Breve introduzione alla Semantica Operazionale*

Quello che abbiamo visto può essere visto come un piccolo linguaggio **imperativo**, come quello che Winskel chiama **IMP**.

- Il comportamento di **IMP** viene descritto formalmente da un insieme di regole che specificano come valutare le espressioni e come eseguire i comandi.
- Le regole danno una *semantica operazionale* al linguaggio, operazionale proprio perché vicina all'implementazione del linguaggio. Inoltre offrono la base per semplici dimostrazioni di equivalenze tra comandi.
- La semantica operazionale, che fornisce un modello matematico, l'**astrazione**, dell'esecuzione di un interprete, viene usata per scrivere compilatori e interpreti ci descrive l'effetto di ogni comando sullo stato.

* Dal Capitolo 2 di "The Formal Semantics of Programming Languages: An Introduction" di Glynn Winskel. - Edizione italiana: "La semantica formale dei linguaggi di programmazione" di G. Winskel, F. Turini, P. Baldan e A. Bracciali

Categorie sintattiche

Cominciamo con l'introduzione delle categorie sintattiche del nostro linguaggio:

- numeri $\mathbf{N} \ni n$ (solo interi, per semplicità)
- valori di verità $\mathbf{T} = \{\mathbf{true}, \mathbf{false}\}$
- locazioni $\mathbf{Loc} \ni X$ (*variabili* che possono essere modificate)
- espressioni aritmetiche $\mathbf{Aexp} \ni a$
- espressioni booleane $\mathbf{Bexp} \ni b$
- istruzioni o comandi $\mathbf{Com} \ni c$
- Espressioni e istruzioni includono variabili, quindi il loro significato dipende dal contesto, dallo "stato" in cui vengono valutate.
- L'insieme degli *stati* Σ è dato dalle funzioni $\sigma : \mathbf{Loc} \to \mathbf{N}$, t.c. $\sigma(X)$ rappresenta il valore o il contenuto della locazione X nello stato σ.
- Adesso vedremo come costruire gli elementi di questi insiemi.

Categorie sintattiche, cont.

Per quanto riguarda la definizione delle ultime tre categorie sintattiche (**Aexp**,**Bexp**, e **Com**) daremo una *definizione induttiva*, ricorrendo ad apposite regole di formazione, con le quali esprimeremo cose del tipo seguente.

- Se a_0 e a_1 sono espressioni aritmetiche, lo è anche l'espressione composta $a_0 + a_1$.
- Analogamente, se b_0 e b_1 sono espressioni booleane, lo è anche l'espressione composta $b_0 \wedge b_1$.
- Analogamente, se c_0 e c_1 sono comandi, lo è anche il comando composto $c_0; c_1$.
- I simboli a_i, b_i e c_i sono usati per rappresentare una qualsiasi espressione aritmetica (espressione booleana, o comando). Più precisamente, si tratta di *metavariabili* che variano all'interno degli insiemi **Aexp**,**Bexp**, e **Com**.
- Diamo ora le regole di formazione delle espressioni (usando una notazione il più vicina possibile a quella già vista per il C)

Espressioni Aritmetiche

$$a ::= n|X|a_0 + a_1|a_0 - a_1|a_0 \times a_1$$

dove " :: =" deve essere letto come "può essere" e il simbolo "|" va inteso come "oppure".

- La regola $a ::= n$ ci dice che il numero $n \in \mathbf{N}$ è considerato un'espressione aritmetica (elementare).
- La regola $a ::= X$ ci dice che la variabile $X \in \mathbf{Loc}$ è considerata un'espressione aritmetica (elementare).
- La regola $a ::= a_0 + a_1$ ci dice che se a_0 e a_1 sono espressioni aritmetiche, allora lo è anche $a_0 + a_1$, dove le a_i sono meta-variabili. Ad es. se $a_0 = X$ and $a_1 = 5$ allora $X + 5$ è ancora un'espressione aritmetica.
- Le altre regole vanno interpretate in modo analogo.
- Per indicare che due espressioni sono equivalenti usiamo il simbolo $\equiv$.

Espressioni Booleane

$$b ::= \textbf{true} | \textbf{false} | a_0 == a_1 | a_0 <= a_1 | \neg b | b_0 \vee b_1 | b_0 \wedge b_1$$

- La regola $b ::= \textbf{true}$ ci dice che la costante **true** è considerata un'espressione booleana (elementare), analogamente per **false**.
- La regola $b ::= a_0 == a_1$ ci dice che se a_0 e a_1 sono espressioni aritmetiche, allora $a_0 == a_1$ è un'espressione booleana, analogamente per $a_0 <= a_1$.
- La regola $b ::= \neg b$ ci dice che se b è un'espressione booleana, lo è anche $\neg b$.
- La regola $b ::= b_0 \vee b_1$ ci dice che se b_0 e b_1 sono espressioni booleane, allora lo è anche $b_0 \vee b_1$
- Le altre regole vanno interpretate in modo analogo.
- Per indicare che due espressioni sono equivalenti usiamo il simbolo $\equiv$.

Comandi

$$c ::= \textbf{skip} | X = a | c_0; c_1 | \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 | \textbf{while } b \textbf{ do } c$$

- La regola $c ::= \textbf{skip}$ ci dice che **skip** è considerato un comando (elementare).
- La regola $c ::= X = a$ ci dice che se X è una variabile, ed a è un'espressione aritmetica, allora l'assegnamento $X = a$ è un comando.
- La regola $c ::= c_0; c_1$ ci dice che se c_0 e c_1 sono comandi, lo è anche il comando composto $c_0; c_1$.
- La regola $c ::= \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1$ ci dice che se b è un'espressione booleana e c_0 e c_1 sono comandi, lo è anche il comando composto $\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1$.
- La regola $c ::= \textbf{while } b \textbf{ do } c$ ci dice che se b è un'espressione booleana e c è un comando, lo è anche il comando composto $\textbf{while } b \textbf{ do } c$.
- Per indicare che due espressioni sono equivalenti usiamo il simbolo $\equiv$.

Valutazione delle Espressioni Aritmetiche

- Il significato di un'espressione **dipende** dal valore delle variabili, ovvero da quello che chiamiamo **stato**
- L'insieme degli stati Σ consiste nelle funzioni $\sigma : \mathbf{Loc} \to \mathbf{N}$ da locazioni a numeri, per cui $\sigma(X)$ rappresenta il valore associato a X, ovvero il contenuto della locazione X nello stato σ (nella nostra notazione corrisponde all'associazione $X \rightsquigarrow \sigma(X)$)
- La valutazione di un'espressione viene fatta dunque in un certo stato σ: abbiamo cioè *configurazioni* (coppie) della forma $\langle a, \sigma \rangle$ in attesa di essere valutate, arrivando al valore n.

$$\langle a, \sigma \rangle \to n$$

  Vedremo un insieme di regole da applicare per riscrivere le configurazioni fino a quando non raggiungiamo il valore finale n.
- La semantica operazionale si dice **strutturata**, quando usa la sintassi per guidare il processo di valutazione.

Regole di inferenza logica

- Regola di inferenza $\frac{\mathbf{A_1}, \ldots, \mathbf{A_n}}{\mathbf{B}}$: se sono vere le affermazioni sopra la riga (premesse), allora è vera l'affermazione sotto la riga (conclusione). Equivale a $A_1 \wedge \ldots \wedge A_n \Rightarrow B$
- $$\frac{\text{piove} \ \wedge \ \text{non ho l'ombrello}}{\text{mi bagno}}$$
- Un'inferenza è logicamente valida (le premesse implicano logicamente la conclusione) quando è impossibile che la conclusione sia falsa se le premesse sono vere. Altrimenti non è valida.
- $A \Rightarrow B$ è sempre vero, tranne nel caso in cui A è vero e B è falso.
- $A \Rightarrow B$ equivale a $\neg A \vee B$.

Regole di inferenza logica (cont.)

- Regola di inferenza $\frac{\mathbf{A_1}, ..., \mathbf{A_n}}{\mathbf{B}}$: se sono vere le affermazioni sopra la riga (premesse), allora è vera l'affermazione sotto la riga (conclusione). Equivale a $A_1 \wedge ... \wedge A_n \Rightarrow B$
- Assioma $\frac{}{\mathbf{B}}$: regola di inferenza senza premesse, B quindi la conclusione è sempre vera
- Una derivazione di B prende la forma di un albero (detto di dimostrazione o derivazione) che può essere una istanza di un assioma $\frac{}{\mathbf{B}}$ o includere le derivazioni delle premesse di B: $\frac{\frac{\mathbf{D_1}, ..., \mathbf{D_n}}{\mathbf{A_1}}, ..., \frac{\mathbf{E_1}, ..., \mathbf{E_n}}{\mathbf{A_n}}}{\mathbf{B}}$: la radice è l'asserzione da dimostrare, le foglie sono assiomi e i nodi intermedi sono ottenuti applicando regole di inf

Valutazione delle Espressioni Aritmetiche (cont.)

- La valutazione di un'espressione non richiede di modificare la memoria.
- $\langle n, \sigma \rangle \to n$ **assioma** (premessa vuota)
- $\langle X, \sigma \rangle \to \sigma(X)$
- $$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 + a_1, \sigma \rangle \to n_0 + n_1}$$ **premessa** / **conclusione**
- $$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 - a_1, \sigma \rangle \to n_0 - n_1}$$
- $$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \times a_1, \sigma \rangle \to n_0 \times n_1}$$
- N.B. Gli operatori in nero ($+$,$-$, $\times$) rappresentano simboli della sintassi, mentre i corrispondenti operatori in blu l'effettiva implementazione dell'operazione. Ad es. $n_0 + n_1$ sta per la somma di n_0 e di n_1.

Valutazione delle Espressioni Aritmetiche (esempio)

- Per valutare l'espressione aritmetica $a = (X + 5)$ nello stato σ_0, nel quale $\sigma_0(X) = 0$ si deve istanziare la regola della somma, cioè sostituire i valori concreti su cui fare la valutazione alle metavariabili della regola, X ad a_0, 5 ad a_1 e σ_0 a σ.

$$\frac{\langle \underline{a_0}, \underline{\sigma} \rangle \to n_0 \quad \langle \underline{a_1}, \underline{\sigma} \rangle \to n_1}{\langle \underline{a_0} + \underline{a_1}, \underline{\sigma} \rangle \to n_0 + n_1} \qquad \frac{\langle X, \sigma_0 \rangle \to n_0 \quad \langle 5, \sigma_0 \rangle \to n_1}{\langle X + 5, \sigma_0 \rangle \to n_0 + n_1}$$

- A questo punto il problema ci si riconduce alla valutazione delle sotto-espressioni $X \in \mathbf{Loc}$ e $5 \in \mathbf{N}$. Essendo espressioni aritmetiche semplici, ognuna ha il suo assioma da istanziare:

$$\langle X, \underline{\sigma} \rangle \to \underline{\sigma}(X) \quad \langle \underline{n}, \underline{\sigma} \rangle \to \underline{n} \quad \langle X, \sigma_0 \rangle \to \sigma_0(X) \quad \langle 5, \sigma_0 \rangle \to 5$$

- Adesso posso completare l'albero di derivazione

$$\frac{\dfrac{}{\langle X, \sigma_0 \rangle \to 0} \qquad \dfrac{}{\langle 5, \sigma_0 \rangle \to 5}}{\langle (X + 5), \sigma_0 \rangle \to 5}$$

Valutazione delle Espressioni Aritmetiche (esempio)

Dalle regole si ottiene un algoritmo per la valutazione di un'espressione. Consideriamo ad esempio la valutazione dell'espressione aritmetica più complessa $a_0 = (X + 5) + (7 + 9)$ sempre nello stato σ_0, dove $\sigma_0(X) = 0$ Procedendo analogamente otteniamo il seguente albero di derivazioni.

$$\dfrac{\dfrac{\dfrac{}{\langle X, \sigma_0\rangle \to 0} \quad \dfrac{}{\langle 5, \sigma_0\rangle \to 5}}{\langle (X + 5), \sigma_0\rangle \to 5} \quad \dfrac{\dfrac{}{\langle 7, \sigma_0\rangle \to 7} \quad \dfrac{}{\langle 9, \sigma_0\rangle \to 9}}{\langle (7 + 9), \sigma_0\rangle \to 16}}{\langle (X + 5) + (7 + 9), \sigma_0\rangle \to 21}$$

Valutazione delle Espressioni Booleane

- $\langle \textbf{true}, \sigma \rangle \to \textbf{true} \qquad \langle \textbf{false}, \sigma \rangle \to \textbf{false}$
- $$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 == a_1, \sigma \rangle \to \textbf{true}/\textbf{false}}$$ se $n_0 = n_1 / n_0 \neq n_1$
- $$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 <= a_1, \sigma \rangle \to \textbf{true}/\textbf{false}}$$ se $n_0 \leq n_1 / n_0 \not\leq n_1$
- $$\frac{\langle b, \sigma \rangle \to \textbf{true}}{\langle \neg b, \sigma \rangle \to \textbf{false}} \qquad \frac{\langle b, \sigma \rangle \to \textbf{false}}{\langle \neg b, \sigma \rangle \to \textbf{true}}$$
- $$\frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \vee b_1, \sigma \rangle \to t}$$ con t a seconda dei valori t_0 e t_1
- $$\frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \wedge b_1, \sigma \rangle \to t}$$ con t a seconda dei valori t_0 e t_1

Valutazione "lazy" di $\wedge$

- In C, la valutazione della congiunzione *and* è chiamata *lazy* o pigra, dato che inizialmente si rimanda la valutazione della seconda espressione e poi la si fa solo se necessario.
- In particolare, prima si controlla se la prima espressione è falsa, nel qual caso si evita di valutare la seconda parte dell'espressione, dato che la valutazione complessiva sarà comunque falsa.
- Questo tipo di valutazione può essere reso con le regole della semantica operazionale seguenti:

$$\frac{\langle b_0, \sigma\rangle \to \textbf{false}}{\langle b_0 \wedge b_1, \sigma\rangle \to \textbf{false}} \qquad \frac{\langle b_0, \sigma\rangle \to \textbf{true} \quad \langle b_1, \sigma\rangle \to t}{\langle b_0 \wedge b_1, \sigma\rangle \to t}$$

Esecuzione dei comandi

- L'esecuzione di un comando porta a un nuovo stato: $\langle c, \sigma \rangle \to \sigma'$, ammesso che l'esecuzione termini!
- Si suppone che lo *stato iniziale* σ_0 abbia la proprietà che $\sigma_0(X) = 0$ per tutte le locazioni X.
- L'ordine di valutazione è importante.
- $\langle \textbf{skip}, \sigma \rangle \to \sigma$
- $\dfrac{\langle a, \sigma \rangle \to m}{\langle X = a, \sigma \rangle \to \sigma[m/X]}$ $\sigma[m/X]$ è lo stato aggiornato*
- $\dfrac{\langle c_0, \sigma \rangle \to \sigma'' \quad \langle c_1, \sigma'' \rangle \to \sigma'}{\langle c_0; c_1, \sigma \rangle \to \sigma'}$ prima va eseguito il primo comando
- $\dfrac{\langle b, \sigma \rangle \to \textbf{true}/\textbf{false} \quad \langle c_i, \sigma \rangle \to \sigma_i}{\langle \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma \rangle \to \sigma_0/\sigma_1}$

$$^*\sigma[m/X](Y) = \begin{cases} m & \text{if } Y = X; \\ \sigma(Y) & \text{if } Y \neq X. \end{cases}$$

Modifiche dello stato

- L'insieme degli *stati* Σ è dato dalle funzioni σ : **Loc** $\rightarrow$ **N**, t.c. $\sigma(X)$ rappresenta il valore o il contenuto della locazione X nello stato σ.
- Quando si esegue un **assegnamento** $X = a$ nello stato σ si va a modificare lo stato limitatamente a X.
- Formalmente la modifica dello stato in solo punto si scrive: $\sigma[m/X]$, che indica che nello stato modificato il valore m sostituirà il valore precedente associato a X.
- I valori associati alle altre locazioni non cambiano: nel nuovo stato, otterrò il nuovo valore m se la locazione è X, i vecchi valori in tutti gli altri casi.

$$\sigma[m/X](Y) = \begin{cases} m & \text{if } Y = X; \\ \sigma(Y) & \text{if } Y \neq X. \end{cases}$$

- Se si esegue $X = 3$ in σ_0, ad es. si otterrà un nuovo stato σ_1 t.c. $\sigma_1(X) = 3$, mentre $\sigma_1(Y) = \sigma_0(Y) = 0$ per ogni $Y \neq X$.

Valutazione dei comandi (cont.)

- $$\frac{\langle b, \sigma\rangle \to \textbf{false}}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma}$$
- $$\frac{\langle b, \sigma\rangle \to \textbf{true}\ \langle c; \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma'}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma'}$$

Completezza e Correttezza

- Vorremmo che la nostra semantica ci permettesse di dimostrare tutto ciò che è vero e niente di falso. La semantica è
  - **completa** se può dimostrare ogni *asserzione* vera. Ad es., con la regola seguente, *non* potrei dimostrare che $\langle 5-3,\sigma\rangle \to 2$ (vero).

$$\frac{\langle a_0,\sigma\rangle \to n_0 \quad \langle a_1,\sigma\rangle \to 0}{\langle a_0 - a_1,\sigma\rangle \to n_0}$$

  - **corretta** se ogni *asserzione* dimostrabile è vera. Ad es., con la regola seguente, *potrei* dimostrare che $\langle 5-3,\sigma\rangle \to 7$ (falso)

$$\frac{\langle a_0,\sigma\rangle \to n_0 \quad \langle a_1,\sigma\rangle \to n_1}{\langle a_0 - a_1,\sigma\rangle \to n_0 + 2}$$

Equivalenze

- Due espressioni aritmetiche a_0 e a_1 sono *equivalenti* $a_0 \sim a_1$ se e solo se

$$\forall n \in \mathbf{N}, \forall \sigma \in \Sigma. \langle a_0, \sigma \rangle \to n \Leftrightarrow \langle a_1, \sigma \rangle \to n$$

- Due espressioni booleane b_0 e b_1 sono *equivalenti* $b_0 \sim b_1$ se e solo se

$$\forall t, \forall \sigma \in \Sigma. \langle b_0, \sigma \rangle \to t \Leftrightarrow \langle b_1, \sigma \rangle \to t$$

- Due comandi c_0 e c_1 sono *equivalenti* $c_0 \sim c_1$ se e solo se

$$\forall \sigma, \sigma' \in \Sigma. \langle c_0, \sigma \rangle \to \sigma' \Leftrightarrow \langle c_1, \sigma \rangle \to \sigma'$$

Es. di dimostrazione sui comandi

Proposizione
Sia $w \equiv$ **while** b **do** c. Allora

$$w \sim \textbf{if } b \textbf{ then } c; w \textbf{ else skip}$$

Dimostrazione
Si deve dimostrare che per ogni scelta di σ, σ':

$$\langle w, \sigma \rangle \to \sigma' \text{ sse } \langle \textbf{if } b \textbf{ then } c; w \textbf{ else skip}, \sigma \rangle \to \sigma'$$

Lo si fa usando le derivazioni delle regole