



ISEL
INSTITUTO SUPERIOR DE
ENGENHARIA DE LISBOA

Instituto Superior de Engenharia de Lisboa

Lógica e Computação

Relatório da Primeira Fase do Trabalho Prático

Aluno

43552 Samuel Costa

Docente: Prof. Walter Vieira

Novembro 2019

Índice

Introdução	3
1ª Parte – Cálculo de Proposições.....	4
1.....	4
2.....	5
3.....	5
4.....	7
2ª Parte – Cálculo de predicados	9
1.....	9
2.....	9
3.....	12
4.....	15
3ª Parte – Prolog (iniciação).....	17
1.....	17
2.....	20
3.....	21

Introdução

Este relatório, referente à primeira fase do trabalho prático, foi elaborado no âmbito da unidade curricular de Lógica e Computação e acompanha a realização de uma série de exercícios constituída por 3 partes, que dizem respeito aos 3 primeiros capítulos abordados no estudo da disciplina: cálculo de proposições, cálculo de predicados e introdução à programação em Prolog.

1ª Parte – Cálculo de Proposições

1.

$a \equiv$ chove

$b \equiv$ trago o chapéu de chuva

$c \equiv$ faz sol

$$F_1 = a \rightarrow b$$

$$F_2 = c \rightarrow \neg b$$

a) $G_0 = c \wedge a = \square \Leftrightarrow \neg c \vee \neg a = \blacksquare$

Por consulta à tabela de verdade (linhas a verde na coluna G_0), podemos concluir que

$$F_1 \wedge F_2 \rightarrow G_0$$

b) $G_1 = a \leftrightarrow \neg c = (\neg a \wedge c) \vee (a \wedge \neg c)$

Por consulta à tabela de verdade (linhas a vermelho na coluna G_1), não podemos concluir que $F_1 \wedge F_2 \rightarrow G_1$

a	b	c	$F_1 = a \rightarrow b$	$F_2 = c \rightarrow \neg b$	$G_0 = \neg c \vee \neg a$	$G_1 = (\neg a \wedge c) \vee (a \wedge \neg c)$
F	F	F	T	T	T	F
F	F	T	T	T	T	T
F	T	F	T	T	T	F
F	T	T	T	F	T	T
T	F	F	F	T	T	T
T	F	T	F	T	F	F
T	T	F	T	T	T	T
T	T	T	T	F	F	F

2.

$a \equiv$ passar

$b \equiv$ estudar

$c \equiv$ ganhar um automóvel

$$F1 : a \leftrightarrow b = (a \rightarrow b) \wedge (b \rightarrow a) = (\neg a \wedge \neg b) \vee (a \wedge b)$$

$$F2: a \rightarrow c$$

a	b	c	F1 = $a \leftrightarrow b$	F2 = $a \rightarrow c$	G0 = $b \rightarrow c$	G1 = $c \vee (\neg a \wedge \neg b)$	G2 = $\neg a \vee (a \wedge b \wedge c)$
F	F	F	T	T	T	T	T
F	F	T	T	T	T	T	T
F	T	F	F	T	F	F	T
F	T	T	F	T	T	T	F
T	F	F	F	F	T	F	T
T	F	T	F	T	T	T	F
T	T	F	T	F	F	F	T
T	T	T	T	T	T	T	T

Das premissas F1 e F2 podemos derivar G0, G1 e G2, por exemplo.

3.

$a \equiv$ chove

$b \equiv$ trago o chapéu de chuva

$c \equiv$ vou à praia

$d \equiv$ faz sol

$e \equiv$ fico satisfeito

$$F1 = a \rightarrow b$$

$$F2 = a \rightarrow \neg c$$

$$F3 = d \rightarrow e$$

$$F4 = e$$

$$G = c \rightarrow \neg a$$

É necessário transformar as premissas na forma clausal, para poder aplicar a dedução linear $F1 \wedge F2 \wedge F3 \wedge F4 \wedge \neg G = \square$, seguindo, para tal, o algoritmo, de onde obtemos o conjunto das cláusulas.

Passo 1 – Eliminar implicações e equivalências

$$F1 = a \vee \neg b$$

$$F2 = \neg a \vee \neg c$$

$$F3 = \neg d \vee e$$

$$F4 = e$$

Passo 2 – Reduzir os domínios das negações

...

Passo 3 – Aplicar a lei distributiva

...

Passo 4 – Devolver o conjunto das cláusulas

$$C = \{ a \vee \neg b, \neg a \vee \neg c, \neg d \vee e, e \}$$

$$G = c \rightarrow \neg a = \neg c \vee \neg a \Leftrightarrow \neg G = \neg(\neg c \vee \neg a) = c \wedge a$$

$$C01 = c$$

$$C02 = a$$

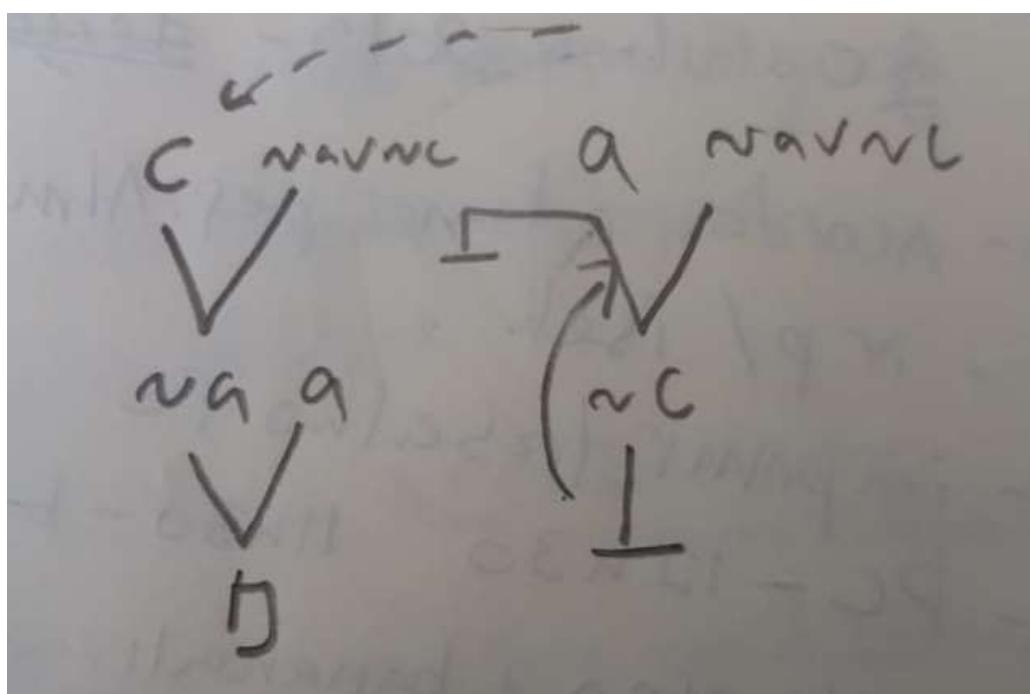
$$C1 = a \vee \neg b$$

$$C2 = \neg a \vee \neg c$$

$$C3 = \neg d \vee e$$

$$C4 = e$$

Obtidas as cláusulas, passemos a elaborar a árvore de dedução, utilizando o princípio de resolução para obter resolventes, a partir dos quais esperamos derivar a inconsistência.



4

Pretendemos demonstrar que um dos alunos tem conhecimento mais profundo que o outro. Para tal, uma primeira aproximação poderia consistir em contar o número de proposições que cada aluno produz. Mas essa abordagem não permite comparar os conhecimentos dos dois alunos, já que estes podem incidir sobre assuntos diferentes. Então, uma abordagem mais adequada seria descobrir se, dados dois conjuntos S_1 e S_2 , com as cláusulas que o aluno 1 e o aluno 2 conseguem produzir, S_1 é subconjunto de S_2 ou S_2 é subconjunto de S_1 , uma vez que, se o conjunto das cláusulas produzidas por um aluno é um subconjunto das cláusulas produzidas por outro, podemos dizer que o conhecimento do segundo aluno é mais profundo do que o do primeiro.

$a \equiv$ É aluno

$b \equiv$ É professor

$c \equiv$ Finge ser professor

Aluno 1

$$F_{1_1} = a \vee b$$

$$F_{1_2} = \neg a \vee c$$

De F_{1_1} e F_{1_2} podemos ainda derivar, pelo princípio da resolução $F_{1_3} = b \vee c$

Aluno 2

$$F_{2_1} = a \leftrightarrow \neg b = a \rightarrow \neg b \wedge \neg b \rightarrow a$$

$$F_{2_2} = c \leftrightarrow a = c \rightarrow a \wedge a \rightarrow c$$

Depois de colocar as premissas do aluno 2 na forma clausal, tentaremos derivar dessas cláusulas as premissas do aluno 1 que estão na forma clausal.

$$C_1 = a \rightarrow \neg b = \neg a \vee \neg b$$

$$C_2 = \neg b \rightarrow a = b \vee a$$

$$C_3 = c \rightarrow a = \neg c \vee a$$

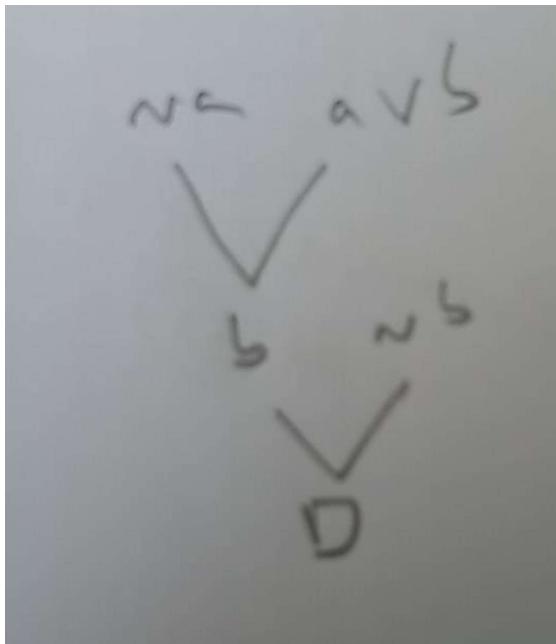
$$C_4 = a \rightarrow c = \neg a \vee c$$

$$G_{01} = a \vee b \Leftrightarrow \neg G_{01} = \neg(a \vee b) = \neg a \wedge \neg b$$

$$G_{02} = \neg a \vee c \Leftrightarrow \neg G_{02} = \neg(\neg a \vee c) = a \wedge \neg c$$

$$C_{01} = \neg a$$

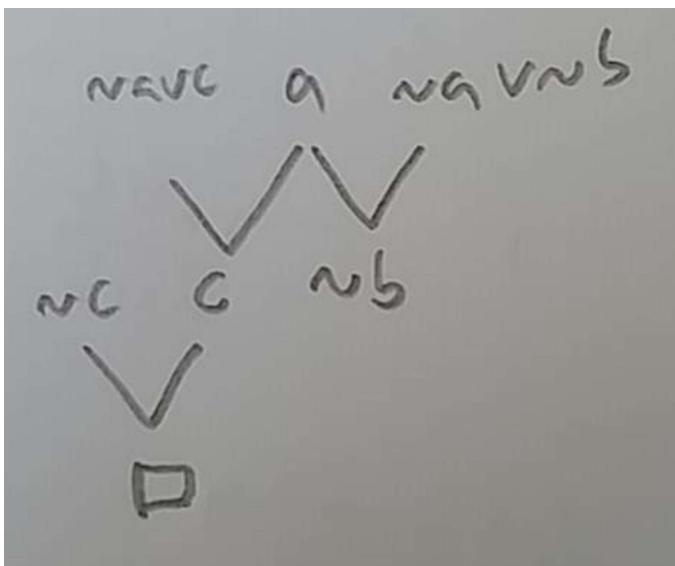
$$C_{02} = \neg b$$



Derivamos a inconsistência a partir de C01, portanto a cláusula F1_1 = $a \vee b$ deriva do conjunto das premissas do aluno 2

$$C03 = a$$

$$C04 = \neg c$$



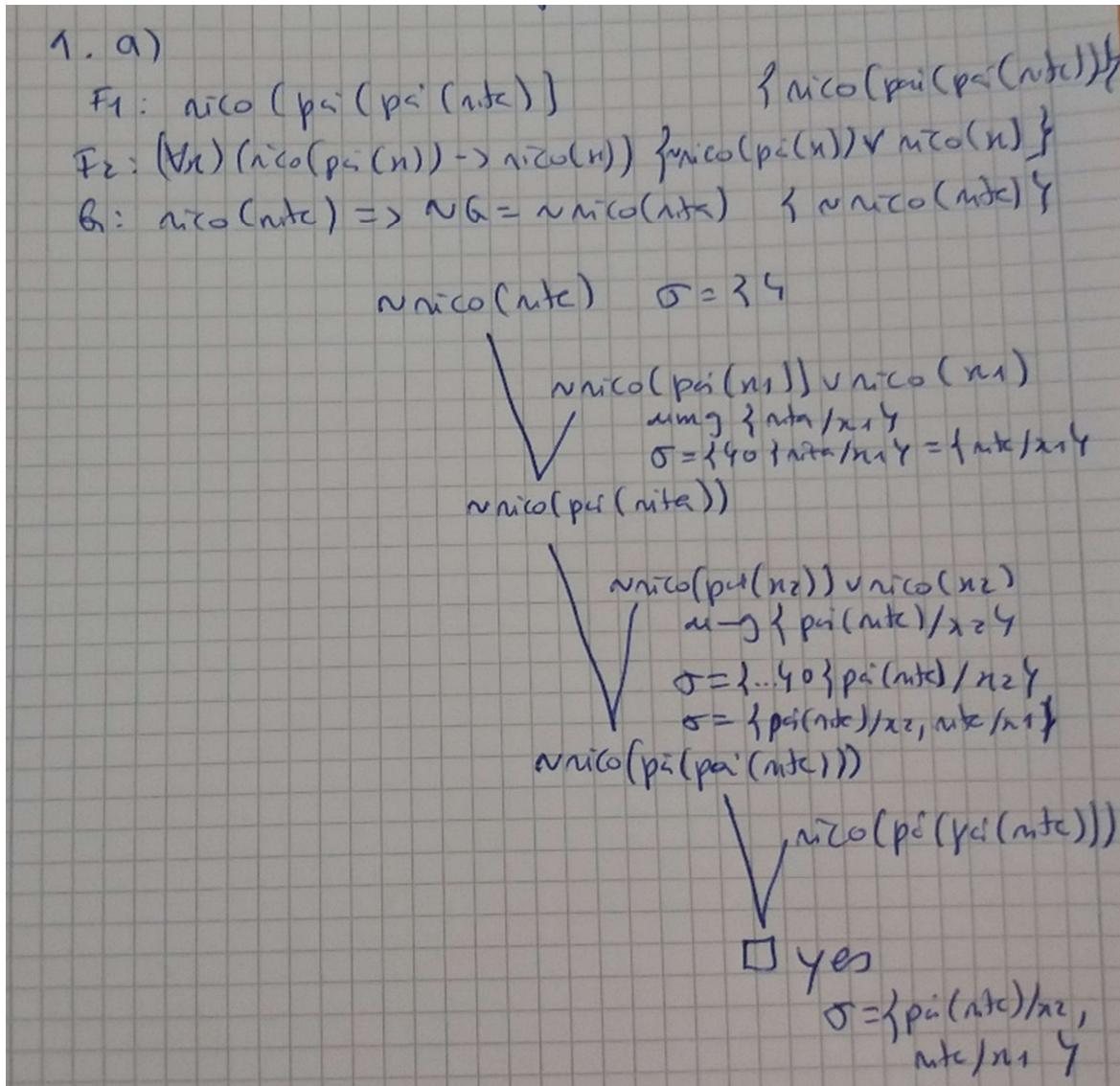
Derivamos a inconsistência a partir de C03, pelo que podemos dizer que F1_2 = $\neg a \vee c$ é consequência lógica das premissas do aluno 2.

Assim, podemos concluir que S1 (conjunto das premissas do aluno 1) é um subconjunto de S2 (conjunto das premissas do aluno 2), pelo que o conhecimento do aluno 2 é mais profundo do que o conhecimento do aluno 1.

2ª Parte – Cálculo de predicados

1

a)



Foi confirmada com o Logis a árvore de resolução apresentada.

b)

Foi eliminada a premissa F_1 , e repetido o exercício da alínea a). O sistema dedutivo entra num ramo infinito, pois encontra sucessivos resolventes, mas as premissas de que partiu e os resolventes obtidos não permitem derivar inconsistência. Trata-se de uma situação em que o universo de Herbrand é infinito. O sistema eventualmente responde “No”, tendo atingido a profundidade máxima configurada, que neste caso é 50, sem conseguir encontrar uma resposta ao objectivo.

2

a)

i.

```
#(x,route(c,f,x)).
```

Respostas:

```
x = l(c,l(d,l(e,l(f,nil))))
```

ii.

```
#(x,route(a,d,x)).
```

Respostas:

```
x = l(a,l(b,l(a,l(b,l(a,l(b,l(a,l(b,l(c,l(d,nil))))))))))
```

```
x = l(a,l(b,l(a,l(b,l(a,l(b,l(c,l(d,nil))))))))
```

```
x = l(a,l(b,l(a,l(b,l(c,l(d,nil))))))
```

```
x = l(a,l(b,l(c,l(d,nil))))
```

iii.

```
#(x,route(a,c,x)).
```

Respostas:

```
x = l(a,l(b,l(c,nil)))
```

```
x = l(a,l(b,l(a,l(b,l(c,nil)))))
```

```
x = l(a,l(b,(l(a,l(b,l(a,l(b,l(c,nil))))))))
```

```
x = l(a,l(b,l(a,l(b,l(a,l(b,l(a,l(b,l(c,nil))))))))))
```

```
x = l(a,l(b,l(a,l(b,l(a,l(b,l(a,l(b,l(c,nil))))))))))
```

b)

O conjunto dos resultados obtidos foi o mesmo quer na alínea a) quer na alínea b). No entanto, com a opção depth increment igual a 1, o Logls procurou todas as árvores de resolução começando com as de profundidade 0 até 20, que permitiam calcular uma resposta para os objectivos. Assim, em b), o número de respostas calculadas foi maior. Como todas as árvores de profundidade máxima igual a 20 que permitem calcular resposta para o objectivo incluem as de menor profundidade, na alínea a), onde a opção depth incrementa foi configurada como 20, foram calculadas as mesmas respostas que na alínea b), mas sem repetições de respostas.

c)

Foi introduzido no LogIs o objectivo #(x,#(y,route(a,x,y))). As 8 primeiras respostas calculadas foram:

```
y = l(a,l(g,nil))
x = g
*****
y = l(a,l(b,nil))
x = b
*****
y = l(a,l(g,nil))
x = g
*****
y = l(a,l(b,nil))
x = b
*****
y = l(a,l(g,nil))
x = g
*****
y = l(a,l(b,l(a,nil)))
x = a
*****
y = l(a,l(b,l(c,nil)))
x = c
*****
b)
```

Foi redefinida a opção depth incremente com o valor 20 e foram observados os seguintes resultados para o mesmo objetivo de a)

```
y = l(a,l(g,nil))
x = g
*****
y = l(a,l(b,nil))
x = b
*****
y = l(a,l(b,l(a,nil)))
x = a
*****
y = l(a,l(b,l(c,nil)))
x = c
*****
y = l(a,l(b,l(a,l(g,nil))))
x = g
*****
y = l(a,l(b,l(a,l(b,nil))))
x = b
*****
```

```
y = l(a,l(b,l(a,l(b,l(a,nil)))))
```

```
x = a
```

```
*****
```

```
y = l(a,l(b,l(a,l(b,l(c,nil)))))
```

```
x = c
```

```
*****
```

Conclusão: na alínea a), os oito resultados obtidos correspondiam na verdade apenas a quatro caminhos únicos ($a \rightarrow g, a \rightarrow b, a \rightarrow b \rightarrow a$ e $a \rightarrow b \rightarrow c$). A alínea b) permitiu acrescidamente calcular quatro resultados que não se encontravam no conjunto obtido em a) ($a \rightarrow b \rightarrow a \rightarrow g, a \rightarrow b \rightarrow a \rightarrow b, a \rightarrow b \rightarrow a \rightarrow b \rightarrow a, a \rightarrow b \rightarrow a \rightarrow b \rightarrow c$). Estes resultados adicionais não incluem novos nós, antes são caminhos compostos pelo percurso entre nós que já estavam incluídos nos percursos calculados em a). Conclui-se que o uso deste algoritmo para cálculo de rotas tem que tomar em consideração que a busca não apresenta preocupação em cálculo de caminhos óptimos, antes funciona numa busca em profundidade não orientada, limitando-se a calcular caminhos possíveis entre dois pontos do grafo, incluindo caminhos obtidos pelo percurso repetido entre 2 nós. Adicionalmente, pode ser acrescentado que, se tomarmos apenas as primeiras n respostas dadas pelo sistema dedutivo, a definição da propriedade depth increment com um número mais elevado (igual a maximum depth) permite cálculo de mais respostas, ainda que as respostas adicionais possam não incluir caminhos que passem por nós não incluídos nas respostas calculadas em a).

3

a)

Foi introduzida no LogIs o seguinte objectivo:

```
@(m, #(sublist, #(x, diml(sublist, &add(1,0)) * member(m, sublist) * diml(x, &add(2,1)) * member(sublist, x ))).
```

Foram calculadas três respostas, que se apresentam a seguir:

```
x = l(l(skolem44,nil),l(?104864,l(?104882,nil)))
```

```
sublist = l(skolem44,nil)
```

```
m = skolem44
```

```
*****
```

```
x = l(?104846,l(l(skolem44,nil),l(?104882,nil)))
```

```
sublist = l(skolem44,nil)
```

```
m = skolem44
```

```
*****
```

```
x = l(?104846,l(?104864,l(l(skolem44,nil),nil)))
```

```
sublist = l(skolem44,nil)
```

```
m = skolem44
```

b) Foi calculada a árvore de resolução para o objectivo: #(l,invert(l,l(a,l(b,nil)))).

$c_1 : \text{diml}(\text{nil}, \sigma)$

$c_2 : \text{ndiml}(r, dr) \vee \text{diml}(l(c, r), dr + 1)$

$c_3 : \text{member}(c, l(c, r))$

$c_4 : \text{number}(x, r) \vee \text{mem}(x, l(c, r))$

$c_5 : \text{addEnd}(x, \text{nil}, l(k, \text{nil}))$

$c_6 : \text{naddEnd}(x, r, k) \vee \text{addEnd}(x, l(c, r), l(c, k))$

$c_7 : \text{invert}(\text{nil}, \text{nil})$

$c_8 : \text{n invert}(r, rc) \vee \text{naddEnd}(c, rc, i) \vee \text{invert}(l(c, r), i)$

$G = (\exists x) \text{ invert}(l, l(a, l(b, \text{nil})))$

$c_0 : \text{n invert}(l, l(a, l(b, \text{nil})))$

$c_0 \sigma = \gamma$

$\sqrt{c_8}$
 $\sigma = \{ l(a, l(b, \text{nil})) / i, i, l(c, r) / l \}$
 $\text{n invert}(r, rc) \vee \text{naddEnd}(c, rc, l(a, l(b, \text{nil})))$

$\boxed{c_8}$
 $\sigma = \{ l(c_1, l(c_2, \text{nil})) / i, i, l(c_2, \text{nil}) / r_1, r_1, l(c_2, r_2) / r_2, r_2 / 2 \}$
 $\text{n invert}(r_2, rc_2) \vee$
 $\text{naddEnd}(c_2, rc_2, K_2) \vee$
 $\text{n addEnd}(c_1, K_1, l(a, l(b, \text{nil}))) \quad \text{falle}$

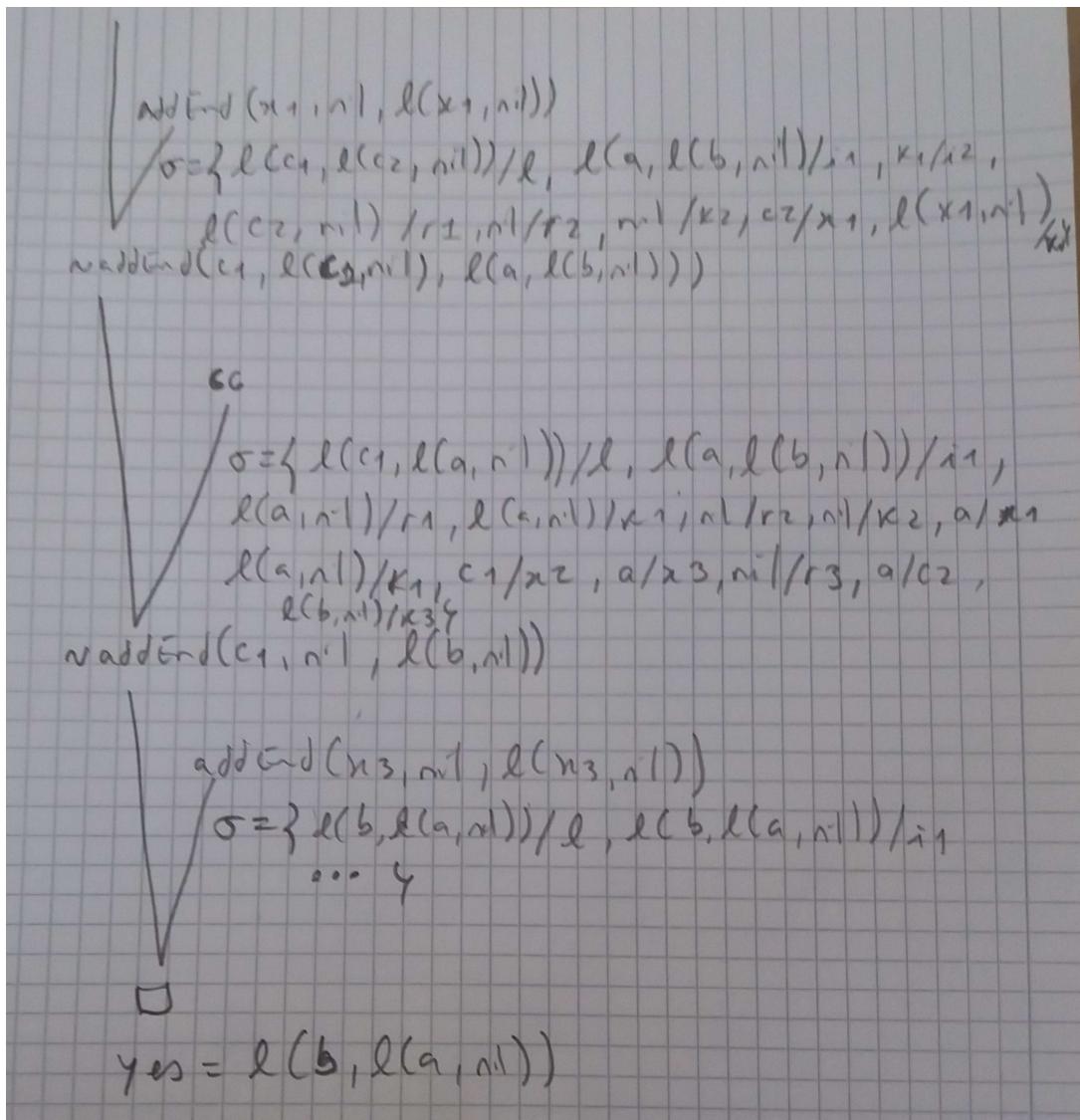
$\sqrt{\sigma = \{ \dots \}}$

$\sigma = \{ \dots \}$

$\text{naddEnd}(c_2, \text{nil}, K_2) \vee$

$\text{naddEnd}(c_1, K_1, l(a, l(b, \text{nil})))$

...



c) Foi introduzido no LogIs o seguinte predicado palíndromo:

$@(l1, invert(l1, l1) -> \text{palindromo}(l1)).$

d) Foi introduzido no LogIs o seguinte objectivo:

$@(a, \#(e, \text{palindromo}(e) * \text{member}(a, e))).$

e) Foi introduzido no LogIs o seguinte objectivo:

$@(a, @(b, \#(lista, \text{palindromo}(lista) * \text{diml}(lista, \&\text{add}(2, 1)) * \text{member}(a, lista) * \text{member}(b, lista)))).$

Foram calculadas duas respostas:

$\text{lista} = l(\text{skolem10}, l(\text{skolem11}, l(\text{skolem10}, \text{nil})))$

$b = \text{skolem11}$

$a = \text{skolem10}$

e

lista = l(skolem11,l(skolem10,l(skolem11,nil)))

b = skolem11

a = skolem10 respostas:

4

a) Foram inseridas no LogIIS as duas seguintes premissas:

@(nome,@(c,@(r, member(nome,c) -> procurar(l(c,r), nome,0)))).

@(r, @(c, @(nome, @(numeroCarroagem, procurar(r, nome, numeroCarroagem) -> procurar(l(c,r), nome, &add(numeroCarroagem,1)))))).

Foi inserido o objectivo:

#(x,procurar(l(nil,l(l(jorge,l(paula,l(ana,nil))),l(nil,nil))), paula, x)).

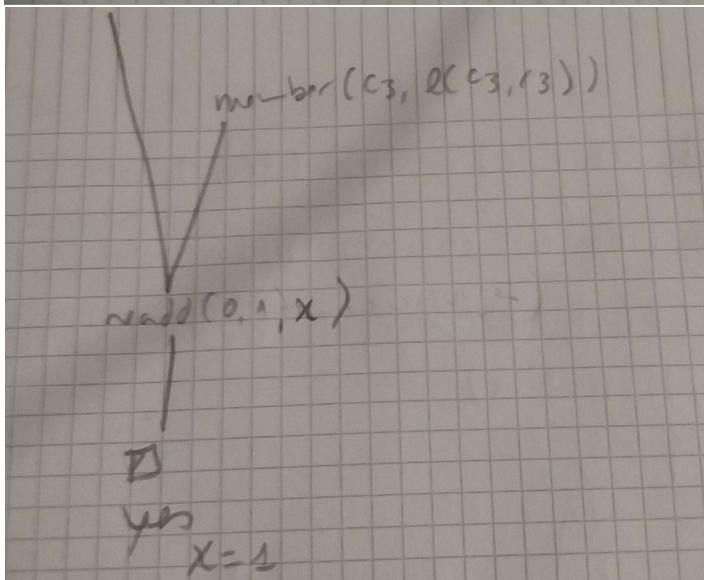
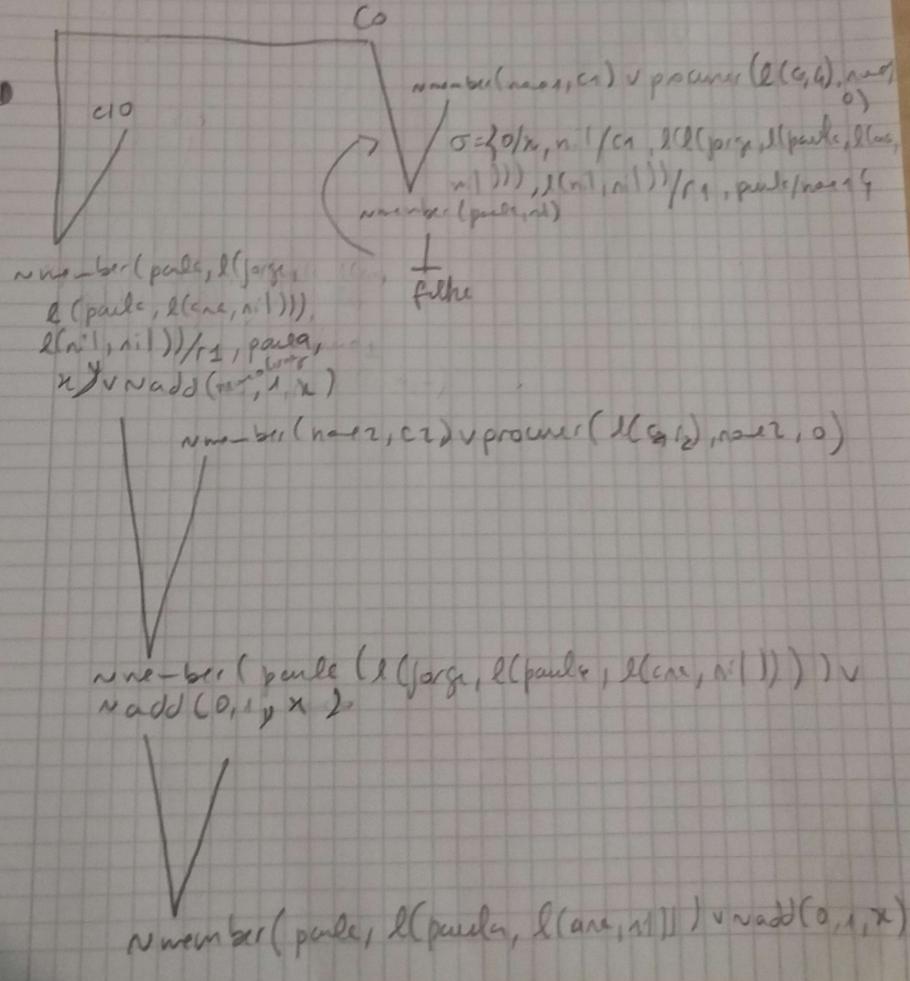
b) Foi calculada a árvore de resolução para o objectivo

#(x,procurar(l(nil,l(l(jorge,l(paula,l(ana,nil))),l(nil,nil))), paula, x)).

C9: $\neg \text{member}(\text{none}, c) \vee \text{procuer}(\ell(c, r), \text{none}, 0)$

C10: $\neg \text{procuer}(r, \text{none}, \text{none}(\text{corbage})) \vee \text{procuer}(\ell(c, r), \text{none}, 0)$

C0: $\neg \text{procuer}(\ell(r1), \ell(\ell(jorge, \ell(paula, \ell(ana, n1)))),$
 $\ell(n1))), paula, 2)$



3ª Parte – Prolog (iniciação)

1

Foram escritas as cláusulas referentes a premissas e objectivos na forma clausal. A sua tradução para Prolog foi incluída nos ficheiros 3.1.premissas.pl e 3.1.objectivos.pl.

3ª parte

1.

$F_1 : (\forall n) (\text{animal}(n) \rightarrow \text{morre}(n))$

(1) $(\forall n) (\text{animal}(n) \vee \text{morre}(n))$

(2) -

(3) -

(4) -

(5) -

(6) $(\text{animal}(n) \vee \text{morre}(n))$

(7) -

(8) { $\text{animal}(n) \vee \text{morre}(n)$ }
morre(X) :- animal(X).

$F_2 = (\forall n) (\text{animal}(n))$

(1) -

(2) -

(3) -

(4) -

(5) -

(6) animal(X)

(7) -

(8) { animal(X) }
animal(X).

F3. $(\forall n) (\text{gato}(n) \rightarrow (\exists y) \text{neto}(y) \wedge \text{cone}(n, y))$

① $(\forall n) (\neg \text{gato}(n) \vee ((\exists y) \text{neto}(y) \wedge \text{cone}(n, y)))$

② -

③ $(\forall n) (\exists y) (\neg \text{gato}(n) \vee (\text{neto}(y) \wedge \text{cone}(n, y)))$

④ $(\forall n) (\neg \text{gato}(n) \vee (\text{neto}(\text{skoleo}) \wedge \text{cone}(n, \text{skoleo}(n))))$

⑤ -

⑥ $\neg \text{gato}(n) \vee (\text{neto}(\text{skoleo}) \wedge \text{cone}(n, \text{skoleo}(n)))$

⑦ $(\neg \text{gato}(n) \vee \text{neto}(\text{skoleo})) \wedge (\neg \text{gato}(n) \vee \text{cone}(n, \text{skoleo}))$

⑧ } $\neg \text{gato}(n) \vee \text{neto}(\text{skoleo})$

$\neg \text{gato}(n) \vee \text{cone}(n, \text{skoleo}(n))$ }

$\text{neto}(\text{Guider}) := \neg \text{gato}(X),$

$\text{cone}(X, \text{Guider}(X)) := \text{gato}(X).$

F4: $\text{gato}(\text{tanaco}) \wedge \text{gato}(\text{furusco}) \wedge \text{cas}(\text{pluto})$

① -

② -

③ -

④ -

⑤ -

⑥ -

⑦ -

⑧ } $\text{gab}(\text{tanaco}), \text{gato}(\text{furusco}), \text{cas}(\text{pluto})$ }

$\text{gato}(\text{tanaco})$

$\text{gato}(\text{furusco})$

$\text{cas}(\text{pluto}).$

$$F_5 = (\forall n) \left(\text{gato}(n) \rightarrow (\text{mío}(n) \wedge \text{arrante}(n)) \right)$$

$$\textcircled{1} \quad (\forall n) \sim \text{gato}(n) \vee (\text{mío}(n) \wedge \text{arrante}(n))$$

\textcircled{2} -

\textcircled{3} -

\textcircled{4} -

\textcircled{5} -

$$\textcircled{6} \quad \sim \text{gato}(n) \vee (\text{mío}(n) \wedge \text{arrante}(n))$$

$$\textcircled{7} \quad (\sim \text{gato}(n) \vee \text{mío}(n)) \wedge (\sim \text{gato}(n) \vee \text{arrante}(n))$$

$$\textcircled{8} \quad \left\{ \begin{array}{l} \sim \text{gato}(n) \vee \text{mío}(n), \\ \sim \text{gato}(n) \vee \text{arrante}(n) \end{array} \right\}$$

$$\text{mío}(x) := \text{gato}(x).$$

$$\text{arrante}(x) := \text{gato}(x).$$

$$F_6 = (\forall n) \left(\text{caz}(n) \wedge (\text{muyHumor}(n) \vee \text{muy}(n)) \rightarrow \text{morde}(n) \right)$$

$$\textcircled{1} \quad (\forall n) \sim (\text{caz}(n) \wedge (\text{muyHumor}(n) \vee \text{muy}(n))) \vee \text{morde}(n)$$

$$\textcircled{2} \quad (\forall n) \sim \text{caz}(n) \vee \sim (\text{muyHumor}(n) \vee \text{muy}(n)) \vee \text{morde}(n)$$

$$(\forall n) \sim \text{caz}(n) \vee (\sim \text{muyHumor}(n) \wedge \sim \text{muy}(n)) \vee \text{morde}(n)$$

\textcircled{3} -

\textcircled{4} -

\textcircled{5} -

$$\textcircled{6} \quad \sim \text{caz}(n) \vee \left(\begin{array}{l} \text{muyHumor}(n) \wedge \text{muy}(n) \\ \text{morde}(n) \end{array} \right) \vee \text{morde}(n)$$

$$\textcircled{7} \quad \left(\begin{array}{l} \sim \text{caz}(n) \vee \sim \text{muyHumor}(n) \vee \text{morde}(n) \\ \sim \text{caz}(n) \vee \text{muy}(n) \vee \text{morde}(n) \end{array} \right) \wedge$$

$$(\sim \text{caz}(n) \vee \sim \text{muy}(n) \vee \text{morde}(n))$$

$$\textcircled{8} \quad \left\{ \begin{array}{l} \sim \text{caz}(n) \vee \sim \text{muyHumor}(n) \vee \text{morde}(n), \\ \sim \text{caz}(n) \vee \text{muy}(n) \vee \text{morde}(n) \end{array} \right\}$$

$$\text{morde}(x) := \text{caz}(x), \text{muy}(x) -$$

$$\text{morde}(x) := \text{caz}(x), \text{muyHumor}(x).$$

$G_1 = \text{arranha}(\text{tareco}) \wedge \text{arranha}(\text{farrusco}) \wedge$
 $\quad (\exists y)(\exists x) \text{ rato}(y) \wedge \text{rato}(x) \wedge \text{cate}(tareco, y) \wedge \neg r(x)$
 $\quad \text{cate}(\text{farrusco}, x)$
 $\neg G_1 = \text{arranha}(\text{tareco}) \vee \text{arranha}(\text{farrusco}) \vee$
 $\quad \neg (\exists y) \text{ rato}(y) \vee \neg (\exists x) (\text{rato}(x) \wedge \text{cate}(\text{tareco}, y))$
 $\vee \neg \text{cate}(\text{farrusco}, x)$
 $\neg G_1 = \forall y \forall x \neg$
 $\quad \left\{ \begin{array}{l} \text{arranha}(\text{tareco}) \vee \text{arranha}(\text{farrusco}) \\ \neg \text{rato}(y) \vee \neg \text{rato}(x) \vee \neg \text{cate}(\text{farrusco}, x) \\ \vee \neg \text{cate}(\text{tareco}, y) \end{array} \right.$

$G_2 = \text{morde}(\text{pluto}) \vee \text{arranha}(\text{farrusco})$.
 $\neg G_2 = \neg \text{morde}(\text{pluto}) \wedge \neg \text{arranha}(\text{farrusco})$
 $\quad \left\{ \begin{array}{l} \neg \text{morde}(\text{pluto}), \\ \neg \text{arranha}(\text{farrusco}) \end{array} \right.$
 $G_3 = (\exists x) \text{ rato}(x) \wedge (\exists y) \text{ gato}(y) \rightarrow \text{cate}(y, x)$
 $\neg G_3 = \neg (\exists x) \text{ rato}(x) \wedge \neg (\exists y) \text{ gato}(y) \wedge \neg \text{cate}(y, x)$
 $\quad \text{Por conséq. ao logis obtém-se:}$
 $\quad \left\{ \begin{array}{l} \neg \text{rato}(x), \\ \neg \text{gato}(y), \\ \neg \text{cate}(y, x) \end{array} \right.$
 $G_4 = \forall x \text{ morre}(x)$
 $\neg G_4 = \neg \forall x \text{ morre}(x)$
 $= \exists x \neg \text{morre}(x)$
 $= \neg \text{morre}(\text{skolem})$
 $\quad \left\{ \begin{array}{l} \neg \text{morre}(\text{skolem}) \end{array} \right.$

Exploração do programa para G1:

O programa retorna true, pois se sabe que o tareco e o farrusco são gatos e, por isso, arranham.

Exploração do programa para G2:

Não é possível calcular uma resposta, pois não há cláusula para combinar com mau(x). O sistema retorna “ERROR: undefined procedure mau/1”.

Exploração do programa para G3:

Exploração do programa para G4:

O objectivo retorna true, depois de combinar com C2 e C1.

2

Quanto ao primeiro objectivo, quando a cláusula :-select(ana,X), capable(X). combina com select(X2,Y2) :- likes(X2, Y2)., o resultado é likes(ana,X), capable(X). Essa cláusula combina com

`likes(ana,rui)` e há lugar a backtracking. De seguida, `likes(ana,X)`, `capable(X)` combina com `likes(ana,pedro)`, dando-se a unificação de X com pedro. Há lugar a resolução com `capable(pedro)` e é respondido “YES”.

O segundo objectivo tem um corte na clausula `select1(X,Y) :- likes(X,Y), !.`, o que significa que depois de substituir X por ana e Y por rui, não existe backtracking, o que não permite que a clausula `likes(ana,pedro)` seja combinada com a clausula `-likes(X,Y)`, por esse motivo a resposta é “NO”.

Quanto ao terceiro objectivo, quando a clausula `-likes(X,Y)` é utilizada na resolução, ambos X e Y já estão unificados com ana e pedro, por isso a única clausula que combina com `-likes(x,y)` é `likes(ana,pedro)`, não havendo lugar a backtracking e a resposta é SIM.

3

a)

- a. `fact(0,X)` combina com `fact(0,1)`. X unifica com 1 e é essa a resposta obtida ($X=1$).
- b. `fact1(0,X)` combina com `fact1(0,1)`. X unifica com 1 e é essa a resposta obtida ($X=1$).
- c. `fact2(0,X)` combina com `fact2(0,1)`. X unifica com 1 e é essa a resposta obtida ($X=1$).
- d. `fact3(0,X)` combina com `fact3(0,suc(0))`. X unifica com `suc(0)` e é essa a resposta obtida ($X=suc(0)$).

b) Ao invocar o objectivo `fact(3,X)`, este não é resolvido, pois o predicado `N1 * X1` não sucede, dado que `X1` não está instanciado. Nos outros casos, a unificação elabora as substituições à medida que se avança na árvore de resolução, de modo que quando se chega ao caso de paragem `fact1(0,X3)`, `fact2(0,X3)`, `fact3(0,X3)` se podem elaborar substituições nos resolventes calculados e chegar à resposta $X = 6$, ou $X = suc(suc(suc(suc(suc(0))))))$, que é o mesmo.

c)

- a. `fact(X,6)`. A resolução de 6 com `N1 * X1` não sucede, pelo que o predicado retorna falso.
- b. `fact1(X,6)`. O predicado `N1 is N-1` falha, pois o lado direito da expressão `is` não está totalmente iniciado (`N`). Por essa razão, o sistema reporta erro: “os argumentos não estão suficientemente instanciados”.
- c. `fact2(X,6)`. A avaliação da expressão `is` em `N is N1+1` só é feita depois da combinação com o caso terminal. Por essa razão, tanto `N1` como `Y` estão instanciados. A ordem de avaliação entre `N is N1+1` e `X is N*Y` é importante, pois só depois de unificar `N` com um valor numérico se dá a avaliação de `N*Y`, impedindo que ocorra o mesmo erro que em b).
- d. `fact3(X,suc(suc(suc(suc(suc(0))))))). fact3(N,Y)` serve neste caso para instanciar `Y`, que é o factorial de `N`. Assim, segundo a definição do factorial, o factorial de $N+1$ é igual ao factorial de $N * N+1$. A multiplicação é feita partindo da soma. Se $X*Y = M$, então $X * (Y+1) = M+X$.

d)

- a. `fact(X,Y)` pela mesma razão identificada nas alíneas anteriores para `fact`, só é possível calcular uma resposta, que é $X=0$ e $Y=1$.
- b. `fact1(X,Y)`. O mesmo erro detectado na alínea anterior, só permite calcular uma resposta $X=0$, e $Y=1$, como consta de uma das premissas do programa.

- c. fact2(X,Y). Pelas razões apontadas em 3.c)c., é possível calcular várias respostas. Há sucessivas resoluções até chegar ao caso terminal, e depois disso, o algoritmo da unificação permite calcular unificadores numéricos para X e Y.
- d. fact3(X,Y) ...