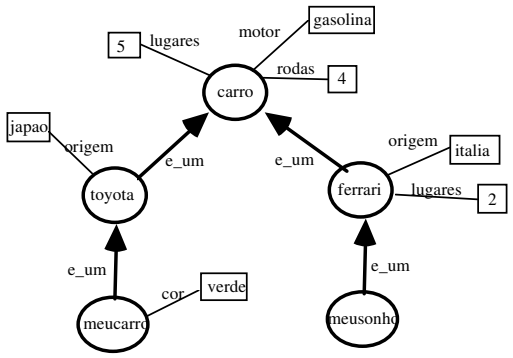
CONCEITOS BÁSICOS:
OBJECTOS. CLASSES. FRAMES.

Taxionomias. Herança.

Atributos. Valores por omissão.

Exemplo:




```
atr(carro,motor,gasolina).
atr(carro,lugares,5).
atr(carro,rodas,4).
atr(toyota,origem,japao).
atr(ferrari,origem,italia).
atr(ferrari,lugares,2).
atr(meucarro,cor,verde).

e_um(toyota,carro).
e_um(meucarro,toyta).
e_um(ferrari,carro).
e_um(meusonho,ferrari).
```

```
atributo(X,A,V) :- atr(X,A,V).
atributo(X,A,V) :- e_um(X,Y), atributo(Y,A,V).
```

Testar herança:

?-atributo(carro,lugares,L).
?-atributo(meucarro,origem,O).
?-atributo(meucarro,lugares,L).

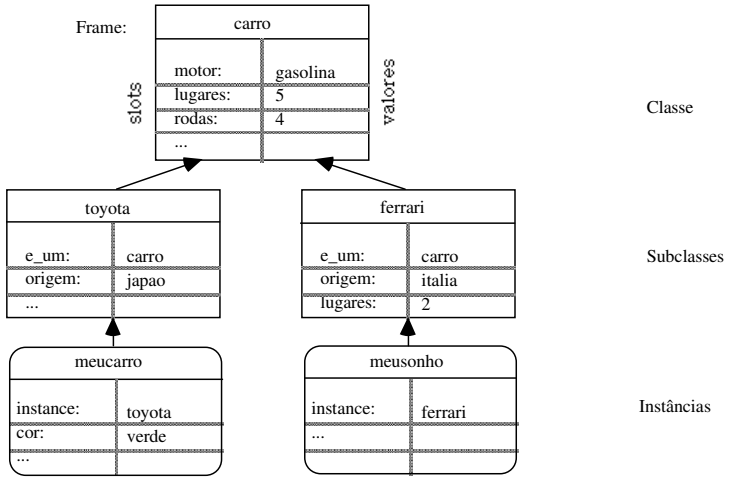
Redefinição local:

atr(meucarro,lugares,4).

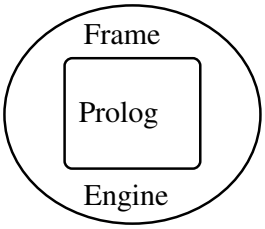
Uso do cut para evitar soluções múltiplas:

```
atributo(X,A,V) :- atr(X,A,V), !.
atributo(X,A,V) :- e_um(X,Y), atributo(Y,A,V).
```

Frames.
Classes. Instâncias.
Slots. Relações. Herança.



GOLOG: Um mini Frame Engine em Prolog



-Criação e manipulação de frames e seus slots e valores.
-Definição de relações e mecanismos de herança.
-Definição de métodos e programação reactiva.

Operações:

A. Sobre frames
- •new_frame(F)
- •frame_exists(F)
- •show_frame(F)
- •delete_frame(F)

B. Sobre slots
- •new_slot(F,S)
- •new_slot(F,S,V)
- •delete_slot(F,S)
- •frame_local_slots(F,LS)
- •get_all_slots(F,LS)

C. Sobre valores
- •new_value(F,S,V)
- •new_values(F,S,LV)
- •add_value(F,S,V)
- •add_values(F,S,LV)
- •get_value(F,S,V)
- •get_values(F,S,LV)
- •delete_value(F,S,V)
- •delete_values(F,S)

D. Sobre relações
- •new_relation(Relation,Transitivity, Restriction, Inverse)

Transitivity: transitive, intransitive
Restriction: all
none
inclusion(LS)
exclusion(LS)
Inverse: nome de relação inversa ou nil

•delete_relation(Relation)
•frame_actual_relations(F,LR)

Exemplo:

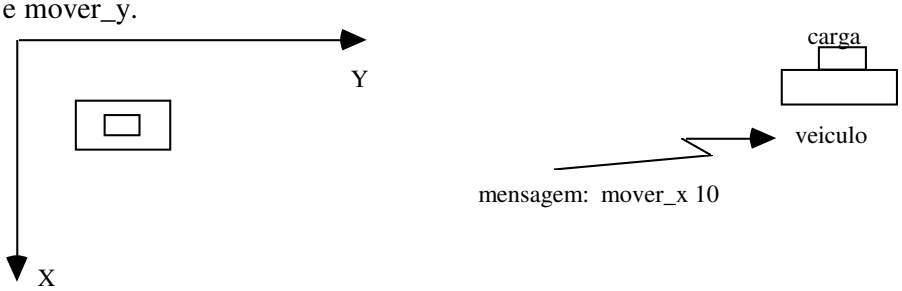
```
?-new_frame(carro).
?-new_slot(carro,motor,gasolina).
?-new_slot(carro,lugares,5).
?-new_slot(carro,rodas,4).

?-new_relation(e_um,  transitive, all, nil).

?-new_frame(toyota).
?-new_slot(toyota, e_um, carro).
?-new_slot(toyota, origem, japao).
?-new_frame(ferrari).
?-new_slot(ferrari, e_um, carro).
?-new_slot(ferrari, origem, italia).
?-new_value(ferrari, lugares,2).

?-new_relation(instancia, intransitive, all, nil).

?-new_frame(meucarro).
?-new_slot(meucarro, instancia, toyota).
?-new_slot(meucarro,cor,verde).
?-new_frame(meusonho).
?-new_slot(meusonho,instancia,ferrari).
```

Exercitar operações: get_value / new_value / etc.

E. Métodos

•new_slot(F,S,Metodo)
•call_method(F,S,LPar)
•call_method_0(F,S)
•call_method_1(F,S,P)
•call_method_2(F,S,P1,P2)
•call_method_3(F,S,P1,P2,P3)

Exemplo: Seja um veículo que se move num espaço 2D, obedecendo aos comandos mover_x e mover_y.




```
?-new_frame(veiculo).
?-new_slot(veiculo,posicao, [15, 8]).
        ...
?-new_slot(veiculo, mover_x,  desloca_x).
```

?-new_slot(veiculo, mover_y, desloca_y).
...

Os métodos seriam definidos em Prolog.

Exemplo:

```
desloca_x(F,Dx) :- get_values(F,posicao,[X,Y]), Nx is X + Dx,
                        new_values(F,posicao,[Nx,Y]),
                   write('Desloquei-me '), write(Dx),
                   write(' passos ao longo de X'), nl.
desloca_y(F,Dy) :- ....
```

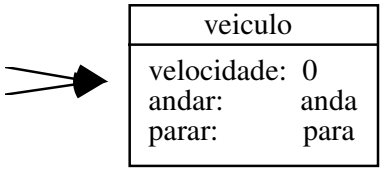
```
?-new_relation(em_cima_de,transitive,inclusion([posicao]), em_baixo_de).
?-new_frame(carga).
?-new_slot(carga,em_cima_de, veiculo).
```

A relação "em_cima_de" permite que o objecto "carga" herde o atributo "posição" do "veículo". Teste:

```
?-get_values(carga,posicao,P).
?-call_method_1(veiculo,mover_x, 10).     /* envio de mensagem */
?-get_values(veiculo,posicao,P).
?-get_values(carga,posicao,P).
```

Exemplo: Seja uma representação dum veículo com as operações (métodos): andar, parar.

```
?-new_frame(veiculo).
?-new_slot(veiculo, velocidade, 0).
?-new_slot(veiculo, andar, anda).
?-new_slot(veiculo, parar, para).
```



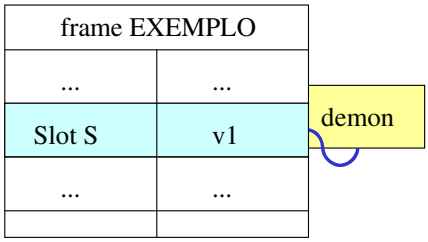
onde:

```
para(F):- new_value(F, velocidade, 0), write('Parei ! '), nl.
anda(F, V):- new_value(F, velocidade, V),
             write('Estou a andar a '), write(V), write(' Km'), nl.
```

```
?-call_method_1(veiculo, andar, 120).
?-get_value(veiculo,velocidade,V).
?-call_method_0(veiculo, parar).
?-get_value(veiculo,velocidade,V).
```

Programação reactiva.

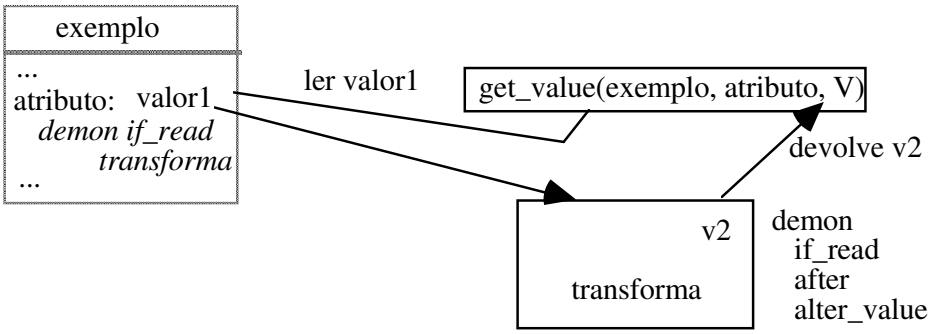
F. Attached Predicates ou *demons*

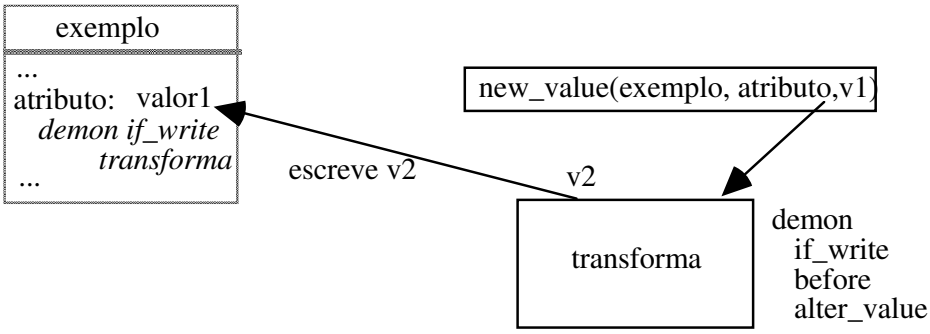



```
•new_demon(F,S,Demfunc, Demtype, When, Effect)
                                  before      alter-value
                                  after       side_effect
                        if_read
                        if_write
                        if_executed
                        if_deleted
```

•add_demon(F,S,Demfunc, Demtype, When, Effect)

•remove_all_demons(F,S)





Formato da regra associada ao slot:
Ex:

```
transforma(Frame, Slot, Valor_recebido, Valor_gerado) :- ....
```

Exemplo:
Em relação ao exemplo do veículo, usar programação reactiva para impedir velocidades superiores a um dado limite.

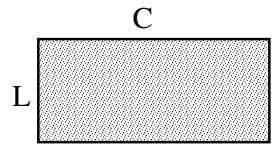
```
?-new_slot(veiculo, limite, 175).
?-new_demon(veiculo, velocidade, policia, if_write, before, side_effect).
```

onde:

```
policia(F, S, V, V) :- get_value(F,limite,Vl), V <= Vl, !.
policia(F, S, V, _)  :- write('Excesso de velocidade '), !, fail.
```

Exercício proposto: Construir um "demónio" que limite a velocidade máxima a 120 Km/h
(sem abortar o programa).

Atributos derivados - calculados dinamicamente a partir de outros.
Exemplo: Modelar um rectângulo



Area = C x L
Perimetro = 2 x (C + L)

```
?-new_frame(rectangulo).
?-new_slot(rectangulo, largura).
?-new_slot(rectangulo, comprimento).
?-new_slot(rectangulo,area).
?-new_frame(rectangulo, perimetro).
?-new_frame(rect1).
?-new_slot(rect1, e_um, rectangulo).
?-new_value(rect1, largura, 25).
?-new_value(rect1, comprimento, 40).
```

```
?-new_demon(rectangulo, area, calc_area, if_read, after, alter_value).
?-new_demon(rectangulo, perimetro, calc_perim, if_read, after, alter_value).
```

onde:

```
calc_area(F,S, _, A) :- get_value(F, largura, L),
                        get_value(F, comprimento, C), A is C * L.
calc_perim(F,S, _, P):- get_value(F, largura, L),
                        get_value(F, comprimento, C), P is 2 * (C + L).
```

Testar:

```
?-show_frame(rectangulo).
?-get_value(rect1, area, A).
?-get_value(rect1, perimetro, B).
```

Outro exemplo:
Supondo que os atributos "largura" e "comprimento" estão em mm, definir outros dois atributos que permitam obter esses valores em cm.

```
?-new_slot(rectangulo, cm_larg).
?-new_slot(rectangulo, cm_compr).
?-new_demon(rectangulo, cm_larg, converte_l, if_read, after, alter_value).
?-new_demon(rectangulo, cm_compr, converte_c, if_read, after, alter_value).
```

onde:

```
converte_l(F, S, _, Lcm) :- get_value(F, largura, L), Lcm is L / 10 .
converte_c(F,S, _, Ccm) :- get_value(F, comprimento, C), Ccm is C / 10.
```

Testar:

```
?-get_value(rect1, cm_larg, L).
?-get_value(rect1, cm_compr, C).
```

Exercício: Modificar o modelo de veículo para incluir os atributos "velocidade", "aceleração" e "tempo", e os métodos "acelerar", "travar" e "parar". Cada vez que se aumentar o tempo, a velocidade deve ser actualizada.

```
frame veiculo {
       velocidade:  0
       aceleracao:  0
       tempo:       0
           demon if_write act_veloc
       acelerar:    acelera
       travar:      trava
       parar:       para
       }
```

onde:

```
acelera(F, Da, T) :- get_value(F, tempo, Ta), Ta <= T,
                     get_value(F, aceleracao, A), Na is A + Da,
```

```
                    new_value(F, aceleracao, Na), new_value(F, tempo, T).

trava(F, Da, T) :- get_value(F, tempo, Ta), Ta <= T,
                   get_value(F, aceleracao, A), A >Da, Na is A - Da,
                    new_value(F, aceleracao, Na).

para(F) :- new_value(F, velocidade, 0), new_value(F, aceleracao, 0).
```

Demónio que reage à actualização do tempo:

```
?-new_demon(F, tempo, act_veloc, if_write, before, side_effect).

act_veloc(F, _, T, _) :- get_value(F, tempo, Ta), T < Ta, !, fail.
act_veloc(F, _, T, _) :- get_value(F, velocidade, Va),
                         get_value(F, tempo, Ta),
                         get_value(F, aceleracao, A), Td is T - Ta,
                         Vn is Va + A * Td,
                          new_value(F, velocidade, Vn).
```

Testar:

```
?-call_method_2(veiculo, acelerar, 20, 5).
?-get_value(veiculo, aceleracao, A).
?-get_value(veiculo, tempo, T).
?-get_value(veiculo, velocidade, V).
```

Exercício proposto:
Junte um novo atributo a veículo - espaço - que represente o espaço andado pelo veículo até ao tempo corrente. Use demónios para calcular esse valor. Se necessário considere outros slots auxiliares.