Logica del Primo Ordine
– ESERCITAZIONE –

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia
Alessandro Saetti
saetti@ing.unibs.it

Materiale per il Corso di Intelligenza Artificiale
(Prof. Alfonso Gerevini)

Esercizio

- Ogni persona lavora o come infermiere oppure come insegnante
  $\forall x$ Lavora(x,Insegnante) $\rightarrow$ $\neg$Lavora(x,Infermiere)
  $\forall x$ $\neg$Lavora(x,Insegnante) $\rightarrow$ Lavora(x,Infermiere)

- Gli infermieri sono maschi
  $\forall x$ Lavora(x,Infermiere) $\rightarrow$ Maschio(x)

- Le persone sono o maschi o femmine
  $\forall x$ $\neg$Machio(x)$\rightarrow$Femmina(x)
  $\forall x$ Maschio(x)$\rightarrow$$\neg$Femmina($x$)

- Steve è un maschio e Roberta è una femmina
  Maschio(Steve), Femmina(Roberta),

- Dimostrare che Roberta è un'insegnante.

Sintassi Otter

- Costanti, variabili, simboli di funzione e predicati: stringa composta da alfanumerici, _

- Predicati/funzioni: simbolo (argomenti separati da ,)
  NO spaziatura tra simbolo di predicato/funzione e (

- Nel contesto di clausole:

  – Le variabili iniziano con `u, v, w, x, y, z`.

  – Se il flag prolog_style_variables é ON:
    le variabili iniziano con _ oppure con una lettera maiuscola

- Nel contesto di formule:

  – Il simbolo é una variabile se e solo se è quantificato.

Sintassi Otter (cont.)

Operazione	Simbolo	Priorità
Quantificatore esistenziale	`exists`	-
Quantificatore universale	`all`	-

- Quantificazione richiesta in formule, da evitare in clausole

- Forma prefissa:
  `$Quantified(all, x, y, exists, z, |(P(x,y), -(Q(z)))).`

- Forma infissa: `all x all y exists z (P(x,y) | -Q(z)).`
  Forma semplificata: `all x y exists z (P(x,y) | -Q(z)).`

- Obbligatoria parentesi dopo uno (o più) quantificatori

- Spaziatura obbligatoria prima/dopo “all” e “exists” e prima “–”

- `=, eq, Eq, EQ` sono predicati di uguaglianza

- `!=` è il predicato di disuguaglianza

Esempio

Scrivere le rappresentazioni logiche delle seguenti formule secondo la sintassi di otter.

C'era una volta due talpe che vivevano in una buca. Nessuna delle due condivideva la tana con l'altra.

```
set(auto).

formula_list(usable).
% In ciascuna buca vive al piu' una talpa
all x y z ( talpa(x) & talpa(y) & buca(z) & abita(x,z) & -eq(x,y) -> -abita(y,z)).

% Ogni talpa vive in una buca.
all x (talpa(x) -> (exists z (buca(z) & abita(x,z)))).

% Esistono 2 talpe
exists x y ( talpa(x) & talpa(y) & x!=y).
all x (x=x).

% Esiste 1 e una sola buca.
exists x ( buca(x) & (all y ( y!=x -> -buca(y)))).
end_of_list.
```

Esempio (cont.)

```
1 [] -talpa(x)| -talpa(y)| -buca(z)| -abita(x,z)|eq(x,y)| -abita(y,z).
2 [] -talpa(x)|buca($f1(x)).
3 [] -talpa(x)|abita(x,$f1(x)).
4 [] $c2!=$c1.
5 [] x=$c3| -buca(x).
6 [] talpa($c2).
7 [] talpa($c1).
8 [] x=x.
9 [] buca($c3).
10 [hyper,6,3] abita($c2,$f1($c2)).
11 [hyper,6,2] buca($f1($c2)).
13 [hyper,7,3] abita($c1,$f1($c1)).
14 [hyper,7,2] buca($f1($c1)).
18,17 [hyper,11,5] $f1($c2)=$c3.
19 [back_demod,10,demod,18] abita($c2,$c3).
22,21 [hyper,14,5] $f1($c1)=$c3.
23 [back_demod,13,demod,22] abita($c1,$c3).
30,29 [hyper,23,1,6,7,9,19] eq($c2,$c1).
35 [back_demod,4,demod,30] $c1!=$c1.
36 [binary,35.1,8.1] $F.
```

Esercizio 1

Un'indagine fatta su un gruppo di lavoratori siderurgici ha fornito le seguenti informazioni:

- Almeno uno dei lavoratori intervistati non e' straniero e almeno un maschio e' straniero
$\exists x \ \neg\texttt{Straniero}(x)$. $\exists x$ $\texttt{Maschio(x)} \wedge \texttt{Straniero}(x)$.
- Se un lavoratore e' di colore e' straniero
$\forall x \ \texttt{Colore}(x) \rightarrow \texttt{Straniero}(x)$.
- Non e' vero che almeno un machio e' biondo
$\neg(\exists x \, \texttt{Maschio}(x) \wedge \texttt{Biondo}(x))$.

Utilizzare OTTER per mostare quali affermazioni sono vere.

A. Tutti gli stranieri sono di colore

B. Almeno un maschio straniero non e' biondo

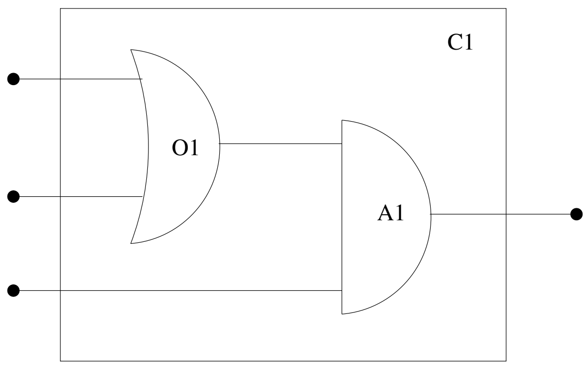
C. Nessun maschio bruno e' straniero

D. Almeno un maschio non e' straniero

E. Almeno una femmina e' straniera

Esercizio 2

Sia dato il seguente circuito elettronico.



Utilizzare OTTER per dimostrare che a fronte dell'ingresso $\langle 1, 0, 1 \rangle$ l'uscita è 1.

Esercizio 2 (cont.)

1. Se due terminali sono connessi, allora hanno lo stesso segnale
   $\forall t_1\, t_2\ \texttt{Connesso}(t_1, t_2) \rightarrow (\texttt{Segnale}(t_1) = \texttt{Segnale}(t_2))$
2. Il segnale ad ogni terminale è 1 oppure 0 ($1 \neq 0$)
   $\forall t\ \texttt{Segnale}(t)=1 \ \leftrightarrow\ \neg\texttt{Segnale}(t)=0$
   $1 \neq 0$
3. Il terminale A è connesso al terminale B sse il terminale B è connesso al terminale A
   $\forall t_1\, t_2\ \texttt{Connesso}(t_1, t_2) \leftrightarrow \texttt{Connesso}(t_2, t_1)$
4. Il segnale di uscita di una porta OR è 1 sse almeno uno dei due segnali di ingresso è 1
   $\forall g\ \texttt{Tipo}(g)=\texttt{OR} \ \rightarrow\ \texttt{Segnale}(\texttt{Out}(1, g))=1 \ \leftrightarrow$
   $(\texttt{Segnale}(\texttt{In}(1, g))=1 \ \vee\ \texttt{Segnale}(\texttt{In}(2, g))=1)$
5. Il segnale di uscita di una porta AND è 1 sse entrambi i due segnali di ingresso sono 1
   $\forall g\ \texttt{Tipo}(g)=\texttt{AND} \ \rightarrow\ \texttt{Segnale}(\texttt{Out}(1, g))=0 \ \leftrightarrow$
   $(\texttt{Segnale}(\texttt{In}(1, g))=0 \ \vee\ \texttt{Segnale}(\texttt{In}(2, g))=0)$

Esercizio 2 (cont.)

6. Porte logiche: `Tipo(O1)=OR`, `Tipo(A1)=AND`
7. Connessioni:
   `Connesso(Out(1,O1),In(1,A1))`
   `Conensso(In(1,C1),In(1,O1))`
   `Connesso(In(2,C1),In(2,O1))`
   `Connesso(In(3,C1),In(2,A1))`
   `Connesso(Out(1,A1),Out(1,C1))`
8. Segnali Input/Output:
   `Segnale(In(1,C1))=0`
   `Segnale(In(2,C1))=0`
   `Segnale(In(3,C1))=0`
   `Segnale(Out(1,C1))=1` (Contraddizione)

Esercizio 2 (cont.)

```
set(auto).

formula_list(usable).

% Conoscenza del dominio dei circuiti
all g (tipo(g,OR) ->((signal(out(1,g))=1)<->(signal(in(1,g))=1 | signal(in(2,g))=1))).
all g (tipo(g,AND) ->((signal(out(1,g))=0)<->(signal(in(1,g))=0 | signal(in(2,g))=0))).
all x y (conn(x,y) <-> conn(y,x)).
all x y (conn(x,y) -> eq(signal(y),signal(x))).
all x ((signal(x)=1) <-> -(signal(x)=0)).
-(1=0).
```

Esercizio 2 (cont.)

```
% Porte logiche
tipo(A1,AND).
tipo(O1,OR).

% Connessioni
conn(in(1,C1),in(1,O1)).
conn(in(2,C1),in(2,O1)).
conn(in(3,C1),in(2,A1)).
conn(out(1,O1),in(1,A1)).
conn(out(1,A1),out(1,C1)).

% Segnali di Input/Output
(signal(in(1,C1))=1).
(signal(in(2,C1))=0).
(signal(in(3,C1))=1).
-(signal(out(1,C1))=1).
end_of_list.
```

Esercizio 3 (per casa)

Date le seguenti affermazioni:

- Gli studenti che viaggiano su un qualsiasi taxi non possiedono un auto,
- I tassisti possiedono un auto,
- I tassisti viaggiano su almeno un taxi,

Utilizzare Otter per dimostrare che i tassisti non sono studenti.

Esercizio 4 (per casa)

Si supponga che un ladro è arrestato da al più un poliziotto (ovvero se un ladro è arrestato da un poliziotto allora il ladro non può essere arrestato da un secondo differente poliziotto).

Utilizzare Otter per dimostrare che non può essere che due (differenti) poliziotti arrestano uno stesso ladro.

Esercizio 5 (per casa)

Date le seguenti affermazioni:

- Tutti gli italiani votano per almeno un partito,
- Gli astenuti non votano per i partiti,
- Gli astenuti sono italiani

Utilizzare Otter per dimostrare che non possono esistere astenuti.