Appunti di Linguaggi di Programmazione

Lorenzo D'Antoni

30 maggio 2021

Indice

1 Introduzione

Per definire un linguaggio di programmazione è necessario darne:

- la sintassi (cioè una specifica dell'insieme dei programmi)
- la semantica (cioè una descrizione dell'effetto dell'esecuzione dei programmi)

Nell'implementazione di un linguaggio, è possibile distinguere diverse componenti. Il parser o analizzatore sintattico riceve in input una stringa di simboli e stabilisce se questa è in accordo con la sintassi, cioè verifica le regole della grammatica data. In caso positivo, il parser provvede anche a convertire la stringa di simboli in una rappresentazione interna più comoda per le fasi successive. Il type-checker o analizzatore statico riceve in input un programma e stabilisce se questo è un programma corretto, cioè verifica i vincoli contestuali dati. Infine l'interprete riceve in input un programma (la cui correttezza è stata provata nella fase precedente) e ne simula l'esecuzione.

2 Stringhe e linguaggi

Un programma in un linguaggio di programmazione può essere visto come una stringa di simboli costruita in modo da soddisfare certe regole (la sintassi del linguaggio).

Per illustrare il modo in cui viene in genere descritta formalmente la sintassi, occorre anzitutto precisare cosa si intende per simbolo, stringa e linguaggio.

Definizione 1 (Alfabeto)**.** Un alfabeto è un insieme finito non vuoto di oggetti detti simboli.

Definizione 2 (Stringa)**.** Una stringa u su un alfabeto A è una funzione (totale) da $[1, n]$ in A, per qualche $n \in \mathbb{N}$; n si dice lunghezza di u, e si indica con $|u|$.

L'unica stringa u tale che $|u| = 0$ si chiama stringa vuota e si indica con Λ; l'insieme delle stringhe su A si indica con A^* e l'insieme delle stringhe non vuote su A si indica con A^+. Indicando con A^n, per $n \in \mathbb{N}$, l'insieme delle funzioni da $[1, n]$ in A, si ha dunque per definizione che $A^+ = \cup_{n>0} A^n$.

Definizione 3 (Linguaggio)**.** Un linguaggio su un alfabeto A è un insieme di stringhe su A.

Si noti che A^+ è sempre un insieme infinito; più precisamente è un insieme numerabile.

In genere, scriviamo le stringhe utilizzando la rappresentazione per giustapposizione, cioè semplicemente scrivendo i simboli uno dopo l'altro da sinistra a destra. Per esempio se $A_{it} = a, b, c, d, \ldots, z$ è l'alfabeto italiano, una stringa è la parola italiana *amica*. Tuttavia, l'unico modo rigoroso di definire tale stringa è di dire che essa è la funzione $v\colon [1,5] \to A_{it}$ definita da:

$$v(i) = \begin{cases} a & \text{se } i = 1 \\ m & \text{se } i = 2 \\ i & \text{se } i = 3 \\ c & \text{se } i = 4 \\ a & \text{se } i = 5 \end{cases}$$

Infatti, la rappresentazione per giustapposizione (oltre che arbitraria e legata a ragioni culturali: altri popoli utilizzano per esempio la rappresentazione da destra a sinistra), può risultare ambigua per certi alfabeti.

Ad esempio con questa rappresentazione la stringa sull'alfabeto D_{it} formata dalle due parole *arco* e *baleno* è indistinguibile da quella formata dall'unica parola *arcobaleno*. In casi di questo genere si ricorre tipicamente all'uso di *separatori*, cioè simboli che non fanno parte dell'alfabeto ma servono solo come convenzione per delimitare i simboli dell'alfabeto, come il *blank*.

Un'altra rappresentazione spesso usata è quella a n-uple, cioè un elemento $w \in A^n$ definito da

$$w(i) = \begin{cases} a_1 & \text{se } i = 1 \\ a_2 & \text{se } i = 2 \\ \vdots & \\ a_n & \text{se } i = n \end{cases}$$

è scritto come $(a_1, \ldots, a_n)$. La stringa precedente v in questo modo viene scritta (a, m, i, c, a). Anche questa rappresentazione può creare ambiguità, nel caso in cui i separatori parentesi aperta, chiusa e virgola facciano parte dell'alfabeto.

L'unico modo rigoroso e indipendente dalla rappresentazione di definire le stringhe su un alfabeto è quello dato nella Def. 2, ed è conveniente richiamarsi a questa definizione tutte le volte che sorge qualche problema di ambiguità.

Definizione 4 (Concatenazione di stringhe e linguaggi)**.** Se v e w sono due stringhe su un alfabeto A di lunghezza n e m rispettivamente, allora $v \cdot w$ è

la stringa su A, di lunghezza $n + m$, definita da

$$(v \cdot w)(k) = \begin{cases} v(k) & \text{se } 1 \leq k \leq n \\ w(k-n) & \text{se } n < k \leq n+m \end{cases}$$

Inoltre $v^0 = \Lambda$, $v^{n+1} = v \cdot v^n$.

Se X, Y sono insiemi di stringhe su A, $X \cdot Y = \{x \cdot y \mid x \in X, y \in Y\}$

Utilizzando l'ultima definizione, si può dare la seguente definizione induttiva di A^n, per A alfabeto e $n \in \mathbb{N}$:

$$X^0 = \{\Lambda\}$$
$$X^{n+1} = X \cdot X^n$$

Un altro esempio "classico" di stringhe è dato dai numeri (interi positivi) nella rappresentazione in base 10, che sono esattamente le stringhe su $Digit = \{0, 1, \ldots, 9\}$, se si accetta 0 in posizione non significativa (es: 002); altrimenti sono un sottoinsieme proprio Num di queste stringhe, descritto da

$$Num = \{v \mid v(1) \neq 0 \text{ oppure } |v| = 1\}.$$

Gli elementi di Num vengono chiamati *numerali*.

Analogamente le espressioni costruite a partire dai numerali con le operazioni $+$ e $*$ e le parentesi tonde (e) sono un linguaggio sull'alfabeto $Digit \cup \{+, *, (,)\}$. Alternativamente possono essere viste come un linguaggio su $Num \cup \{+, *, (,)\}$, ma in questo caso l'alfabeto non è più finito.

3 Grammatiche context-free

Il modo usuale in cui si trova descritta formalmente la sintassi di un linguaggio è tramite una grammatica libera da contesto. Tale metodo deriva dagli studi di Noam Chomsky negli anni cinquanta sulle grammatiche per i linguaggi naturali. La Backus & Naur Form (BNF), o Backus Normal Form è un particolare stile di rappresentazione di una grammatica libera da contesto, e fu introdotta alla fine degli anni cinquanta per descrivere la sintassi dell'Algol 60.

Definizione 5 (Grammatica)**.** Una grammatica libera da contesto è una tripla (T, N, P) dove:

- T è un alfabeto di simboli, detti terminali,
- N è un alfabeto di simboli (diversi da quelli di T), detti non terminali,

- P è un insieme finito di coppie dette produzioni (A, α) con $A \in N$ e $\alpha \in (T \cup N)^*$ (cioè, A è un simbolo non terminale e α è una stringa formata da simboli terminali e non terminali). Nello stile BNF le produzioni sono scritte nella forma $A ::= \alpha$.

Useremo u, v, w per indicare generiche stringhe di terminali e α, β, γ per indicare generiche stringhe di terminali e non terminali. I terminali sono i simboli del linguaggio che si vuole definire tramite la grammatica, mentre i non terminali indicano i possibili "tipi" di oggetti sintattici.

Nello stile BNF, si usa l'abbreviazione $A ::= \alpha \mid \beta \mid \gamma \ldots$ per indicare le produzioni $A ::= \alpha$, $A ::= \beta$, $A ::= \gamma \ldots$

Nel seguente esempio vengono definite espressioni intere e booleane utilizzando due diversi non terminali.

$$
\begin{aligned}
Exp &::= (Exp + Exp) \mid (Exp * Exp) \mid Num \\
Num &::= 0 \mid 1 \mid \ldots \\
BExp &::= \text{true} \mid \text{false} \mid (BExp \ \text{or} \ Bexp) \mid (BExp \ \text{and} \ BExp) \mid (Exp = Exp)
\end{aligned}
$$

Per capire in che senso una grammatica definisca un linguaggio di programmazione, introduciamo il concetto di derivazione.

Definizione 6 (Derivazione in un passo)**.** Sia data una grammatica $G = (T, N, P)$ e due stringhe $\beta, \gamma \in (T \cup N)^*$. Diremo che γ è derivabile da β in un passo se e solo se esistono delle stringhe $\alpha_1, \alpha_2 \in (T \cup N)^*$, ed esiste in P una produzione $A ::= \alpha$, tali che $\beta = \alpha_1 A \alpha_2$ e $\gamma = \alpha_1 \alpha \alpha_2$. Diremo anche che $\beta \to \gamma$ è una derivazione (in un passo).

È chiario che $\to$ è una relazione su $(T \cup N)^*$. Indicheremo con $\to^*$ la sua chiusura riflessiva e transitiva, e diremo che γ è derivabile da β se vale che $\beta \to^* \gamma$. Diremo anche che $\beta \to^* \gamma$ è una derivazione.

Il linguaggio generato da una grammatica G, rispetto a un suo non terminale A, è l'insieme delle stringhe composte solo di terminali che si possono derivare da A. Formalmente:

Definizione 7 (Linguaggio generato da una grammatica)**.** Sia data una grammatica $G = (T, N, P)$ e un non terminale $A \in N$. Il linguaggio generato da G, rispetto ad A, è l'insieme:

$$L_A(G) = \{w \in T^* \mid A \to^* w\}.$$

Con l'abuso di terminologia già menzionato, si parla del linguaggio generato da G anche intendendo la famiglia $\{L_A(G)\}_{A \in N}$. Per semplicità, si usa spesso lo stesso nome per denotare una categoria sintattica e il linguaggio generato

corrispondente. Occorre però aver presente che sono due concetti distinti: il primo è semplicemente un simbolo, il secondo è un insieme di stringhe (di simboli terminali).

Consideriamo per esempio le seguenti grammatiche $G1$, $G2$, $G3$, $G4$.

$$
\begin{aligned}
&S ::= \mathrm{a}S \mid S\mathrm{b} \mid \mathrm{c}\\
&S ::= \mathrm{a}S\mathrm{b} \mid \Lambda\\
&S ::= \mathrm{a}S\mathrm{b} \mid \mathrm{ab}\\
&S ::= \mathrm{a}S\mathrm{a} \mid \mathrm{b}S\mathrm{b} \mid \Lambda \mid \mathrm{a} \mid \mathrm{b}
\end{aligned}
$$

I linguaggi generati da queste grammatiche sono rispettivamente:

$$
\begin{aligned}
&L(G1) = \{a^n c\, b^m \mid n, m \geq 0\}\\
&L(G2) = \{a^n b^n \mid n \geq 0\}\\
&L(G3) = \{a^n b^n \mid n \geq 1\}\\
&L(G4) = \{u \mid u \in \{a, b\}^*, u \,\text{palindroma}\}.
\end{aligned}
$$