Lógica e Prolog

- Prolog: Alain Colmerauer e seus colaboradores por volta de 1970
- Primeira linguagem a possibilitar que o programador especifique sua tarefa em lógica, ao invés de em termos de programação convencional

Cálculo de Predicado

- Lógica: maneira de representar a forma dos argumentos, para verificar de maneira formal se eles são válidos ou não
- Podemos expressar proposições, a relação entre proposições e como podemos inferir algumas proposições a partir de outras
- Cálculo de predicados: forma de lógica
- Objetos são representados por termos

Termos

- Constante: um único indivíduo ou conceito, nomes próprios).
  Exemplo: grego, paz, ágata, França, vinho, estados_unidos, etc.
- Váriável: diferentes indívíduos em tempos diferentes.
  Exemplo: X, Homem, Grego.
- Termo composto: função, com um conjunto ordenado de termos como seus argumentos.
  Exemplo: esposa(pedro), distancia(point1,X), humana(maria), gosta(Homem,vinho).

Conectores

- Conectores : "not", "and", "or", "implica" e "é equivalente à".
  Negação: $\sim \alpha$ "não α"
  Conjunção: $\alpha \,\&\, \beta$ "α e β"
  Disjunção: $\alpha \,\#\, \beta$ "α ou β"
  Implicatura: $\alpha \rightarrow \beta$ "α implica β"
  Equivalência: $\alpha \leftrightarrow \beta$ "α é equivalente a β"

- Exemplo:
  homem(fred) # mulher(fred)
  homem(fred) $\rightarrow$ humano(fred)

Quantificadores

- Quantificadores: permitem falar sobre conjuntos de indivíduos e o que é verdade sobre eles.
- Dois tipos: universal e existencial.
- Se v representa qualquer variável e P qualquer proposição, então:
  paratodo(v,P) "P é verdadeiro para todo v"
  existe(v,P) "existe algum v tal que P é verdadeiro"
- Exemplos:
  paratodo(X, (homem(X) $\rightarrow$ humano(X)))
  existe(Z, (pai(joao,Z) & feminino(Z)))
  paratodo(X, (animal(X) $\leftrightarrow$ existe(Y, mae(X,Y)))).

Exercícios

1. A França exporta vinho para a Inglaterra.
2. A França é um país.
3. Paris é a capital da França.
4. A França é européia.
5. Paris é uma cidade européia.
6. A França é um país que faz fronteira com a Espanha.
7. Zambia exporta minerais.
8. Alguns pássaros migram.
9. Não existem rios na Antártida.
10. Homens habitam todos os continentes.

11. Todo menino caiu.
12. Todos os meninos caíram.
13. Cada menino caiu.
14. Um menino caiu.
15. Todos os meninos que viram Maria caíram.
16. Todo homem ama uma mulher.

Respostas

1. exporta(França,vinho,inglaterra).
2. pais(França).
3. capital(franca,paris).
4. europeia(franca).
5. europeia(paris) & cidade(paris).
6. pais(franca) & faz_fronteira(franca,espanha).
7. existe(X, (mineral(X) & exporta(zambia,X))).
8. existe(X, (passaro(X) & migra(X))).
9. $\sim$ existe (X, (rio(X) & em(X, antartida))).

10. existe(X, (continente(X) & habita(homen,X))).
11. paratodo(X, (menino(X) $\rightarrow$ caiu(X))).
12. paratodo(X, (menino(X) $\rightarrow$ caiu(X))).
13. paratodo(X, (menino(X) $\rightarrow$ caiu(X))).
14. paratodo (X, (homem (X) $\rightarrow$ existe (Y, (mulher (Y) & ama(X,Y))))))

Continuação do Projeto

A) Estender a gramática de cláusulas definidas do primeiro projeto para gerar as fórmulas lógicas correspondentes ao significado das seguintes sentenças:

- (a) Todo menino caiu.
- (b) Todos os meninos caíram.
- (c) Cada menino caiu.
- (d) Um menino caiu.
- (e) Todos os meninos que viram Maria caíram.

Para efeito de geração de fórmulas lógicas, as sentenças (a), (b) e (c) são equivalentes.

B) Nesta segunda etapa, o artigo definido “a” deve ser tratado como existencial, de maior escopo. Assim, para a sentença

(f) Todo homem que ama a mulher que caiu viu um menino.

deve ser gerada a seguinte fórmula lógica:

existe (Y, (mulher(Y) & caiu(Y)) & paratodo(X, (homem(X) & ama(X,Y) $\rightarrow$ existe(Z, (menino(Z) & viu(X,Z))))))

Opcional

- (Opcional) Gerar para a sentença (f), todas as combinações aceitáveis de escopo para os quantificadores.

Como gerar fórmulas lógicas em Prolog

?- op(1200,xfy,&).

s(P) → sn(X,P1,P), sv(X,P1).
sn(X,P1,P) → art(X,P2,P1,P), subst(X,P2).
art(X,P1,P2,existe(X,(P1 & P2))) → [um].
subst(X,menino(X)) → [menino].
sv(X,P) → iverbo(X,P).
iverbo(X,caiu(X)) → [caiu].

```
[trace]  ?- s(X,[um,menino,caiu],[]).
   Call: (7) s(_G321, [um, menino, caiu], []) ?
   Call: (8) sn(_L193, _L194, _G321, [um, menino, caiu],
          _L195) ?
   Call: (9) art(_L193, _L216, _L194, _G321,
          [um, menino, caiu], _L217) ?
   Exit: (9) art(_G397, _G400, _G401, existe(_G397,
          (_G400&_G401)),[um, menino, caiu],
          [menino, caiu]) ?
   Call: (9) subst(_G397, _G400, [menino, caiu], _L195) ?
   Exit: (9) subst(_G397, menino(_G397), [menino, caiu],
          [caiu]) ?
   Exit: (8) sn(_G397, _G401, existe(_G397,
             (menino(_G397)&_G401)),[um, menino, caiu],
             [caiu]) ?
```

```
   Call: (8) sv(_G397, _G401, [caiu], []) ?
   Call: (9) iverbo(_G397, _G401, [caiu], []) ?
   Exit: (9) iverbo(_G397, caiu(_G397), [caiu], []) ?
   Exit: (8) sv(_G397, caiu(_G397), [caiu], []) ?
   Exit: (7) s(existe(_G397, (menino(_G397)&caiu(_G397))),
             [um, menino, caiu], []) ?

X = existe(_G397, (menino(_G397)&caiu(_G397)))
```

Agora com verbos transitivos

?- op(1200,xfy,&).

s(P) → sn(X,P1,P), sv(X,P1).
sn(X,P1,P) → art(X,P2,P1,P), subst(X,P2).
art(X,P1,P2,existe(X,(P1 & P2))) → [um].
art(X,P1,P2,existe(X,(P1 & P2))) → [uma].
subst(X,menino(X)) → [menino].
subst(X,menina(X)) → [menina].
sv(X,P) → verbo(X,P).
sv(X,P) → verbot(X,Y,P1), sn(Y,P1,P)
verboi(X,caiu(X)) → [caiu].
verbot(X,Y,viu(X,Y)) → [viu].

Agora com quantificadores

todo(X, menino(X) $\rightarrow$ existe(Y,(menina(Y)&ama(X,Y)))))

Todo menino ama uma menina

```
?- op(1150,xfy,&).
?- op(1200,xfy,→).
s(P) → sn(X,P1,P), sv(X,P1).
sn(X,P1,P) → art(X,P2,P1,P), subst(X,P2).
art(X,P1,P2,existe(X,(P1 & P2))) → [uma].
art(X,P1,P2,paratodo(X,(P1 → P2))) → [todo].
subst(X,menino(X)) → [menino].
subst(X,menina(X)) → [menina].
sv(X,P) → verbo(X,P).
sv(X,P) → verbot(X,Y,P1), sn(Y,P1,P)
verboi(X,caiu(X)) → [caiu].
verbot(X,Y,viu(X,Y)) → [viu].
verbot(X,Y,ama(X,Y)) → [ama].
```