Sapienza Università di Roma

Basi di Dati, Modulo 2

Sapienza Università di Roma
Facoltà di Ing. dell'Informazione, Informatica e Statistica
Laurea in Informatica
Prof. Toni Mancini
http://tmancini.di.uniroma1.it

Documento A.4.1 (D.A.4.1)

Analisi dei requisiti

Logica del primo ordine (FOL) nell'Analisi Concettuale

Definire insiemi ordinati

Versione 2024-02-08

In alcuni casi, durante la fase di analisi concettuale è necessario definire un ordinamento totale tra gli elementi di un insieme, in quanto tale ordinamento è concettualmente rilevante e non può essere ignorato.

Ad esempio, si consideri il seguente frammento di specifica dei requisiti (dal progetto QuickHospital):

> Il sistema deve permettere di calcolare, su richiesta di un medico, il suo itineriario delle visite, ovvero un insieme ordinato delle stanze cui accedere (che sono tutte e sole le stanze che ospitano i pazienti che ha in cura).
>
> L'ordinamento è dato in primo luogo dal piano delle stanze dei pazienti da visitare, ed in secondo luogo dal settore di appartenenza di tali stanze (entrambi in ordine crescente). I settori sono infatti numerati secondo un criterio di vicinanza topologica. Pertanto se un dato medico deve visitare le stanze $\{(7,4),(7,1),(1,3),(1,1),(3,4)\}$ dove la prima componente di ognuna è il piano e la seconda il settore, l'itinerario di visita proposto deve essere $[(1,1),(1,3),(3,4),(7,1),(7,7)]$.

In casi come questo, l'operazione di use-case modellata dall'analista deve necessariamente restituire un insieme di oggetti (di classe Stanza) ordinato secondo un certo criterio, pena il non soddisfacimento dei requisiti.

In questo breve documento vedremo come le estensioni già applicate a FOL (in particolare teoria degli insiemi e aritmetica) di permettono di definire uno scheletro generale per imporre un ordinamento totale ad un insieme di oggetti di una qualche classe o valori di un qualche tipo di dato (ad es., composto).

Iniziamo quindi con il vedere come si può modellare una operazione che, dato in input un insieme di oggetti di una certa classe (o insieme di valori di un certo tipo di dato) T restituisca un *ordinamento totale* di tali elementi.

1 Segnatura dell'operazione

Per comprendere quale sia il tipo di ritorno di tale operazione, basta figurarsi la forma che ci attendiamo per il risultato.

Partiamo con un esempio: immaginiamo che l'insieme in input sia $\{a, b, c, d\}$ (tutti elementi di classe/tipo T). Vorremo in output un qualcosa che rappresenti, ad es: "d in posizione 1, b in posizione 2, a in posizione 3, c in posizione 4".

Tale forma per il risultato può essere modellata come istanza di una struttura dati che associ, ad ogni elemento dell'insieme dato, un intero da 1 in su, che ne indichi la posizione nell'ordinamento calcolato.

In altri termini, vogliamo in output un *insieme di coppie* del tipo:

$$(\text{elem} : T, \quad \text{pos} : \text{Intero} > 0).$$

Dunque, la specifica dell'operazione definirà la segnatura seguente:

Specifica Use-Case Mio use case

- sorted(elem : **T [0..*]**) : **(elem : T, pos : Intero > 0) [0..*]**

  precondizioni: ...

  postcondizioni: ...

dove abbiamo utilizzato la notazione (elem : T, pos : Intero > 0) per definire un tipo anonimo composto da due campi: il primo di nome elem di tipo T, ed il secondo di nome pos di tipo Intero > 0.

Alternativamente possiamo definire il tipo di ritorno nel documento di specifica dei tipi di dato come tipo composto nel modo usuale (dandogli un nome).

2 Precondizioni

Nel caso tipico in cui qualunque insieme di valori per l'argomento elem in input potrà essere ordinato in qualunque stato del livello estensionale, la specifica dell'operazione non avrà alcuna precondizione.

Altrimenti le precondizioni definiranno, al solito, le ulteriori condizioni necessarie, sui valori del parametro attuale (multivalore) elem e sul livello estensionale, affinché il raggiungimento delle post-condizioni sia garantito. Ad esempio, le precondizioni potrebbero imporre che tutti i valori del parametro attuale elem abbiano valori definiti per alcuni attributi/campi (opzionali nel diagramma delle classi o nella definizione del tipo T), senza i quali non possono essere ordinati.

3 Postcondizioni

Avremo bisogno di modellare il *criterio* che, dati in input due elementi a e b di classe/tipo T, definisca se a debba occorrere prima o al pari di b nell'ordinamento. Se T è un tipo di dato numerico base o specializzato (ad es., Intero, Reale, Intero ≤ 0), oppure un tipo di dato base naturalmente ordinato come Data, DataOra, Durata, basterà sfruttare il simbolo di predicato $\leq$, che, grazie alla semantica di mondo reale a cui è soggetto, si comporterà correttamente sui valori del tipo.

Nel caso più generale invece, dovremo modellare una operazione ausiliaria (che chiameremo $T_{\leq}()$) avente segnatura: $T_{\leq}(a : T,\ b : T)$: Booleano, la cui specifica sarà opportunamente definita affinché, su input (a, b), restituisca true se e solo se a deve occorrere prima o a pari merito di b nell'ordinamento cercato.

Osserviamo che, in generale, non è detto che l'ordinamento cercato sia unico. Ad esempio, è possibile che due elementi di T siano considerati a pari merito per il nostro criterio di ordinamento. Questo avverrà quando, per due elementi a e b di T, risulta sia $T_{\leq}(a, b) =$ true che $T_{\leq}(b, a) =$ true.

Forti della definizione dell'operazione ausiliaria $T_{\leq}()$, possiamo ora procedere a definire le postcondizioni dell'operazione sorted().

Queste constano di due passi:

1. Definire un insieme O ('ordinamenti') che contenga tutti e soli gli ordinamenti corretti dei valori in input (i valori dell'argomento multivalore elem:T [0..*] dell'operazione) secondo il criterio imposto da $T_{\leq}()$

2. Definire result come un elemento arbitrario di O.

Al solito, i valori attuali dell'argomento elem : T [0..*] dell'operazione formano un insieme. Chiamiamo tale insieme ins $\subseteq T$.

L'insieme O di tutti gli ordinamenti corretti di ins sarà dunque un insieme di insiemi, in quanto ogni elemento di O sarà a sua volta un insieme di coppie (elem : T, pos : Intero > 0).

$$O = \{\mathsf{ins}_{\mathsf{ord}} \mid \text{“}\mathsf{ins}_{\mathsf{ord}} \text{ è un ordinamento corretto di ins'}\} .$$

Veniamo ai requisiti che il generico elemento $\mathsf{ins}_{\mathsf{ord}}$ di O deve soddisfare (requisiti da definire a destra del 'tale che').

Ogni $\mathsf{ins}_{\mathsf{ord}}$ deve essere necessariamente un sottoinsieme (anche non proprio) di ins $\times$ $1..|\mathsf{ins}|$, ovvero dell'insieme delle coppie aventi un elemento di ins come primo elemento ed un intero tra 1 e |ins| (la cardinalità di ins) come secondo elemento:

$$\mathsf{ins}_{\mathsf{ord}} \subseteq \mathsf{ins} \times 1..|\mathsf{ins}|. \tag{3.1}$$

Ma questo non basta. Difatti, un insieme $\mathsf{ins}_{\mathsf{ord}}$ che soddisfa la condizione qui sopra potrebbe non definire ordinamento totale di ins. In particolare, potrebbe *non* menzionare un certo elemento di ins, potrebbe assegnare la *stessa* posizione nell'ordinamento a *due* elementi diversi di ins. Infine, anche se definisse un ordinamento totale di ins, potrebbe violare il criterio definito da $T_{\leq}()$.

Vanno quindi aggiunti i seguenti ulteriori vincoli su $\mathsf{ins}_{\mathsf{ord}}$:

1. Ogni elemento di ins deve comparire esattamente una volta in $\mathsf{ins}_{\mathsf{ord}}$. Un modo semplice di imporre questo vincolo è di scomporlo in due parti:

   1.1. Ogni elemento di ins compare almeno una volta in $\mathsf{ins}_{\mathsf{ord}}$:

   $$\forall o \;\; o \in S \;\rightarrow\; \exists i \; (o, i) \in \mathsf{ins}_{\mathsf{ord}}. \tag{3.2}$$

   1.2. Ogni elemento di ins compare al più una volta in $\mathsf{ins}_{\mathsf{ord}}$:

   $$\forall o \; o \in S \;\rightarrow\; \neg\exists i_1, i_2 \;\; i_1 \neq i_2 \wedge (o, i_1) \in \mathsf{ins}_{\mathsf{ord}} \wedge (o, i_2) \in \mathsf{ins}_{\mathsf{ord}} \tag{3.3}$$

2. Ogni indice tra 1 e |ins| compare esattamente una volta in $\mathsf{ins}_{\mathsf{ord}}$. Di nuovo, un modo semplice di imporre questo vincolo è di scomporlo in due parti:

   2.1. Ogni indice i compare almeno una volta in $\mathsf{ins}_{\mathsf{ord}}$:

   $$\forall i \; (\mathsf{Intero}(i) \wedge 1 \leq i \wedge i \leq |\mathsf{ins}|) \;\rightarrow\; \exists o \; (o, i) \in \mathsf{ins}_{\mathsf{ord}} \tag{3.4}$$

   2.2. Ogni indice i compare al più una volta in $\mathsf{ins}_{\mathsf{ord}}$. Questo vincolo è implicato dagli altri e può quindi essere omesso, grazie al fatto che la funzione matematica definita dal predicato $\mathsf{ins}_{\mathsf{ord}}$ ha dominio (ins) e codominio (interi tra 1 e |ins|) della stessa taglia.

Un insieme $\mathsf{ins}_{\mathsf{ord}}$ sottoinsieme (anche non proprio) di ins $\times$ $1..|\mathsf{ins}|$ che soddisfa i vincoli qui sopra definirà un ordinamento totale (o *permutazione*) di ins.

Quel che ci resta da imporre ad $\mathsf{ins}_{\mathsf{ord}}$ è che l'ordinamento rispetti il criterio definito dall'operazione ausiliaria $T_{\leq}()$:

3. $\mathsf{ins}_{\mathsf{ord}}$ definisce un ordinamento conforme a quanto imposto da $T_{\leq}()$:

$$\forall o_1, o_2, i_1, i_2 \;\; (o_1, i_1) \in \mathsf{ins}_{\mathsf{ord}} \wedge (o_2, i_2) \in \mathsf{ins}_{\mathsf{ord}} \wedge i_1 \leq i_2 \;\rightarrow\; T_{\leq}(o_1, o_2) = \mathsf{true} \tag{3.5}$$

4 Specifica completa

La specifica completa è dunque la seguente:

Specifica Use-Case Mio use case

- **sorted(elem : T [0..*]) : (elem : T, pos : Intero > 0) [0..*]**

  precondizioni: nessuna (o altro, in base al caso in esame)

  postcondizioni: Detto ins $\subseteq T$ l'insieme dei valori attuali dell'argomento elem, sia:

  $$O = \{\text{ins}_{\text{ord}} \mid \text{ins}_{\text{ord}} \subseteq \text{ins} \times 1..|\text{ins}| \wedge \text{da (3.2) a (3.5)}\}$$

  result è (un qualunque assegnamento) tale da soddisfare la formula:

  $$\text{result} \in O.$$

5 La famiglia di operazioni sorted$_{T,\ T_{\leq}}()$

Si noti come la specifica precedente dipenda esclusivamente dal tipo T degli elementi da ordinare e dall'operazione ausiliaria $T_{\leq}()$. Per il resto, sarà sempre la stessa e dunque ha poco senso doverla scrivere ogni volta. Possiamo inoltre fare sempre in modo che la specifica non abbia alcuna precondizione, delegando eventuali restrizioni sulle istanze del tipo T ordinabili alla operazione ausiliaria $T_{\leq}()$.

Arricchiamo dunque ulteriormente il nostro armamentario formale per l'analisi concettuale, definendo, una volta per tutte, una *famiglia di operazioni* (che chiamiamo sorted) che calcolino un ordinamento di un insieme di oggetti/valori dati come argomento, secondo un criterio specificato dall'analista.

Formalmente avremo una operazione della famiglia sorted *per ogni* classe/tipo di dato T degli elementi da ordinare, e *per ogni* operazione ausiliaria $T_{\leq}(a : T,\ b : T)$: Booleano che definisce l'ordinamento tra due elmenti di T (con eventuali precondizioni opportune che impediscano di invocarla con valori dei parametri attuali non ordinabili).

In altri termini, per ogni classe o tipo di dato T e per ogni operazione ausiliaria $T_{\leq}()$, l'operazione della famiglia sorted relativa a T e $T_{\leq}$ avrà nome e segnatura seguenti:

$$\text{sorted}_{T,\ T_{\leq}}(\text{elem} : T\ [0..*]) : (\text{elem} : T,\ \text{pos} : \text{Intero} > 0)\ [0..*].$$

Ovvero, l'operazione prenderà in input zero o più elementi di classe/tipo T e dipenderà dall'operazione ausiliaria $T_{\leq}(a : T,\ b : T)$: Booleano menzionata a pedice.

Anche questa estensione del nostro armamentario formale per l'analisi concettuale è in linea con quanto viene effettuato dai linguaggi di uso pratico per la modellazione di conoscenza (in primis OCL, il linguaggio per le specifiche formali parte dello standard UML).