Università degli Studi di Roma "La Sapienza"
Facoltà di Ingegneria – Corso di Laurea Specialistica in Ingegneria Informatica
Corso di Metodi Formali nell'Ingegneria del Software
Prof. Toni Mancini

Esercizio **E.III.20060703**

versione del 4 luglio 2007

Si considerino i seguenti requisiti:

> Un ristorante offre come portate primi, secondi e dolci; ogni portata appartiene ad una sola categoria. Fra i secondi, vengono offerti quelli di carne. Le lasagne sono sia primi sia secondi di carne.

1. Rappresentare i requisiti mediante un opportuno diagramma UML.
2. Rappresentare i requisiti in un linguaggio formale fra quelli considerati durante il corso.
3. Considerando la rappresentazione formale, quali deduzioni interessanti possono essere effettuate?
4. Quale strumento software fra quelli utilizzati nel corso utilizzereste per dimostrare tali deduzioni?
5. Per lo strumento software scelto, fornire il file di input e l'output atteso.

Una possibile soluzione è riportata nella pagina seguente.

Soluzione

1. Uml: cfr. figura 1 (diagramma delle classi).

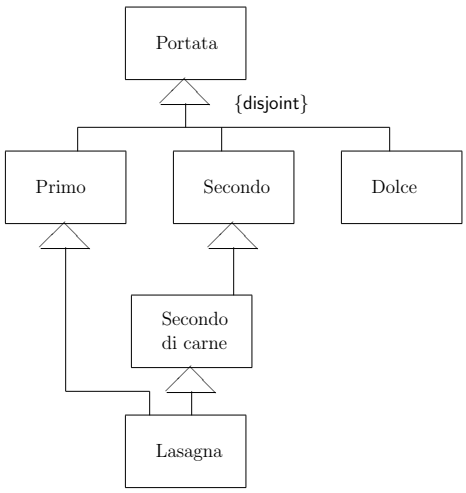

Figura 1: Diagramma Uml per la domanda 1

2. Rappresentiamo i requisiti in logica del prim'ordine:

$$
\begin{array}{ll}
 & //ISA \\
\forall X & Primo(X) \rightarrow Portata(X), \\
\forall X & Secondo(X) \rightarrow Portata(X), \\
\forall X & Dolce(X) \rightarrow Portata(X), \\
\forall X & SecondoDiCarne(X) \rightarrow Secondo(X), \\
\forall X & Lasagna(X) \rightarrow Primo(X), \\
\forall X & Lasagna(X) \rightarrow SecondoDiCarne(X), \\
 & //DISJOINT \\
\forall X & Dolce(X) \rightarrow \neg Primo(X), \\
\forall X & Dolce(X) \rightarrow \neg Secondo(X), \\
\forall X & Secondo(X) \rightarrow \neg Primo(X).
\end{array}
$$

3. La classe *Lasagna* è inconsistente (o insoddisfacibile). Di conseguenza i requisiti contengono un errore (ad esempio, non possiamo assumere che le portate siano disgiunte).

   Formalmente, detta Φ la formula di cui al punto 2, vale la seguente implicazione logica:

   $$\Phi \models \forall X \ \neg Lasagna(X).$$

4. Per dimostrare la deduzione di cui al punto 3 possiamo usare OTTER, chiedendo una refutazione.

5. Il file OTTER è il seguente:

```
%%% File: portate.ott
%%% Time-stamp: "2006-06-13 15:37:29 marco"
%%% Formato: file di input per OTTER 3.3

set(auto).
set(display_terms).
formula_list(usable).
%%% ISA
all X (Primo(X) -> Portata(X)).
all X (Secondo(X) -> Portata(X)).
all X (Dolce(X) -> Portata(X)).
all X (SecondoDiCarne(X) -> Secondo(X)).
all X (Lasagna(X) -> Primo(X)).
all X (Lasagna(X) -> SecondoDiCarne(X)).
%%% DISJOINT
all X (Dolce(X) -> -Primo(X)).
all X (Dolce(X) -> -Secondo(X)).
all X (Secondo(X) -> -Primo(X)).

%%% Conseguenza logica: non esiste alcuna lasagna
exists X Lasagna(X).
end_of_list.
```

   Ci si aspetta che OTTER dimostri una contraddizione, ovvero che generi la clausola vuota.

   Infatti, l'output di OTTER comprende:

```
-----> EMPTY CLAUSE at   0.00 sec ----> 15 [hyper,13,9,12] $F.
```