```
prove1(Goal) :- rule(Goal, Before, Cut, After),
prove1_body(Before),
ifthen(Cut=:=1,!),
prove1_body(After).
prove1_body([]).
prove1_body([Goal|Goals]) :- prove1(Goal),
prove1_body(Goals).

rule(a(1), [b,c],0,[]).  % a(1) :- b,c.
rule(a(2), [],0,[]). % a(2).
rule(b,[],0,[]). % b.
rule(c,[d],1,[f]). % c    :- d,!,f.
rule(d,[],0,[]). % d.

a(1) :- b,c. a(2). b. c:-d,!,f. d.
```