

Logica Proposizionale

– 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 1

Se l'unicorno è un animale mitico, è immortale, ma se non è mitico, allora è un mammifero mortale. Se è immortale o un mammifero, allora ha un corno. L'unicorno è magico se ha un corno.

Scrivere le rappresentazioni logiche delle precedenti formule in modo tale che siano utilizzabili con la risoluzione unitaria.
Dimostrare per refutazione e tramite la strategia basata su insieme di supporto quanto segue.

- *L'unicorno è mitico?*
- *L'unicorno è magico?*
- *L'unicorno ha un corno?*

Esercizio 1 (Soluzione)

Se l'unicorno e' un animale mitico e' immortale

1. $\neg \text{mitico} \vee \text{immortale}$

Se non e' mitico allora e' un mammifero mortale

2. $\text{mitico} \vee \neg \text{immortale}$

3. $\text{mitico} \vee \text{mammifero}$

Se e' immortale o e' un mammifero allora ha un corno

4. $\neg \text{immortale} \vee \text{cornuto}$

5. $\neg \text{mammifero} \vee \text{cornuto}$

L'unico e' magico se ha un corno

6. $\neg \text{cornuto} \vee \text{magico}$

Esercizio 2

John lo sfregiato, Bill lo squartatore e Jack il duro vengono interrogati con l'accusa di aver rubato la cassetta delle elemosine della parrocchia. Jack dice che il furto e' stato commesso da Bill; Bill si professa innocente e John afferma di non aver compiuto il furto. Sapendo che solo uno dei tre dice la verità, chi è il ladro.

Scrivere le rappresentazioni logiche delle precedenti formule in modo tale che siano utilizzabili con la risoluzione unitaria.
Scoprire per refutazione e tramite la strategia di risoluzione basata su insieme di supporto chi tra John, Jack e Bill ha commesso il furto.

Suggerimento:

Usare i letterali FurtoBill, FurtoJack, FurtoJohn, VeritaBill, VeritaJack, VeritaJohn

Esercizio 2 (Soluzione)

Le testimonianze dei tre ceffi

$\text{FurtoBill} \leftrightarrow \text{VeritaJack}$

$\neg \text{FurtoBill} \leftrightarrow \text{VeritaBill}$

$\neg \text{FurtoJohn} \leftrightarrow \text{VeritaJohn}$

Solo uno dei tre ceffi dice la verita

$\text{VeritaJack} \leftrightarrow \neg \text{VeritaBill} \wedge \neg \text{VeritaJohn}$

$\text{VeritaBill} \leftrightarrow \neg \text{VeritaJack} \wedge \neg \text{VeritaJohn}$

$\text{VeritaJohn} \leftrightarrow \neg \text{VeritaBill} \wedge \neg \text{VeritaJack}$

Esercizio 2 (Soluzione)

Le testimonianze dei tre ceffi

1. $\neg \text{FurtoBill} \vee \text{VeritaJack}, \text{FurtoBill} \vee \neg \text{VeritaJack}$

2. $\text{FurtoBill} \vee \text{VeritaBill}, \neg \text{FurtoBill} \vee \neg \text{VeritaBill}$

3. $\text{FurtoJohn} \vee \text{VeritaJohn}, \neg \text{FurtoJohn} \vee \neg \text{VeritaJohn}$

Solo uno dei tre ceffi dice la verita

4. $\neg \text{VeritaJack} \vee \neg \text{VeritaBill}, \neg \text{VeritaJack} \vee \neg \text{VeritaJohn},$
 $\text{VeritaJack} \vee \text{VeritaBill} \vee \text{VeritaJohn}$

5. $\neg \text{VeritaBill} \vee \neg \text{VeritaJohn}, \neg \text{VeritaBill} \vee \neg \text{VeritaJack},$
 $\text{VeritaBill} \vee \text{VeritaJohn} \vee \text{VeritaJack}$

6. $\neg \text{VeritaJohn} \vee \neg \text{VeritaBill}, \neg \text{VeritaJohn} \vee \neg \text{VeritaJack},$
 $\text{VeritaJohn} \vee \text{VeritaBill} \vee \text{VeritaJack}$

Esercizio 2 (Soluzione)

$\text{KB} = \{\neg \text{FurtoBill} \vee \text{VeritaJack}, \text{FurtoBill} \vee \text{VeritaBill}, \text{FurtoJohn} \vee \text{VeritaJohn},$
 $\text{FurtoBill} \vee \neg \text{VeritaJack}, \neg \text{VeritaJack} \vee \neg \text{VeritaJohn}, \neg \text{VeritaJohn} \vee \neg \text{VeritaBill}, \dots\}$

1. $S = \{\neg \text{FurtoJohn}\} \quad \frac{\text{FurtoJohn} \vee \text{VeritaJohn}, \neg \text{FurtoJohn}}{\text{VeritaJohn}}$

2. $S = \{\text{VeritaJohn}\} \quad \frac{\neg \text{VeritaJohn} \vee \neg \text{VeritaBill}, \text{VeritaJohn}}{\neg \text{VeritaBill}},$
 $\frac{\neg \text{VeritaJack} \vee \neg \text{VeritaJohn}, \text{VeritaJohn}}{\neg \text{VeritaJack}}$

3. $S = \{\neg \text{VeritaBill}, \neg \text{VeritaJack}\} \quad \frac{\text{FurtoBill} \vee \text{VeritaBill}, \neg \text{VeritaBill}}{\text{FurtoBill}}$

4. $S = \{\neg \text{VeritaJack}, \text{FurtoBill}\} \quad \frac{\neg \text{FurtoBill} \vee \text{VeritaJack}, \neg \text{VeritaJack}}{\neg \text{FurtoBill}}$

5. $S = \{\text{FurtoBill}\} \quad \frac{\text{FurtoBill} \vee \text{VeritaBill}, \neg \text{FurtoBill}}{\text{VeritaBill}},$

$\frac{\text{FurtoBill} \vee \neg \text{VeritaJack}, \neg \text{FurtoBill}}{\neg \text{VeritaJack}}$

6. $S = \{\text{FurtoBill}, \text{VeritaBill}, \neg \text{VeritaJack}\} \quad \frac{\text{FurtoBill}, \neg \text{FurtoBill}}{\text{F}}$

Applicazioni Dimostratori di Teoremi

- Dimostrazione, supervisione e verifica di teoremi
 - Teorema di incompletezza di Godel
- Verifica software, hardware, protocolli di Sicurezza
 - Sistema di controllo dello space-shuttle
 - Divisione Floating-Point per il processore AMD5k86
 - Protocollo RSA
- Generazione di software
 - Programmi per comandare satelliti

Otter

- Otter = Organized Techniques for Theorem-proving and Effective Research
- È basato su logica del primo ordine
- Sistema semi-automatico
- Funziona anche in batch mode: `otter < input > output`
- **Indirizzi Internet utili:**
Home Page di Otter: <http://www.mcs.anl.gov/AR/otter/>
Manuale: http://www-unix.mcs.anl.gov/AR/otter/otter3_manual.pdf

Sintassi

- Nomi ordinari: stringa composta da alfanumerici, `_`
- Caratteri speciali:
 - `%`: commenti
 - `.` : fine espressione
 - `,()` : punteggiatura e simboli di raggruppamento
- Nomi che iniziano con `$` sono riservati (Es. `$AND`)
- Lettere minuscole differenti da lettere maiuscole

Sintassi - Clausole

Operatori	Simbolo	Priorità
Negazione	<code>-</code>	1
Disgiunzione	<code> </code>	3

- Forma prefissa: `I( -(a), I( -(b), c) ).`
- Forma infissa: `-a | -b | c.`
- Spaziatura obbligatoria prima di `"_"`
- Tutte le clausole terminano con `"."`

Sintassi - Formule

Operazione	Simbolo	Priorità
Negazione	<code>-</code>	1
Congiunzione	<code>&</code>	2
Disgiunzione	<code> </code>	3
Implicazione	<code>-></code>	4
Equivalenza	<code><-></code>	4

- Tutte le formule terminano con `"."`
- Otter trasforma le formule in clausole

Liste di Clausole/Formule

Otter utilizza la strategia basata su insieme di supporto

- usable: clausole/formule usate per l'inferenza
- sos: clausole/formule che partecipano alla ricerca
- passive: clausole/formule usate per controllare il processo in dimostrazioni difficili (sussunzioni/unit-conflict)
- demodulators: uguaglianze utilizzate per la riscrittura di clausole/formule inferite

Comandi e File di Input

- Otter riconosce due tipi di opzioni:
 - flag: opzioni booleane assegnate con `set` e `clear`
ES: `set(binary_res);`
 - parametri: valori interi assegnati con `assign`
ES: `assign(max_seconds,100).`
- Le liste iniziano con `list(list_name).`, `formula_list(list_name).`
`list_name` è `usable`, `sos`, `demodulators`, oppure `passive`
- Le liste terminano con `end_of_list.`

Comandi e File di Input

Comando	Descrizione
<code>include(file_name)</code>	include file_name nel file corrente
<code>op(precedenza, tipo, nome)</code>	dichiara un operatore
<code>make_evaluable(simb, simb_valutato)</code>	dichiara un alias di simbolo_valutato
<code>set(flag_name)</code>	setta il flag flag_name
<code>clear(flag_name)</code>	resetta il flag flag_name
<code>assign(parameter_name, integer)</code>	parameter_name=integer
<code>list(list_name)</code>	lista delle clausole
<code>formula_list(list_name)</code>	lista delle formule
<code>weight_list(weight_list_name)</code>	lista dei pesi
<code>lex(symbol_list)</code>	assegna un ordinamento ai simboli

Modalità Automatica

- Impostata tramite il flag auto: `set(auto)`
- `set(auto)` DEVE essere il primo comando
- La KB è interamente in `list(usable)`
- Otter decide:
 - regole di inferenza;
 - strategia di ricerca;
 - lista sos.

Esempio

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

Due talpe vivono in una buca. Nessuna delle due vuole condividere la tana con l'altra.

```
set(auto).
list(usable).
% Ogni talpa vive in un buco.
ViveT1B1.
ViveT2B1.
% Nella buca vive al piu' una talpa.
-ViveT1B1 | -ViveT2B1.
end_of_list.
```

Regole di inferenza

- **Binary resolution** (flag `binary_res`)
Risoluzione con due clausole.
$$\frac{infermiera(Roberta), \neg infermiera(Roberta)}{False}$$
- **Unit resulting resolution** (flag `ur_res`)
Il risultato è una clausola unitaria.

Data una clausola con n termini e $n - 1$ clausole unitarie le risolve in un colpo solo.
$$\frac{\neg P(x,y) \vee Q(y,z) \vee \neg R(u,v) \vee S(x,z), \quad P(John,Mary), \neg Q(Mary,Cats), R(Birds,Mice)}{S(John,Cats)}$$
- **Hyper-Resolution** (flag `hyper_res`)
Una clausola ha almeno un letterale negativo, mentre le altre non ne hanno. Il risultato non contiene letterali negativi.
$$\frac{\neg P(x,y) \vee Q(x) \vee \neg R(y) \quad P(Spock,Data) \quad R(z) \vee S(z)}{Q(Spock) \vee S(Data)}$$

Regole di inferenza

- **Paramodulation**
Uno degli argomenti del predicato di uguaglianza deve unificare con un sottotermine dell'altra clausola.
$$\frac{difference(two_squared,b) \quad equal(two_squared,four)}{difference(four,b)}$$
 - **Paramodulation from the given clause** (flag `para_from`): la given clause contiene un predicato di uguaglianza
 - **Paramodulation into the given clause** (flag `para_into`): il predicato di uguaglianza non è contenuto nella given clause, ma il risultato dell'inferenza è la sostituzione di un parametro della given_clause.

Caratteristiche

- La KB è specificata tramite clausole oppure formule del primo ordine
- Forward demodulation: semplifica le clausole inferite
- Backward Demodulation: semplifica le clausole esistenti
- Forward Sussumption: cancella le clausole inferite
- Backward Sussumption: cancella le clausole esistenti
- pesi e ordinamento lessicale

Algoritmo di Otter

```
while (sos non vuota & nessuna dimostrazione e' stata trovata)
begin
  given_clause := migliore_clausola(sos);
  usable := usable U given_clause;
  sos := sos - given_clause;
  new_clauses := inferisci(given_clause, clause_usables)
  if retention_test(new_clauses) then
    sos:=sos U new_clauses;
endwhile.
```

Esempio (cont.)

Due talpe vivono in una buca. Nessuna delle due vuole condividere la tana con l'altra.

Utilizzare OTTER per dimostrare che non è possibile una convivenza civile.

```
given clause #1: (wt=3) 3 [] ViveT2B1.
** KEPT (pick-wt=0): 4 [hyper,3,2,1].
-----> EMPTY CLAUSE at 0.00 sec ----> 4 [hyper,3,2,1] $F.
```

In conclusione, la dimostrazione è la seguente:

```
1 [] ViveT1B1.
2 [] -ViveT1B1 | -ViveT2B1.
3 [] ViveT2B1.
4 [hyper,3,2,1] $F.
```

Esercizio 1

John lo sfregiato, Bill lo squartatore e Jack il duro vengono interrogati con l'accusa di aver rubato la cassetta delle elemosine della parrocchia. Jack dice che il furto e' stato commesso da Bill; Bill si professa innocente e John afferma di non aver compiuto il furto. Sapendo che solo uno dei tre dice la verità, chi è il ladro.

Scrivere le rappresentazioni logiche delle seguenti formule secondo la sintassi di OTTER ed utilizzare il dimostratore di teoremi per scoprire chi tra Bill, John e Jack ha commesso il furto.

Esercizio 1 (Soluzione)

```
set(ur_res).
```

```
list(usable).
% le testimonianze dei tre ceffi
-FurtoBill | VeritaJack.
FurtoBill | VeritaBill.
FurtoJohn | VeritaJohn.
% solo uno dei tre ceffi dice la verita
-VeritaJack | -VeritaBill.
-VeritaJack | -VeritaJohn.
-VeritaJohn | -VeritaBill.
end_of_list.
```

```
list(sos).
-FurtoJohn.
end_of_list.
```

Esercizio 2

Estensione dell'esempio delle 2 talpe a 3 talpe:

Tre talpe vivono in una valle con due buche. Nessuna delle talpe vuole condividere la tana con un'altra talpa.

Scrivere le rappresentazioni logiche delle seguenti formule secondo la sintassi di otter ed utilizzare il dimostratore di teoremi per dimostrare che non è possibile una convivenza civile.

Esercizio 2 (Soluzione)

```
set(auto).
list(usable).
% Ogni talpa vive in un buco.
ViveT1B1 | ViveT1B2.
ViveT2B1 | ViveT2B2.
ViveT3B1 | ViveT3B2.
% In ciascun buco vive al piu' una talpa.
-ViveT1B1 | -ViveT2B1.
-ViveT1B1 | -ViveT3B1.
-ViveT2B1 | -ViveT3B1.
-ViveT1B2 | -ViveT2B2.
-ViveT1B2 | -ViveT3B2.
-ViveT2B2 | -ViveT3B2.
end_of_list.
```

Esercizio 3

Sapendo che gli enigmi sono dei misteri e che non ci sono indovinelli che non sono enigmi, scrivere le rappresentazioni logiche delle seguenti formule secondo la sintassi di OTTER ed utilizzare il dimostratore di teoremi per determinare quali tra le seguenti affermazioni sono vere:

- *I misteri sono indovinelli.*
- *Non ci sono misteri che non sono enigmi.*
- *Gli indovinelli sono misteri.*
- *Non ci sono enigmi che non sono indovinelli.*
- *Gli enigmi sono indovinelli.*

Esercizio 3 (Soluzione)

```
set(auto).
formula_list(usable).
enigma -> mistero.
-(indovinello & -enigma).

% A. -(mistero -> indovinello).

% B. -(-(mistero & -enigma)).

% C: -(indovinello -> mistero).

% D: -(-(enigma & -indovinello)).

% E. -(enigma -> indovinello).
end_of_list.
```