3 Tipi di Dato Astratti e algebre many-sorted

Si suppongono note le seguenti nozioni:

- segnatura many-sorted Σ. Indichiamo con σ un simbolo di sort e con *op* un simbolo di operazione. Scriviamo $op\colon \sigma_{i\in n} \to \sigma$ per indicare l'arieta' di *op*.
- l'insieme (definito induttivamente) dei Σ-termini $t \in T_\Sigma(X)_\sigma$ di sort σ e con variabili in X, dove X e' una famiglia di insiemi indiciata dalle sort di Σ, quindi X_σ e' l'insieme delle variabili di sort σ.
- Σ-algebra (parziale) A. Indichiamo con σ^A l'insieme che *interpreta* la sort σ, e $op^A\colon \sigma_1^A \times \ldots \times \sigma_n^A \rightharpoonup \sigma^A$ la funzione (parziale) che interpreta l'operazione *op* (con l'arieta' sopra indicata). In genere, si suppone che gli insiemi σ^A non siano vuoti. Si dice che A e' una Σ-algebra totale, se tutte le operazioni sono interpretate da funzioni totali.
- l'interpretazione $t_v^A \in \sigma^A$ del termine t nell'algebra A in funzione di una interpretazione v delle variabili, cioe' $v(x) \in \sigma^A$ se $x \in X_\sigma$. Nel caso di un'algebra parziale t_v^A puo' essere indefinita.

Un tipo di dato astratto puo' essere specificato o in modo assiomatico, cioe' dando degli assiomi (p.e. equazioni o formule della logica del primo ordine) che descrivono proprieta' che devono essere *soddisfatte* dalle operazioni, oppure dando una interpretazione *canonica*. Nell'approccio assiomatico e' naturale identificare una *implementazione* del tipo di dato astratto con un *modello*, cioe' un'algebra che soddisfa gli assiomi. Nell'altro approccio non vi e' una cosi' chiara distinzione tra specifica ed implementazione. Quindi introdurremo una nozione di simuazione di un'algebra (specifica) tramite un'altra algebra (implementazione), simile a quella vista per gli automi.

Definition 3.1 (Simulazione tra algebre) *Date due Σ-algebre (parziali) A e B, una* **simulazione** *di A tramite B e' una famiglia di relazioni $R_\sigma \subseteq \sigma^B \times \sigma^A$ indiciata dalle sort di Σ t.c.*

- $(\forall i \in n. b_i\ R_{\sigma_i}\ a_i)$ *e* $op^A(a_{i\in n})$ *definita implicano* $op^B(b_{i\in n})$ *definita e* $op^B(b_{i\in n})\ R_\sigma\ op^A(a_{i\in n})$

per ogni operazione $op\colon \sigma_{i\in n} \to \sigma$.

In genere e' ragionevole assumere che R_σ sia il grafo di una funzione parziale surgettiva. Quindi ogni elemento di A e' *rappresentabile* da almeno un elemento di B, ed un elemento di B puo' rappresentare al piu' un elemento di A.

Proposition 3.2 (Simulazioni e correttezza totale) *Se R e' una simulazione di A tramite B, $t \in T_\Sigma(X)_\sigma$ e' un termine, u e v sono interpretazioni delle variabili in A e B rispettivamente, allora*

- $(\forall x \in X. u(x)\ R\ v(x))$ *e* t_u^A *definita implicano (*t_v^B *definita e)* $t_v^B\ R_\sigma\ t_u^A$.

In certi casi la nozione di simulazione e' troppo forte. Per esempio, non e' possibile simulare i numeri interi usando rappresentazioni binarie a 32 bit, poiche' le operazioni aritmetiche $+$ o $-$ sono totali sugli interi, ma parziali (causano errore di overflow) sulle rappresentazioni binarie di 32 bit. In questi casi si usa una nozione piu' debole di simulazione.

Definition 3.3 (Simulazione parziale) *Date due Σ-algebre (parziali) A e B, una* **parziale simulazione** *di A tramite B e' una famiglia di relazioni $R_\sigma \subseteq \sigma^B \times \sigma^A$ indiciata dalle sort di Σ t.c.*

- $(\forall i \in n. b_i\ R_{\sigma_i}\ a_i)$ *e* $op^A(a_{i\in n})$ *e* $op^B(b_{i\in n})$ *definite implicano* $op^B(b_{i\in n})\ R_\sigma\ op^A(a_{i\in n})$

per ogni operazione $op\colon \sigma_{i\in n} \to \sigma$.

Quindi la differenza cruciale rispetto alla definizione di simulazione e' data dal fatto che nell'implementazione B una operazione *op* puo' essere indefinita anche quando nella specifica A *op* e' definita.

Proposition 3.4 (Simulazioni parziali e correttezza parziale) *Se R e' una simulazione parziale di A tramite B, $t \in T_\Sigma(X)_\sigma$ e' un termine, u e v sono interpretazioni delle variabili in A e B rispettivamente, allora*

- $(\forall x \in X. u(x)\ R\ v(x))$ *e* t_u^A *e* t_v^B *definite implicano* $t_v^B\ R_\sigma\ t_u^A$.

Example 3.5 Si consideri la seguente segnatura Σ per l'ADT **imsiemi finiti**:

- sorts – $elem, set, bool$
- ops – $empty\colon set$, $\quad member\colon elem, set \to bool$, $\quad ins, del\colon elem, set \to set$, $\quad union, intersect\colon set, set \to set$.

- modello canonico: $elem^A = U$, $set^A = \mathcal{P}_{fin}(U)$ cioe' l'insieme dei sottoinsiemi finiti di U

Descriviamo delle Σ-algebre che corrispondono a diverse implementazioni, e diamo delle relazioni di simulazione che precisamo in che modo si *implementa* il modello canonico (solo la relazione R_{set} e' interessante, poiche' le altri sono la relazione identica):

- implementazione a bit vector (si richiede che U sia finito): $elem^B = U$, $set^B = U \to bool$ cioe' l'insieme degli array di booleani di dimensione U

  $a\ R_{set}\ X \stackrel{\Delta}{\iff} \forall x \in U.x \in X \leftrightarrow a(x) = true$
- implementazione a liste: $elem^B = U$, $set^B = U^*$ cioe' l'insieme delle sequenze finite di elementi di U

  $(x_i|i \in m)\ R_{set}\ X \stackrel{\Delta}{\iff} X$ e' l'insieme $\{x_i|i \in m\}$
- implementazione a liste senza ripetizioni: stessi carrier della precedente algebra, ma operazioni diverse

  $(x_i|i \in m)\ R_{set}\ X \stackrel{\Delta}{\iff} X$ e' l'insieme $\{x_i|i \in m\}$, e la lista non contiene ripetizioni, cioe' $\forall i, j \in m.x_i = x_j \supset i = j$
- implementazione a liste ordinate senza ripetizioni (si richiede un ordine lineare su U): stessi carrier della precedente algebra, ma operazioni diverse

  $(x_i|i \in m)\ R_{set}\ X \stackrel{\Delta}{\iff} X$ e' l'insieme $\{x_i|i \in m\}$, e la lista e' in ordine crescente, cioe' $\forall i, j \in m.i < j \supset x_i < x_j$
- implementazione (parziale) liste di lunghezza limitata e senza ripetizioni implementate con un array: $elem^B = U$, $set^B = [0..k] \times U^k$ cioe' l'insieme delle coppie lunghezza della lista e array di dimensione k

  $\langle m, (x_i|i \in k)\rangle\ R_{set}\ X \stackrel{\Delta}{\iff} 0 \leq m \leq k$, X e' l'insieme $\{x_i|i \in m\}$, e $\forall i, j \in m.x_i = x_j \supset i = j$ (si osservi che la solo i primi m elementi dell'array sono significativi).

Al fine di analizzare la complessita' di un algoritmo che usa un tipo di dato astratto, e' necessario fornire una maggiorazione della complessita' delle operazioni del tipo di dato. Tali maggiorazioni dovranno essere date in termini di parametri definibili a livello di modello canonico. E' opportuno evitare l'uso di parametri che sono specifici dell'implementazione, poiche' questo viola il principio di astrazione (cioe' l'utilizzatore del tipo di dato astratto e' costretto a conoscere i dettagli implementativi per capire cosa succede), e quindi impedisce di utilizzare un *approccio condizionale* all'analisi di complessita'. Per esempio, consideriamo l'operazione $del(x, X)$ di cancellazione dell'elemento x dall'insieme X, ed esemplifichiamo dei parametri appropriati (cioe' che si riferiscono al modello canonico) ed impropri (p.e. che fanno rifierimento all'implementazione a liste):

- parametri propri: la cardinalita' di X, la cardinalita' della sort *elem* (quando e' finita);
- parametri impropri: la lunghezza della lista X (nell'implementazione a liste con la possibilita' di ripetizione un insieme puo' essere rappresentato con liste di diversa lunghezza), la posizione di x nella lista X.