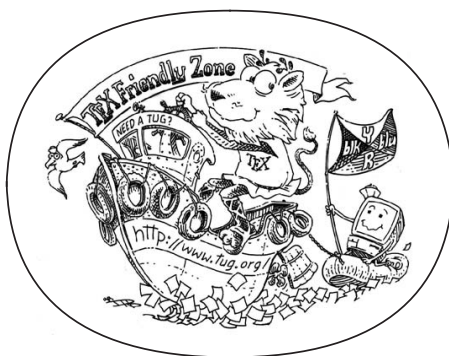


LINGUAGGI E PROGRAMMAZIONE ORIENTATA AGLI OGGETTI

RICCARDO CEREGHINO



Appunti

Settembre 2019 – classicthesis v4.6

INTRODUZIONE AGLI ELEMENTI DI UN LINGUAGGIO DI PROGRAMMAZIONE

I motivi della creazione ed utilizzo di un linguaggio di programmazione di alto livello sono: di fornire una descrizione precisa, ovvero una specifica formale; offrire un'interpretazione tramite interprete da compilare.

Le caratteristiche principali che categorizzano i linguaggi di programmazione sono la sintassi e la semantica, la quale può essere statica o dinamica.

1.1 LINGUAGGI STATICAMENTE TIPATI

Nei linguaggi tipizzati staticamente il *tipo* di variabile viene stabilito nel codice sorgente, per cui si rende necessario che:

- gli operatori e le assegnazioni devono essere usati coerentemente con il *tipo* dichiarato;
- le variabili siano usate consistentemente rispetto la loro dichiarazione.

I vantaggi della staticità risiedono nella preventiva rilevazione degli errori e nell'efficienza di calcolo risultante dalla coerenza tra codice e compilatore.

1.2 LINGUAGGI DINAMICAMENTE TIPATI

Nei linguaggi di programmazione dinamicamente tipati le variabili sono assegnate ai *tipi* durante l'esecuzione del programma, ne consegue che:

- la semantica statica non è definita;
- un utilizzo inconsistente di variabili, operazioni o assegnazioni generano errori dinamici basati sui tipi.

I linguaggi dinamici per questi motivi risultano essere più semplici ed espressivi.

1.2.1 Esempi di errori

Listing 1.1: Errore di sintassi

```
|| x = ;
```

Un errore sintattico generico a molti linguaggi è un'espressione formattata erroneamente.

Listing 1.2: Errore statico

```
int x=0;  
if(y<0) x=3; else x="three"
```

In un linguaggio statico come *Java* l'esempio precedente darebbe un errore in quanto una stringa non può essere convertita in un tipo intero.

Listing 1.3: Errore Dinamico

```
x = null;  
if(y<0) y=1; else y=x.value;
```

L'esempio, per $y > 0$, darebbe un errore dinamico sia in *Java*, che in un linguaggio dinamico come *Javascript*.

Parte I

SINTASSI

STRINGHE

Definizione 1 Un alfabeto A è un insieme finito non vuoto di simboli.

Definizione 2 Sia una stringa in un alfabeto A la successione di simboli in u :

$$u : [1 \dots n] \rightarrow A$$

Sia:

- $[1 \dots n] = m$, l'intervallo dei numeri naturali tale che:

$$1 \leq m \leq n;$$

- u sia una funzione totale;
- n sia la lunghezza di $u : \text{length}(u) = n$.

Definizione 3 Un programma è una stringa in un alfabeto A .

2.1 ESEMPIO DI STRINGHE

2.1.1 Stringa vuota

$$u : [1 \dots 0] \rightarrow A$$

Esiste un'unica funzione $u : 0 \rightarrow A$

Le notazioni standard di una stringa vuota sono: ε, λ

2.1.2 Stringa non vuota

Si consideri $A = \{ 'a', \dots, 'z' \} \cup \{ 'A', \dots, 'Z' \}$, l'alfabeto inglese di lettere minuscole e maiuscole. La funzione $u : [1 \dots 4] \rightarrow A$ rappresenta la stringa "Word" con:

- $u(1) = 'W'$
- $u(2) = 'o'$
- $u(3) = 'r'$
- $u(4) = 'd'$

2.1.3 Concatenazione di stringhe

Definizione 4

$$\text{length}(u \cdot v) = \text{length}(u) + \text{length}(v)$$

Per ogni $i \in [1 \dots \text{length}(u) + \text{length}(v)]$

$$(u \cdot v)(i) = \text{if } i \leq \text{length}(u) \text{ then } u(i) \text{ else } v(i - \text{length}(u))$$

Monoide

La concatenazione è associativa, ma non commutativa.

La stringa vuota è l'identità dell'elemento.

Induzione

La definizione di u^n per induzione su $n \in \mathbb{N}$:

Base: $u^0 = \lambda$

Passo induttivo: $u^{n+1} = u \cdot u^n$ Per cui u^n si concatena con se stesso n volte.

2.1.4 Insiemi di stringhe

Definizione 5 Sia A un alfabeto:

- A^n = l'insieme di tutte le stringhe in A con lunghezza n ;
- A^+ = l'insieme di tutte le stringhe in A con lunghezza maggiore di 0;
- A^* = l'insieme di tutte le stringhe in A ;
- $A^+ = \bigcup_{n>0} A^n$;
- $A^* = \bigcup_{n \geq 0} A^n = A^0 \cup A^+$

2.2 LINGUAGGIO FORMALE

Definizione 6 (Nozione sintattica di linguaggio) Un linguaggio L in un alfabeto A è un sottoinsieme di A^*

ESEMPIO: L'insieme L_{id} di tutti gli identificatori di variabile:

$$A = \{ 'a', \dots, 'z' \} \cup \{ 'A', \dots, 'Z' \} \cup \{ '0', \dots, '9' \}$$

$$L_{\text{id}} = \{ 'a', 'b', \dots, 'a0', 'a1', \dots \}$$

2.2.1 Composizione di operatori tra linguaggi

Le operazioni possono essere di concatenazione o di unione:

- **Concatenazione:** $L_1 \cdot L_2 = \{ u \cdot w \mid u \in L_1, w \in L_2 \}$;
- **Unione:** $L_1 \cup L_2$.

2.2.2 Intuizione

Unione

$L = L_1 \cup L_2$: qualsiasi stringa L è una stringa di L_1 o di L_2 .

ESEMPIO:

$$L' = \{ 'a', \dots, 'z' \} \cup \{ 'A', \dots, 'Z' \}$$

Concatenazione

$L = L_1 \cdot L_2$: qualsiasi stringa L è una stringa di L_1 , seguita da una stringa di L_2 .

ESEMPIO:

$$\{ 'a', 'ab' \} \cdot \{ \lambda, '1' \} = \{ 'a', 'ab', 'a1', 'ab1' \}$$

$$L_{\text{id}} = L' \cdot A^* \text{ con } A = \{ 'a', \dots, 'z' \} \cup \{ 'A', \dots, 'Z' \} \cup \{ '0', \dots, '9' \}$$

2.2.3 Monoide

La concatenazione è associativa, ma non commutativa.

$A^0 (= \{ \lambda \})$ è l'identità dell'elemento; quindi A^0 non è l'elemento neutro, l'elemento neutro è $0 = \{ \}$.

2.2.4 Passo induttivo

L^n è definito per induzione su $n \in \mathbb{N}$: Base: $L^0 = A^0 (= \{ \lambda \})$,

Passo induttivo: $L^{n+1} = L \cdot L^n$.

2.2.5 Operatori + e *

- **Addizione:** $L^+ = \bigcup_{n>0} L^n$;
- **Moltiplicazione:** \star viene chiamata *Kleen star*, *stella di Kleen*.

$$L^* = \bigcup_{n \geq 0} L^n$$

Sono equivalenti $L^* = L^0 \cup L^+, L \cdot L^*$.

Intuizione

- Qualsiasi stringa di L^+ è ottenuta concatenando una o più stringhe di L ;
- Qualsiasi stringa di L^* è ottenuta concatenando 0 o più stringhe di L : *Concatenando zero stringhe si ottiene la stringa vuota.*

ESPRESSIONI REGOLARI

Le espressioni regolari sono un formalismo comunemente utilizzato per definire linguaggi semplici.

Definizione 7 *La definizione induttiva di un espressione regolare su un alfabeto A :*

BASE:

- 0 è un espressione regolare di A ;
- λ è un espressione regolare di A ;
- per ogni $\sigma \in A$, σ è un espressione regolare in A .

PASSO INDUTTIVO:

- se e_1 ed e_2 sono espressioni regolare di A ,
allora $e_1|e_2$ è un espressione regolare di A ;
- se e_1 ed e_2 sono espressioni regolare di A ,
allora e_1e_2 è un espressione regolare di A ;
- se e è un espressione regolare di A ,
allora e^* è un espressione regolarare di A .