Exemplo (Esquema de uma relação)

```
father(Father, Child).
mother(Mother, Child).
male(Person).
female(Person).
```

Exemplo (Explicitando informação implícita)

```
procreated(Man, Woman) <-
 father(Man, Child), mother(Woman,Child).
```

Predicado $\neq$

Exemplo (Relação sibling (irmão))

```
sibling(Sib1,Sib2) <-
 parent(Parent,Sib1), parent(Parent,Sib2).
```

Qual a resposta à query sibling(X,X)?

Exemplo (Relação sibling)

```
sibling(Sib1,Sib2) <-
 parent(Parent,Sib1), parent(Parent,Sib2),
 Sib1 ≠ Sib2.
```

Definição (Predicado $\neq$)

*Term*1 $\neq$ *Term*2 é verdade se os termos *Term*1 e *Term*2 são diferentes.
Nota: neste altura consideramos somente constantes como termos.

Predicado com o mesmo nome mas diferente aridade

Exemplo (Predicado `mother/1`)

```
mother(Woman) <- mother(Woman, Child).
```

Os predicados `mother/2` e `mother/1` podem coexistir.

A estruturação de dados é muito importante na programação.

Exemplo

Considere que pretende representar que o docente Vitor Nogueira lecciona Programação Declarativa (PD) às 3as entre as 14 e as 16 na sala 131 do CLAV:

1. `aula(pd, vitor, nogueira, tercas, 14, 16, clav, 131)`
2. `aula(pd, docente(vitor, nogueira), horario(tercas, 14, 16),local(clav, 131))`

Exemplo

Para cada uma das abordagens vamos definir uma regra para o predicado `professor(Docente, Disciplina)`:

1. `professor(Nome, Apelido, Disciplina) <- aula(Disciplina, Nome,Apelido,Dia,Hora_I, Hora_F,Edif,Sala).`
2. `professor(Docente, Disciplina) <- aula(Disciplina, Docente, Horario, Local).`

Estuturação e Abstracção

+ Modularidade. Por exemplo, podemos alterar a representação do tempo sem afectar a regra para `professor/2`.
+ Representação compacta.
- ?

Exemplo (great* grandparent)

```
grandparent(Ancestor,Descendant) :-
  parent(Ancestor,Person),
  parent(Person,Descendant).

greatgrandparent(Ancestor,Descendant) :-
  parent(Ancestor,Person),
  grandparent(Person,Descendant).

greatgreatgrandparent(Ancestor,Descendant) :-
  parent(Ancestor,Person),
  greatgrandparent(Person,Descendant).
```

Exemplo (Ancestor)

```
/*
ancestor(Ancestor,Descendant) <-
  Ancestor is an ancestor of Descendant
*/

ancestor(Ancestor,Descendant) :-
  parent(Ancestor,Descendant).

ancestor(Ancestor,Descendant) :-
  parent(Ancestor,Person),
  ancestor(Person,Descendant).
```

A relação `ancestor` é o fecho transitivo da relação `parent`.

Exemplo (Mundo dos blocos)

Considere que uma pilha de blocos é descrita por uma colecção de factos `sobre(Bloco1,Bloco2)` que indica que o `Bloco1` está em sobre o `Bloco2`.
Defina o predicado `acima(Bloco1, Bloco2)` que é verdade se o `Bloco1` está acima do `Bloco2`.