Ficha Prática 3

3.1 Objectivos

1. Praticar a escrita de predicados.

3.2 Conceitos

3.2.1 Definição de Operadores

A declaração:

```
:-op(700, xfy, impl)
```

no início de um ficheiro Prolog, define o operador `impl` (terceiro parâmetro de `op`).

O primeiro parâmetro define a precedência do operador. A precedência pode ser um número entre 1 e 1200. Uma precedência de 0 remove o operador. Quanto menor o número, maior a precedência do operador. Neste caso, o operador `impl` tem precedência 700.

O segundo parâmetro define o tipo do operador. O tipo pode ter um dos seguintes valores: `xf`, `yf`, `xfx`, `xfy`, `yfx`, `yfy`, `fy` ou `fx`. O `f` indica a posição do functor (infixa ou prefixa). O `x` e o `y` indicam a posição dos argumentos. O `x` lê-se: 'nesta posição deve ocorrer um termo com precedência estritamente inferior à do functor'. O `y` lê-se: 'nesta posição deve ocorrer um termo com precedência inferior ou igual à do functor'. A precedência de um termo é 0, excepto se o seu principal functor for um operador, caso em que a precedência do termo é a precedência do operador. Um termo entre parêntesis tem precedência 0. A utilização do x e do y permite controlar a forma como as precedências são aplicadas a uma expressão. Neste caso o operador `impl` é um operador infixo.

O terceiro parâmetro, tal como já foi mencionado, é o nome do operador. Este parâmetro pode também ser uma lista de nome[4]. Neste caso, todos os operadores da lista passam a ter a mesma precedência e tipo.

Com a declaração acima é agora possível escrever termos da forma:

termo1 `impl` *termo2*

Esta possibilidade ser-lhe-á útil na resolução do exercício 3.3.3.

Alguns operadores do SWI Prolog são apresentados na tabela 3.1.

3.3 Exercícios

3.3.1 Contactos

Considere agora uma nova Base de Conhecimento contendo os predicados `telefone/2`, `visita/2` e `emCasa/1`:

[4]Listas serão tema de uma Ficha Prática futura.

Precedência	Tipo	Nome do operador
1200	xfx	:-
1200	fx	:-, ?-
1100	xfy	;
1000	xfy	,
700	xfy	<, >, =<, >=
500	yfx	+, -
500	fx	+, -
400	yfx	*, /, //

Tabela 3.1: Alguns operadores SWI-Prolog

```
% telefone(P, T) :-
%   o n. de telefone da casa da pessoa P é T.
telefone(ana, 123).
telefone(ze, 234).
telefone(rui, 345).
telefone(pedro, 456).
telefone(marta, 567).
telefone(olga, 678).


% visita(X, Y) :-
%   a pessoa X está de visita à pessoa Y.
visita(olga, ana).
visita(marta, ze).
visita(rui, olga).
visita(pedro, olga).

% emCasa(X) :- X está em casa.
emCasa(ze).
emCasa(ana).
```

1. Escreva uma *query* que determine se `ana` está a visitar alguém.
2. Escreva uma *query* que determine se `ana` tem visitas.
3. Sabendo que uma pessoa P está acompanhada se tem visitas, acrescente à Base de Conhecimento o predicado `acompanhada/1`.
4. Acrescente à base de conhecimento o predicado `inconsistente/0` que determina se, na base de conhecimento, existe alguém que está simultaneamente em casa e a visitar alguém.
5. Supondo que quando alguém sai para fazer uma visita leva consigo todos os que o estão a visitar[5], acrescente à Base de Conhecimento o predicado `em_casa_de/2` que lhe permite determinar se uma pessoa está em casa de outra.
6. Acrescente à Base de Conhecimento o predicado `contacto/2` que lhe permite determinar qual o número de telefone em que cada pessoa está contactável[6].
7. Sabendo que três ou mais pessoas numa casa correspondem a uma festa, escreva um predicado `a_dar_festa/1` que determina se uma pessoa está a dar uma festa.

[5]Exemplos: O Rui em casa da Ana: o Rui está a visitar a Olga, como a Olga está a visitar a Ana, então o Rui foi com a Olga para a casa da Ana.

[6]Assuma um mundo em que ainda não existem telemóveis!

3.3.2 Mapa de Acessibilidades

1. Considere o mapa da figura 3.5, que indica os tipos de ligações possíveis entre diversas cidades.

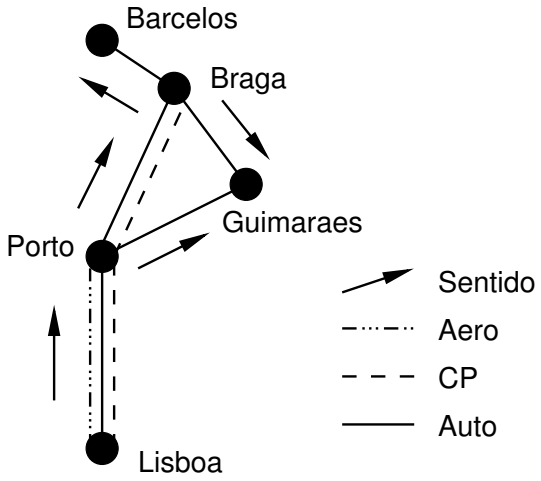

Figura 3.5: Mapa de acessibilidades

   (a) Escreva uma Base de Conhecimento que expresse a informação contida no mapa (utilize o predicado `ligacaodirecta/3` em que `ligacaodirecta(O,D,T)` se existe ligação pelo meio de transporte `T` entre `O` e `D`).

   (b) Escreva os seguintes predicados:

      i. `ha_ligacao/2` — `ha_ligacao(A,B)` sucede se existe ligação entre as cidades A e B.

      ii. `ha_ligacao/3` — `ha_ligacao(A,B,T)` sucede se é possível viajar entre as cidades A e B usando apenas o meio de transporte T.

2. Considere o predicado:

```
ha_ligacao_aux(A,B) :- ha_ligacao(A,B,_).
```

   Identifique e discuta as diferenças existentes entre os predicados `ha_ligacao/2` e `ha_ligacao_aux/2`

3. Cada vez mais os meios de transporte modernos fornecem formas de chegar cada vez mais depressa a zonas cada vez mais congestionadas. Considere que na Base de Conhecimento acima é acrescentado o predicado `nao_engarrafado/2`, indicando que numa dada cidade um dado tipo de meio de transporte não se encontra engarrafado. Redefina os predicados definidos anteriormente de modo a apenas considerar ligações que passem por cidades onde os meios de transporte a utilizar não estão engarrafados (considere que o engarrafamento só afecta quem quer entrar na cidade).

3.3.3 Demonstrador de Teoremas

Escreva um demonstrador de teoremas para o cálculo proposicional.

O demonstrador deverá ser capaz de lidar com equivalências ($\leftrightarrow$), implicações ($\rightarrow$), disjunções ($\vee$), conjunções ($\wedge$) e negação ($\neg$). Utilizando `op/3` defina os seguintes operadores (atribuindo-lhes tipos e precedências apropriados):

- `equiv` — para a equivalência;
- `impl` — para a implicação;
- `ou` — para a disjunção;
- `e` — para a conjunção;
- `nao` — para a negação.

Defina o demonstrador através do predicado `prova/1`. Teste o demonstrador com os seguintes exemplos:

```
?- prova(falso impl verdade).

Yes
?- prova(verdade impl falso).

No
?- prova((falso equiv verdade) equiv falso).

Yes
?- prova(verdade impl X).

X = verdade

Yes
?- prova(falso impl X).

X = _G155
```