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.20060718**

versione del 4 luglio 2007

Si considerino i seguenti requisiti:

> Nell'applicazione da realizzare sono di interesse gli studenti e gli studenti lavoratori. Degli studenti lavoratori interessa il codice fiscale (una stringa). Degli studenti interessa l'anno di nascita e l'età in anni, calcolata in base ad un dato anno.

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 nelle pagine seguenti.

Soluzione

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

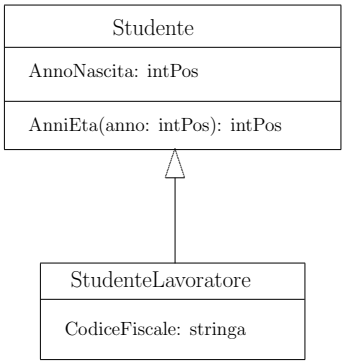

Figura 1: Diagramma Uml per la domanda

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

$$
\begin{array}{rl}
 & // \text{ Operazione classe } \mathit{Studente} \\
\forall XYZ & \mathit{AnniEta}(X,Y,Z) \rightarrow \mathit{Studente}(X) \wedge \mathit{IntPos}(Y) \wedge \mathit{IntPos}(Z), \\
\forall XYWZ & \mathit{AnniEta}(X,Y,W) \wedge \mathit{AnniEta}(X,Y,Z) \rightarrow W = Z, \\
 & // \text{ Attributo classe } \mathit{Studente} \\
\forall XY & \mathit{Studente}(X) \wedge \mathit{AnnoNascita}(X,Y) \rightarrow \mathit{IntPos}(X), \\
\forall X & \mathit{Studente}(X) \rightarrow \exists Y\ \mathit{AnnoNascita}(X,Y), \\
\forall XYZ & \mathit{Studente}(X) \wedge \mathit{AnnoNascita}(X,Y) \wedge \mathit{AnnoNascita}(X,Z) \rightarrow Y = Z, \\
 & // \text{ Attributo classe } \mathit{StudenteLavoratore} \\
\forall XY & \mathit{StudenteLavoratore}(X) \wedge \mathit{CodiceFiscale}(X,Y) \rightarrow \mathit{Stringa}(Y), \\
\forall X & \mathit{StudenteLavoratore}(X) \rightarrow \exists Y\ \mathit{CodiceFiscale}(X,Y), \\
\forall XYZ & \mathit{StudenteLavoratore}(X) \wedge \mathit{CodiceFiscale}(X,Y) \wedge \mathit{CodiceFiscale}(X,Z) \rightarrow Y = Z, \\
 & // \text{ Isa} \\
\forall X & \mathit{StudenteLavoratore}(X) \rightarrow \mathit{Studente}(X), \\
 & // \text{ Le classi sono disgiunte dai tipi} \\
\forall X & \mathit{Studente}(X) \rightarrow \neg \mathit{Stringa}(X), \\
\forall X & \mathit{Studente}(X) \rightarrow \neg \mathit{IntPos}(X), \\
 & // \text{ I tipi sono disgiunti fra loro} \\
\forall X & \mathit{Stringa}(X) \rightarrow \neg \mathit{IntPos}(X).
\end{array}
$$

3. Le deduzioni più interessanti sono le seguenti.

   (a) La classe *StudenteLavoratore* eredita l'attributo *AnnoNascita* con la sua molteplicità.

   (b) La classe *StudenteLavoratore* eredita l'operazione *AnniEta*.

   (c) La classe *Studente non* eredita l'attributo *CodiceFiscale*.

   Formalmente, detta Φ la formula di cui al punto 2, dette γ_1, γ_2 e γ_3 (rispettivamente) le formule:

$$
\begin{array}{rl}
 & // \ \mathit{StudenteLavoratore} \text{ eredita l'attributo } \mathit{AnnoNascita} \\
 & // \text{ l'attributo } \mathit{AnnoNascita} \text{ è monovalore in } \mathit{StudenteLavoratore} \\
\forall XY & \mathit{StudenteLavoratore}(X) \wedge \mathit{AnnoNascita}(X,Y) \rightarrow \mathit{IntPos}(Y), \\
\forall X & \mathit{StudenteLavoratore}(X) \rightarrow \exists Y\ \mathit{AnnoNascita}(X,Y), \\
\forall XYZ & \mathit{StudenteLavoratore}(X) \wedge \mathit{AnnoNascita}(X,Y) \wedge \mathit{AnnoNascita}(X,Z) \rightarrow Y = Z,
\end{array}
$$

detta γ_4 la formula:

$$\begin{array}{ll} & \text{// } \textit{StudenteLavoratore} \text{ eredita l'operazione } \textit{AnniEta} \\ \forall XYZ & \textit{StudenteLavoratore}(X) \wedge \textit{AnniEta}(X,Y,Z) \rightarrow \textit{IntPos}(Y) \wedge \textit{IntPos}(Z), \end{array}$$

detta γ_5 la formula:

$$\begin{array}{ll} & \text{// } \textit{Studente} \text{ eredita l'attributo } \textit{CodiceFiscale} \\ \forall XY & \textit{Studente}(X) \wedge \textit{CodiceFiscale}(X,Y) \rightarrow \textit{Stringa}(Y), \end{array}$$

valgono le seguenti relazioni:

$$\Phi \models \gamma_1 \wedge \gamma_2 \wedge \gamma_3, \quad (1)$$
$$\Phi \models \gamma_4, \quad (2)$$
$$\Phi \not\models \gamma_5. \quad (3)$$

4. Per dimostrare le varie proprietà utilizziamo strumenti differenti.
   (a) Per dimostrare la (1) possiamo usare OTTER, chiedendo una refutazione di $\Phi \wedge \neg(\gamma_1 \wedge \gamma_2 \wedge \gamma_3)$.
   (b) Per dimostrare la (2) possiamo usare OTTER, chiedendo una refutazione di $\Phi \wedge \neg\gamma_4$.
   (c) Per dimostrare la (3) possiamo usare MACE, chiedendo di trovare un modello di $\Phi \wedge \neg\gamma_5$.
5. Il file OTTER/MACE è il seguente:

```
%%% File: studenti.ott
%%% Time-stamp: "2006-07-17 21:13:51 cadoli"
%%% Formato: file di input per OTTER 3.3

set(auto).
formula_list(usable).

%% CLASSE STUDENTE
all X Y (Studente(X) & AnnoNascita(X,Y) -> IntPos(Y)). % TIPO ATTRIBUTO
all X (Studente(X) -> (exists Y AnnoNascita(X,Y))).    % ATTRIBUTO {1..*}
all X Y Z (Studente(X) & AnnoNascita(X,Y) & AnnoNascita(X,Z) -> Y=Z).
                                                       % ATTRIBUTO {0..1}
```

```
all X Y Z (AnniEta(X,Y,Z) -> Studente(X) & IntPos(Y) & IntPos(Z)).
                                                        % TIPO OPERAZIONE
all X Y W Z (AnniEta(X,Y,Z) & AnniEta(X,Y,W) -> W=Z).  % OPERAZIONE È FUNZIONE

%% CLASSE STUDENTELAVORATORE
all X Y (StudenteLavoratore(X) & CodiceFiscale(X,Y) -> Stringa(Y)).
                                                        % TIPO ATTRIBUTO
all X (StudenteLavoratore(X) -> (exists Y CodiceFiscale(X,Y))).
                                                        % ATTRIBUTO {1..*}
all X Y Z (StudenteLavoratore(X) & CodiceFiscale(X,Y) & CodiceFiscale(X,Z)
                                                  -> Y=Z).  % ATTRIBUTO {0..1}
%% ISA
all X (StudenteLavoratore(X) -> Studente(X)).

%% CLASSI DISGIUNTE DA TIPI
all X (Studente(X) -> -Stringa(X)).
all X (Studente(X) -> -IntPos(X)).
%% TIPI DISGIUNTI FRA LORO
all X (Stringa(X) -> -IntPos(X)).

%% CONSEGUENZE LOGICHE (usare otter):
%% 1. StudenteLavoratore EREDITA ATTRIBUTO AnnoNascita
%C1 <-> (all X Y (StudenteLavoratore(X) & AnnoNascita(X,Y) -> IntPos(Y))).
%% 2. ATTRIBUTO AnnoNascita È MONOVALORE
%C2 <-> (all X (StudenteLavoratore(X) -> (exists Y AnnoNascita(X,Y)))).
%C3 <-> (all X Y Z (StudenteLavoratore(X) & AnnoNascita(X,Y) &
%        AnnoNascita(X,Z) -> Y=Z)).
%-(C1 & C2 & C3).

%% CONSEGUENZE LOGICHE (usare otter):
%% 3. StudenteLavoratore EREDITA OPERAZIONE AnniEta
C4 <-> (all X Y Z (AnniEta(X,Y,Z) & StudenteLavoratore(X) ->
        IntPos(Y) & IntPos(Z))).
-C4.

%% NON È UNA CONSEGUENZA (usare mace)
%C5 <-> (all X Y (Studente(X) & CodiceFiscale(X,Y) -> Stringa(Y))).
%-C5.
end_of_list.
```

(a) Ci si aspetta che OTTER (per `-(C1 & C2 & C3).`) dimostri una contraddizione, ovvero che generi la clausola vuota.
Infatti, l'output di OTTER (linea di comando `otter.exe < studenti.ott`) comprende:

```
-----> EMPTY CLAUSE at   0.01 sec ----> 117 [hyper,116,21,73,107] $F.
```

(b) Ci si aspetta che OTTER (per `-C4.`) dimostri una contraddizione, ovvero che generi la clausola vuota.
Infatti, l'output di OTTER (linea di comando `otter.exe < studenti.ott`) comprende:

```
----> UNIT CONFLICT at   0.01 sec ----> 43 [binary,42.1,16.1] $F.
```

(c) Ci si aspetta che MACE (per `-C5.`) dimostri la soddisfacibilità, ovvero che trovi un modello.
Infatti, l'output di MACE (linea di comando `mace2.exe -p -n2 < studenti.ott`) comprende:

```
======================= Model #1 at 0.00 seconds:

 Studente :
        0 1
    -------
        T F

 AnnoNascita :
      | 0 1
    --+----
    0 | F T
    1 | F F

 IntPos :
        0 1
    -------
        F T
```