Strutturare il Controllo (della valutazione)

Sommario: 17 marzo, 2015

- Controllo della sequenza: Linguaggi Machine-Level, High-level e Valori modificabili
- Espressioni: Valutazione e Controllo della sequenza
- Comandi vs. Espressioni: Assegnamento
- Comandi: Controllo della sequenza
- Comandi Strutturati: goto e comandi per iterazione
- Programmazione Strutturata: Antesignana delle moderne Metodologie
- Ricorsione: Programmazione con induzione, Tail Recursion e Memoization

Controllo della sequenza: Linguaggi Machine-Level, e Valori Modificabili

- **Linguaggi Machine-Level**:
  - La struttura di un programma è una sequenza di statement atomici:

```
Mov L0, R0
Add R1, R0
Mov R1, L0
```

    - che usano Locazioni come Valori Modificabili
  - La sequenza di valutazione è facilmente controllata dal registro PC.
  - questa sequenza è determinante quando sono coinvolti Valori Modificabili

- Linguaggi High-Level:

Controllo della sequenza: Linguaggi High-level e Valori Modificabili

- Linguaggi Machine-Level:
- **Linguaggi High-Level**:
  - La struttura di un programma può non essere una seq.
  - Gli statements possono non essere atomici
    - ma possono usare Variabili come Valori Modif.
  - La sequenza di valutazione dipende dalla semantica.
- Semantica e Controllo di sequenza.

Osservazione

La Semantica ci dice:
Come stms non atomici sono decomposti in stm più semplici e in quale sequenza questi ultimi devono essere considerati e valutati

Controllo della sequenza: Espressioni

Espressioni sono presenti in tutti gli High-Level L.

- Semantica: In tutti i linguaggi, la sem. prevede che calcolino un valore, o risultino in una computazione indefinita (non-terminante).
- Struttura:
  - Un operatore (principale) ed $\mathtt{n} \geq 0$ argomenti (a loro volta espressioni)
  - Un'invocazione di funzione applicata a $\mathtt{n} \geq 0$ argomenti (a loro volta espressioni)
  - Un termine atomico (variabile, valore atomico)
- Controllo di sequenza: segue l'ordine e il metodo di valutazione degli argomenti
  - Ordine: leftmost, rightmost, associatività dx o sin.
  - Metodo:
    - Operatori: stretto, corto-circuito,
    - Invocazioni: trasmissione parametri

Espressioni: Ordine e Metodo di valutazione

Espressioni sono presenti in tutti gli High-Level L.

- **Ordine** di valutazione: *leftmost*, *rightmost*, *assoc. dx o sin.*
  - critico quando:
    - operatori su aritmetica finita (virgola mobile)
    - effetti laterali (modifica di variabili):
      Es. (x=3)+(y=x)
- **Metodo** di valutazione: *stretto*, *corto-circuito*
  - critico quando:
    - argomento indefinito.
      Es. Siano:
      - e valuta indefinito,
        $(\lambda\, \mathtt{x}.\lambda\, \mathtt{y}.\mathbf{if}\ (\mathtt{x} > 0)\ \mathbf{then}\ \mathtt{x}\ \mathbf{else}\ \mathtt{x} + \mathtt{y})15\ \mathtt{e}$
        può calcolare 15 se non stretta sul primo a.
      - $\mathtt{e}_1$ o $\mathtt{e}_2$v. indefinito e l'altra valuta **false**. true
        $\mathtt{e}_1$ **Or** $\mathtt{e}_2$
        puó valutare **false** se **Or** è valutata a corto-circuito

Assegnamento: Espressione o Comando

Assegnamento

- è un'espressione in alcuni linguaggi (C, Java).
- è un comando in altri (Pascal).
- Semantica:
  - in tutti i casi modica un valore modificabile
  - Linguaggi differenti possono usare differenti ordini di valutazione degli argomenti.
  - conducendo a sequenze di valutazioni compl. diverse e comp. compl. diversi.

```
b = 0;
a[f(3)] = a[f(3)] +1
```

per f così definito:

```
int f(int 0){
  if(b==0){b = 1; return 1;}
  else return 2;
}
```

  - usare ora ordine leftmost ora rightmost.

Assegnamento: l-value e r-value

Assegnamento:

- Gli argomenti di un assegnamento devono essere un valore modificabile (l-value) e un valore memorizzabile (r-value)
  - valore modificabile = Locazione di memoria
  - valore memorizzabile = valore assegnabile ad una locazione nella memoria
  - valore denotabile = valore associabile ad un identificatore nell'ambiente
  - valore esprimibile = valore che può essere calcolato con un'espressione (non singola variabile)
- Alcune espressioni possono essere valutate per ottenere un l-value oppure un r-value

```
a[f(3)] = a[f(3)] +1
```

Controllo della sequenza: Comandi

Comandi sono presenti in tutti i Linguaggi Prescrittivi (Macchina, Imperativi,...)

- Semantica: In tutti i linguaggi, la sem. prevede che modifichino la memoria (effetti laterali).
- Struttura. Varie classi di comandi:
  - Controllo di sequenza esplicito
  - Condizionali (o di selezione)
  - Iterativi

Comandi di Sequenza Esplicito

Introducono una sequenza di C, o alterano la sequenza corrente.

- Includono:
  - Comando Sequenza: $\texttt{C}_1\texttt{;}\,\texttt{C}_2$
  - Blocco inline
  - goto, break, continue, return.
- Comando Sequenza e Blocco inline sono presenti in tutti i Linguaggi con comandi ed hanno una semantica composizionale ottenuta dalla naturale composizione della semantica dei componenti.
- goto: Iterazione non strutturata
  - oscuro e poco espressivo
  - può sempre essere evitato a favore di iterazione strutturata
  - va evitato
- break, continue, return
  - Utiizzati in C per uscita non strutturata da blocchi.

Comandi di Condizionali

Discriminano tra due o più sequenze alternative

- Includono:
  - if-then-else e if-then
    - con struttura e comportamento analogo in tutti i linguaggi
  - case e switch
    - con struttura simile ma comportamento, a volte, diverso

      $\texttt{case 2 of 1 : C}_1\texttt{; 2 : C}_2\texttt{; 3 : C}_3\texttt{; 4 : C}_4\texttt{; end; C;}$

      $\texttt{switch (2)\{case 1 : C}_1\texttt{; case 2 : C}_2\texttt{; case 3 : C}_3\texttt{; case 4 : C}_4\texttt{; \}C;}$

      conducono alla sequenza:

      $\texttt{C}_2\texttt{; C;}$

      $\texttt{C}_2\texttt{; C}_3\texttt{; C}_4\texttt{; C;}$

      rispettivamente.

Comandi Iterativi (strutturati)

Incapsulano ed evidenziano la sequenza da iterare

- Si differenziano in questo dall'iterazione non strutturata basata sul goto
- Si suddividono in:
  - iterazione indeterminata: while, repeat, for (del C)
  - iterazione determinata: for del Pascal
- Iterazione indeterminata: **while** `E` **do** `C`
  - il numero di iterazioni può non essere determinato utilizzando lo stato iniziale e l'espressione di controllo `E`
  - l'iterazione può anche non terminare
  - `while (x>1 and x<10) do x = 5;`
- Iterazione determinata: **for** `i` $=$ $\mathtt{E}_{\mathtt{inizio}}$ **to** $\mathtt{E}_{\mathtt{fine}}$ **do** `C`
  - il numero di iterazioni è: $\mathtt{n} = \mathtt{v}_{\mathtt{E}_{\mathtt{fine}}} - \mathtt{v}_{\mathtt{E}_{\mathtt{inizio}}} + 1$