Corso di Laurea Ingegneria Informatica

Fondamenti di Informatica

Dispensa 08

Linguaggi – Sintassi e Semantica

Alfonso Miola

Ottobre 2011

Contenuti

- Definizione di un linguaggio
- Sintassi e semantica
- Linguaggi di programmazione e grammatiche
- Meta-linguaggio BNF
- Sintassi dei linguaggi di programmazione
- Sintassi del linguaggio Java
- Semantica del linguaggio Java
- Sintassi, semantica ed errori

Prerequisiti

- Questo capitolo presuppone la conoscenza degli argomenti già trattati nelle precedenti lezioni di questo corso, con riferimento al capitolo 4 del libro di testo e in particolare alla
  - Compilazione di programmi
- Si presuppone anche la conoscenza degli argomenti già trattati nel corso di Geometria e Combinatoria

Linguaggi naturali . . .

- Per definire un linguaggio naturale si parte dalla definizione di un alfabeto
  - in italiano ci sono 21 lettere, in inglese 26, . . .
- Con i caratteri dell'alfabeto possiamo formare un insieme di sequenze, dette parole
- Non tutte le sequenze sono parole del linguaggio naturale
- La grammatica del linguaggio fornisce le regole per decidere quali sequenze sono parole del linguaggio
  - parole corrette grammaticalmente

. . . Linguaggi naturali . . .

- **Con le lettere dell'alfabeto italiano possiamo costruire alcune sequenze**
  - **ad esempio abcdef, ghil, rst - che non sono parole della lingua italiano**
  - **ad esempio andare, aula, corso, acqua, soqquadro, - che sono parole della lingua italiana, cioè sono parole corrette grammaticalmente**

. . . Linguaggi naturali . . .

- Con le parole, corrette, possiamo formare sequenze di parole, dette frasi
- Non tutte le sequenze di parole sono frasi del linguaggio naturale
- La sintassi del linguaggio fornisce le regole per decidere quali sequenze sono frasi del linguaggio
  - frasi corrette sintatticamente, o sintatticamente ben formate

. . . Linguaggi naturali . . .

- **In italiano la regola base della sintassi dice che le frasi sono costruire con sequenze di parole che seguono la struttura**

  soggetto *verbo* *complemento*

  - ***soggetto*, *verbo* e *complemento* non sono altro che dei nomi, cioè denotano, alcuni particolari e ben precisi sottoinsiemi dell'insieme di tutte le parole del linguaggio**
  - **ad esempio la sequenza di parole il lo la non è quindi una frase della lingua italiana**
  - **ad esempio la sequenza di parole gatto mangia topo è una frase della lingua italiana, ovvero è sintatticamente ben formata**

. . . Linguaggi naturali

- Solo alcune delle frasi del linguaggio, cioè di quelle ben formate sono anche valide, cioè hanno un significato
- La semantica del linguaggio stabilisce quali tra le frasi ben formate sono anche valide e quindi si occupa dell'interpretazione (del significato) delle frasi
  - ad esempio la frase il gatto mangia il topo è una frase ben formata che è anche valida, cioè ha un significato

Sintassi

- La sintassi di un linguaggio si occupa della forma delle frasi del linguaggio, ovvero delle regole che permettono di costruire frasi ben formate del linguaggio
- Esempio di frase in italiano

  il gatto mangia il topo
- Frammento della sintassi della lingua italiana

  frase → *soggetto verbo complemento*
  soggetto → *articolo nome*
  verbo → mangia, beve
  complemento → *articolo nome*
  articolo → il, lo, la
  nome → gatto, monte, topo, carne

Semantica

- La semantica di un linguaggio si occupa dell'interpretazione del linguaggio, ovvero del significato delle frasi corrette sintatticamente
- Esempio di frasi corrette sintatticamente in italiano, ma non tutte valide – in rosso le frasi valide
  - il gatto mangia il topo
  - il topo mangia il monte
  - il cane mangia la carne
  - il monte beve il cane

Regole sintattiche . . .

- Le regole della sintassi sono chiamate regole di produzione, per produrre o derivare una frase, come nell'esempio precedente
- Nelle regole di produzione compaiono
  - elementi (simboli) terminali
    - come - mangia, beve, il, lo, la, gatto, monte, topo, carne
  - elementi (simboli) non-terminali
    - come – *frase, soggetto, verbo, ….*
    - che sono "categorie sintattiche" cioè nomi che denotano sottoinsiemi dell'insieme dei simboli terminali

. . . Regole sintattiche

frase → *soggetto verbo complemento*

soggetto → *articolo nome*

verbo → **mangia**, **beve**

complemento → *articolo nome*

articolo → **il**, **lo**, **la**

nome → **gatto**, **monte**, **topo**, **carne**

- **Una fissata categoria sintattica, detta assioma, è quella dalla quale deve partire il processo di produzione o di derivazione di una frase**
- **Nel caso della lingua italiana l'assioma è *frase***

Linguaggi artificiali e grammatiche

- **Un linguaggio di programmazione è un linguaggio artificiale e, per poterlo definire in modo rigoroso, introduciamo di seguito alcuni strumenti necessari, con le relative definizioni**
  - **Alfabeto, o vocabolario**
  - **Universo linguistico**
  - **Grammatica, o sintassi**
  - **Generazione di un linguaggio da una grammatica**

Universo linguistico

- ***Definizione*: Dato un insieme finito non vuoto V, si definisce Universo linguistico su V, e si indica con V*, l'insieme delle sequenze finite di lunghezza arbitraria di elementi di V**
  - **L'insieme V viene di solito chiamato alfabeto, oppure vocabolario o lessico. Gli elementi di V sono chiamati simboli terminali.**
  - **Si noti che talvolta i simboli di V possono essere più complessi di una singola lettera dell'alfabeto della lingua italiana; per esempio 'main', 'class', 'void', ecc. sono simboli dell'alfabeto di Java. Gli elementi di V* vengono detti stringhe costruite su V, o frasi su V**

Linguaggio

- ***Definizione*: Un linguaggio L sull'alfabeto V è un sottoinsieme di V*.**
  - Sebbene V sia finito, V* non lo è; esso è numerabile, ed i sottoinsiemi di V* sono in quantità non numerabile
  - Nel considerare i linguaggi di programmazione, non siamo interessati a tutti i sottoinsiemi di V*, ma solo a quelli che sono descrivibili in maniera finita
  - Questa descrizione può essere per esempio fornita attraverso una grammatica, nel modo che verrà precisato dalle seguenti definizioni

Grammatica . . .

- ***Definizione*: Una *grammatica* o *sintassi* G è definita da:**
  - **V, un alfabeto di simboli terminali**
  - **N, un alfabeto di simboli non terminali (detti anche categorie sintattiche), tale che $V \cap N = \varnothing$**
  - **$S \in N$, detto assioma, o simbolo iniziale, o anche simbolo distinto**
  - **P, un insieme finito di regole sintattiche (o produzioni o regole di produzione) del tipo**

$$X \rightarrow \alpha$$

dove $X \in N$ ed $\alpha \in (N \cup V)^*$

e si legge X produce α

. . . Grammatica . . .

- Le produzioni sono talora scritte nella forma

  $X ::= \alpha$ invece che $X \rightarrow \alpha$

- Se in una grammatica esistono più regole aventi la stessa parte sinistra, ad esempio

  $X \rightarrow \alpha_1 \quad X \rightarrow \alpha_2, \quad \ldots, \quad X \rightarrow \alpha_n$

  esse sono raggruppate, usando la convenzione notazionale

  $X \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$

  e in tal caso si dice che $\alpha_1, \alpha_2, \ldots, \alpha_n$ sono parti destre alternative derivabili da X

. . . Grammatica

- **Come visto gli insiemi dei simboli terminali e dei simboli non terminali sono disgiunti**
- **Per distinguere i simboli di questi due insiemi spesso si usa una delle due seguenti convenzioni:**
  - **nella prima i simboli non terminali sono distinti dai terminali perché racchiusi tra parentesi angolate come ad esempio <frase>, <cifra>**
  - **nella seconda i simboli non terminali sono scritti in corsivo - come ad esempio *frase*, *cifra* - e a volte i terminali sono scritti tra apici**

Derivazione diretta

- ***Definizione*: Data una grammatica G e due stringhe**

$\beta, \gamma \in (N \cup V)^*$, si dice che "$\gamma$ deriva direttamente da β in G" e si scrive

$$\beta \rightarrow \gamma$$

se le stringhe si possono decomporre come

$\beta = \eta A \delta$, $\gamma = \eta \alpha \delta$ con $A \in N$; α, η, $\delta \in (N \cup V)^*$

ed esiste la produzione $A \rightarrow \alpha \in P$

  - **Si noti che le stringhe α, η e δ nella definizione precedente possono anche essere stringhe vuote**

Derivazione

- In modo semplice si può definire una catena di derivazioni dirette

$$\beta_0 \rightarrow \beta_1 \rightarrow \beta_2 \ \ldots \rightarrow \beta_n \text{ o anche } \beta_0 \rightarrow^n \beta_n$$

- *Definizione*. Data una grammatica G e due stringhe β, γ ε $(N \cup V)^*$, si dice che "γ deriva da β in G" e si scrive

$$\beta \rightarrow^* \gamma$$

se esiste un n > 0 tale che

$$\beta_0 \rightarrow^n \beta_n \quad \text{e} \quad \beta_0 = \beta \ , \ \beta_n = \gamma$$

Linguaggi generati . . .

- **Le regole di produzione consentono quindi, a partire dall'assioma che è un simbolo non terminale, di derivare via via gli altri simboli non terminali, o loro combinazioni, fino ad arrivare a derivare simboli terminali, o loro combinazioni, da simboli non terminali**
  - In questo modo posso derivare anche un insieme infinito di frasi costituite da tutti e soli simboli terminali
  - Si riesce quindi a generare un insieme infinito di frasi a partire da una loro descrizione finita che è fornita dalla grammatica
  - In analogia, ad esempio, con quanto succede per la descrizione intensionale (che è finita) dell'insieme infinito dei numeri interi

. . . Linguaggi generati

- ***Definizione***:

  Data una grammatica G, dicesi linguaggio generato da G, e si indica con L_G, l'insieme delle frasi di V* (che ovviamente sono costituite da tutti e soli simboli terminali) derivabili a partire dall'assioma S della grammatica G

Linguaggio di programmazione

- ***Definizione***: Un **linguaggio di programmazione** L **su un alfabeto** V **è un sottoinsieme di** V* **per cui esiste una grammatica** G**, tale che** $L=L_G$**, cioè** L **è un linguaggio generato da** G

- Per definire un linguaggio di programmazione c'è quindi bisogno di avere un alfabeto e una grammatica
- Le stringhe o frasi di un linguaggio di programmazione vengono dette programmi (di tale linguaggio)

Backus-Naur-Form - BNF

- Il formalismo appena introdotto per descrivere la grammatica di un linguaggio di programmazione è un metalinguaggio formale che prende il nome di BNF (Backus-Naur-Form, forma di Backus e Naur, dai nomi dei due studiosi che per primi l'hanno introdotta negli anni '50)
- Un metalinguaggio è un linguaggio usato per parlare di un altro linguaggio
  - per esempio, se diciamo "l'articolo determinativo in inglese è 'the' ", od anche "il pronome personale di terza persona singolare è 'he', oppure 'she' oppure 'it'", stiamo usando l'italiano come metalinguaggio per descrivere l'inglese

Extended - BNF . . .

- **Il formalismo BNF viene spesso usato non nella forma originale, ma utilizzando alcune estensioni che permettono una scrittura più concisa delle grammatiche; si parla in questi casi di EBNF (Extended BNF)**
  - **Se nella parte destra di una produzione un simbolo (o sequenza di simboli, o alternativa di simboli) è racchiuso tra parentesi quadre, questo significa che esso è opzionale, che cioè può comparire zero oppure una volta, per esempio**

$$X \rightarrow [\alpha]\ \beta \qquad \text{equivale a} \qquad X \rightarrow \beta \mid \alpha\beta$$

. . . Extended - BNF

- Se invece esso è racchiuso tra parentesi graffe, con un numero intero ad apice, questo significa zero, una o più occorrenze del simbolo stesso, fino ad un massimo di n; per esempio

$$X \rightarrow \{\alpha\}^n\beta$$

significa che da X si può derivare:

$$\beta \qquad \alpha\beta \qquad \alpha\alpha\beta \qquad \alpha\alpha\alpha\beta \qquad \ldots$$

con un massimo di n occorrenze di α

- Se invece un simbolo α è racchiuso tra parentesi graffe (senza apice), come in

$$X \rightarrow \{\alpha\}\beta$$

questo significa zero, una o più (in numero finito, ma arbitrario) occorrenze del simbolo stesso

Albero sintattico

- Il processo di derivazione di una frase mediante un grammatica può essere convenientemente illustrato mediante un albero, detto albero di derivazione sintattica, o più semplicemente albero sintattico
- Piuttosto che definire formalmente la nozione di albero sintattico, la introduciamo attraverso due esempi di derivazione per
  - la frase (già vista) 'il gatto mangia il topo', della lingua italiana
  - i numeri interi senza segno di una o due cifre

Frammento della grammatica italiana

V = { il, lo, gatto, topo, monte, mangia, beve }

N = { <frase>, <soggetto>, <verbo>, <complemento>, <articolo>, <nome> }

S = <frase>

P consiste di:

<frase> ::= <soggetto><verbo><complemento>

<soggetto> ::= <articolo><nome>

<articolo> ::= il | lo | la

<nome> ::= gatto | topo | monte | carne

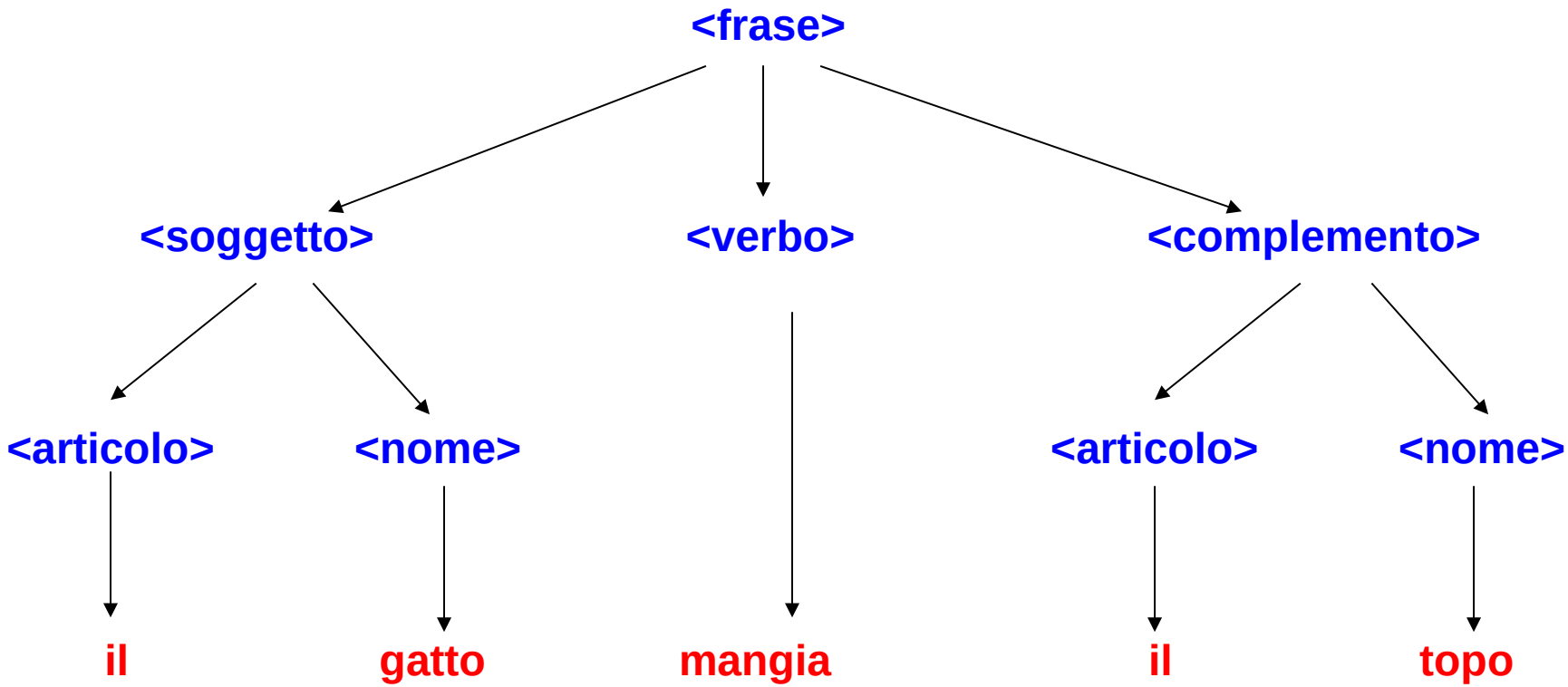
<verbo> ::= mangia | beve

<complemento> ::= <articolo><nome>

N.B. In nero i meta-simboli

Esempio di albero sintattico

- Deriviamo la frase 'il gatto mangia il topo'



Questi ultimi sono simboli terminali del linguaggio

Grammatica per interi senza segno di una o due cifre

V = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

N = {<intero-senza-segno>, <cifra-non-nulla>, <cifra>}

S = <intero-senza-segno>

P consiste di:

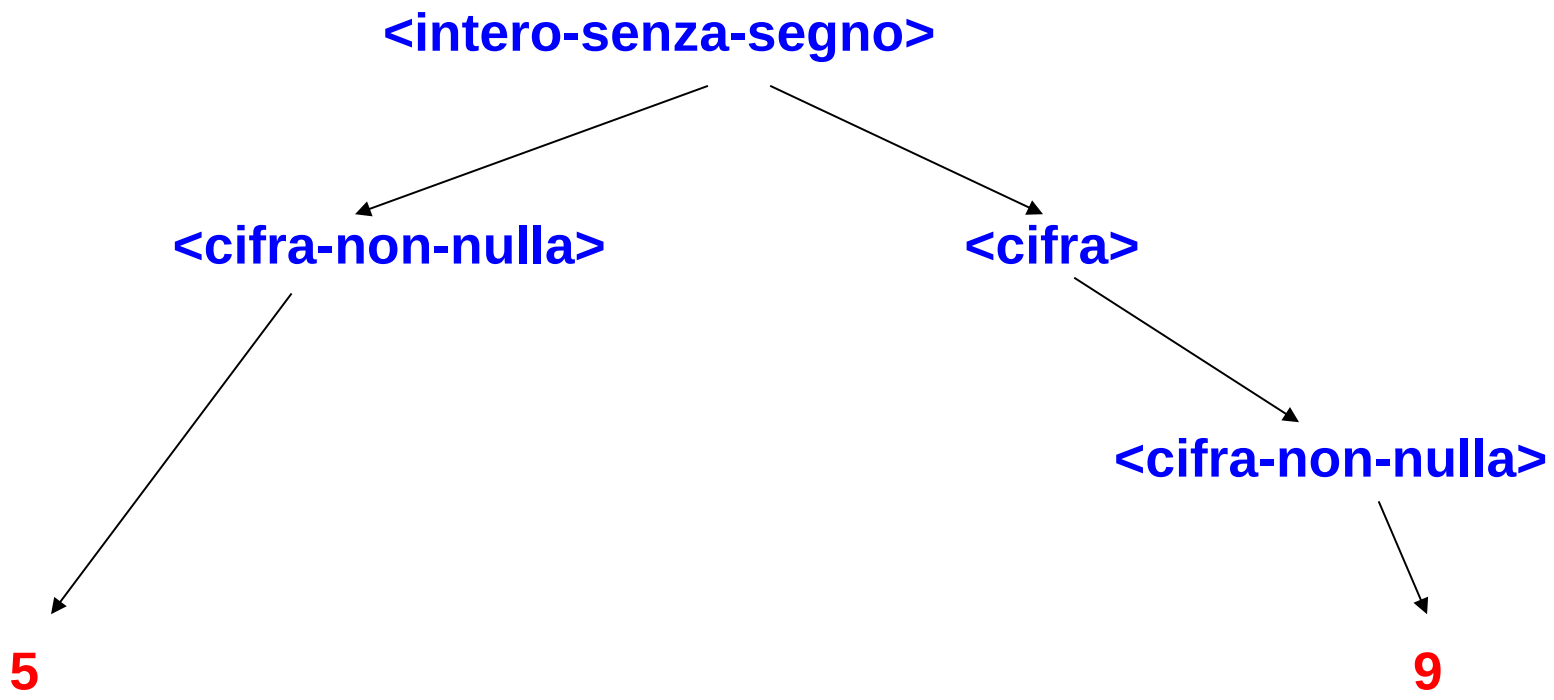
<intero-senza-segno> ::= [<cifra-non-nulla>]<cifra>

<cifra> ::= <cifra-non-nulla> | 0

<cifra-non-nulla> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Esempio di albero sintattico

- Deriviamo il numero intero senza segno 59



Questi ultimi sono simboli terminali del linguaggio

Sintassi dei linguaggi di programmazione

- **La definizione della sintassi di un linguaggio di programmazione viene data definendo la grammatica da cui viene generato**
  - il lessico (cioè un insieme di simboli terminali, che è il vocabolario) del linguaggio
  - un insieme di simboli non terminali, tra cui ne viene scelto uno come simbolo iniziale, cioè l'assioma
  - un insieme di regole di produzione, in genere espresse in una qualche variante della notazione BNF

Il lessico

Il lessico è costituito da

- Un alfabeto di caratteri e cifre che servono a costruire identificatori (ad esempio i nomi di classi, oggetti, metodi, variabili, . . .)
- Un insieme di simboli speciali corrispondenti ad operatori e simboli di interpunzione
- Un insieme finito di parole chiave, cioè sequenze di caratteri dell'alfabeto che sono riservate in quanto assumono, a livello semantico, significati particolari nel linguaggio

Introduzione alla sintassi di Java

- **La sintassi di Java si occupa della formazione di frasi valide in Java, mediante la formalizzazione delle "regole sintattiche"**
  - la definizione di una classe è formata dalla parola class, seguita dal nome della classe e dal corpo della classe
  - il nome di una classe è un identificatore
  - un identificatore è una sequenza non vuota di caratteri alfanumerici, iniziante per un carattere alfabetico
  - il corpo di una classe è formato da un elenco di dichiarazioni della classe, racchiuso tra parentesi graffe { e }
  - possibili dichiarazioni di una classe sono: la definizione di un metodo, la dichiarazione di una variabile
  - la definizione di un metodo è formata dall'intestazione del metodo seguita dal corpo del metodo
  - il corpo di un metodo è un blocco
  - un blocco è una sequenza di istruzioni e dichiarazioni racchiusa tra parentesi graffe { e }

Sintassi di Java

- **La sintassi di Java è descritta da una grammatica composta da**
  - **elementi terminali - lessico o vocabolario – cioè le parole e i simboli che possono comparire nei programmi**

    class public . , ; { } a b c d ... 0 1 2 ...
  - **elementi non terminali - le categorie sintattiche - utilizzate per la descrizione dei programmi - ma che non compaiono nei programmi**

    definizione-classe identificatore corpo-classe definizione-metodo blocco sequenza-istruzioni-blocco istruzione **...**
  - **assioma - l'elemento non terminale *unità-di-compilazione* che guida la scrittura di un intero programma o classe**
  - **produzioni (o regole sintattiche) - le regole che specificano come sia possibile derivare frasi da ciascun non terminale**

Esempi di produzioni . . .

definizione-classe ::=

class *identificatore-classe* *corpo-classe*

corpo-classe ::=

{ {*dichiarazione-corpo-classe*} **}**

N.B. Le parentesi { e } sono simboli terminali del linguaggio, mentre le parentesi { e } sono simboli del metalinguaggio EBNF

dichiarazione-corpo-classe ::=

definizione-metodo | *definizione-costruttore* |

dichiarazione-variabile

. . . Esempi di produzioni . . .

definizione-classe **::=**

class ***identificatore-classe corpo-classe***

- **la prima riga di una produzione contiene un simbolo non- terminale**
  - **la produzione ha lo scopo di descrivere le possibili forme per questo simbolo non terminale**
- **nelle righe successive alla prima vengono descritti i possibili modi per espandere il non terminale**
  - **questa produzione afferma che una *definizione-classe* è formata dal simbolo terminale class, seguito da un *identificatore-classe* e da un *corpo-classe***
  - **le forme per *identificatore-classe* e *corpo-classe* sono descritte dalle rispettive produzioni**

. . . Esempi di produzioni . . .

Una possibile definizione alternativa è

definizione-classe ::=

class *identificatore-classe* *corpo-classe*

corpo-classe ::=

{ *dichiarazioni-corpo-classe* }

dichiarazioni-corpo-classe ::=

dichiarazione-corpo-classe |

dichiarazione-corpo-classe *dichiarazioni-corpo-classe*

dichiarazione-corpo-classe ::=

definizione-metodo | *definizione-costruttore* |

dichiarazione-variabile

. . . Esempi di produzioni

corpo-metodo ::=

{ *istruzioni-dichiarazioni-corpo-metodo* }

istruzioni-dichiarazioni-corpo-metodo ::=

istruzione-dichiarazione-corpo-metodo |

istruzione-dichiarazione-corpo-metodo
istruzioni-dichiarazioni-corpo-metodo

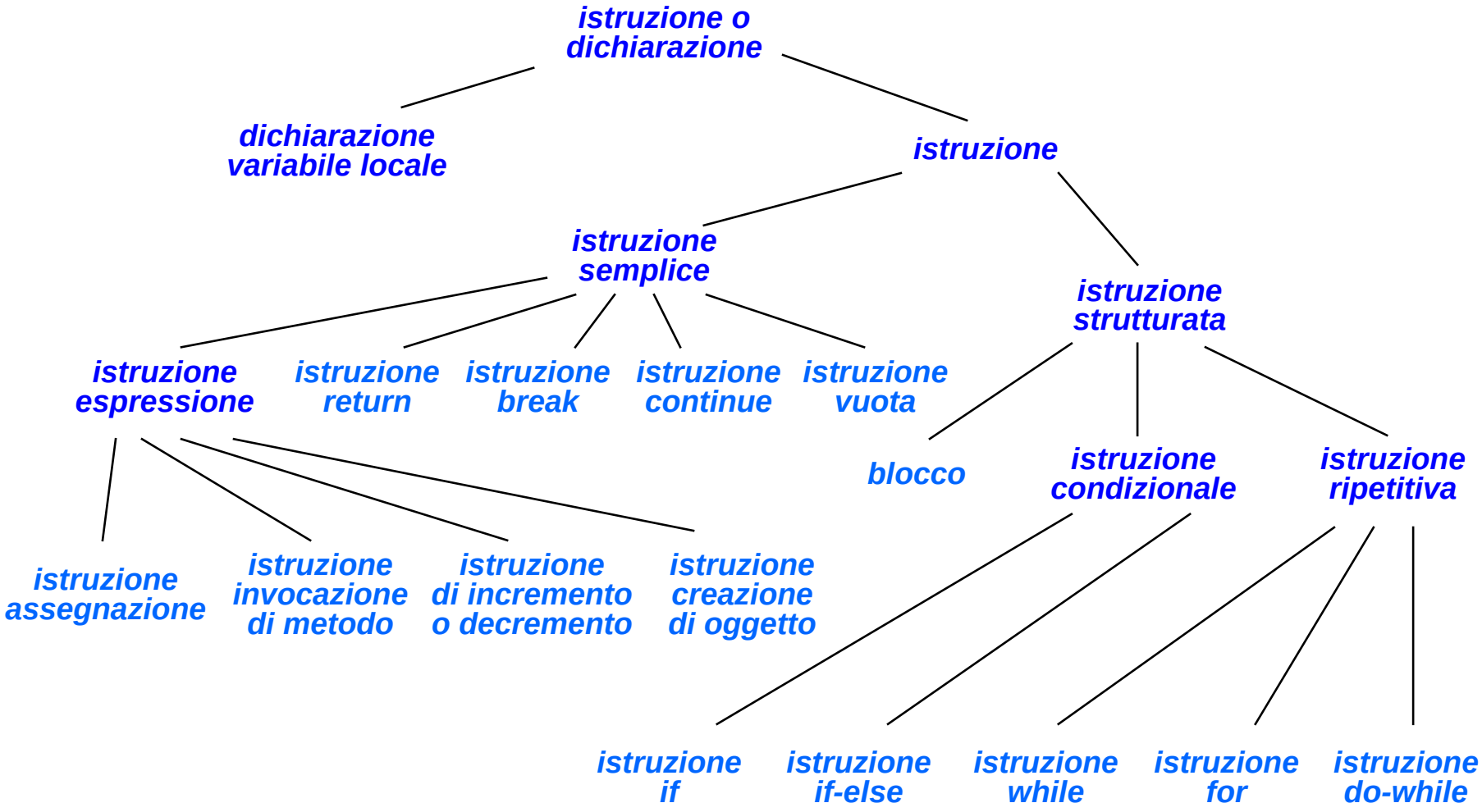
istruzione-dichiarazione-corpo-metodo ::=

dichiarazione-variabile | *istruzione*

istruzione ::=

istruzione-semplice | *istruzione-strutturata*

Albero delle istruzioni di Java



Esempio — identificatori . . .

- **I nomi delle classi, dei metodi e delle variabili appartengono alla categoria grammaticale degli identificatori**
  - **alcuni esempi di identificatori sono**
    - **Math**, **println**, **frase**, **sqrt**, **RadiceQuadrata**
    - **System.out** **non è un identificatore**

. . . Esempio — identificatori

- **La regola (informale) per la formazione degli identificatori è**
  - **un identificatore è una sequenza non vuota di caratteri alfanumerici (alfabetici e numerici), iniziante con un carattere alfabetico**
    - **in realtà, sono ammessi anche alcuni caratteri speciali, come il carattere “underscore” _**
    - **alcune sequenze di caratteri sono riservate — come ad esempio class e public — non sono identificatori ma parole chiave**
  - **l’uso dei caratteri minuscoli e maiuscoli è significativo**
    - **ad esempio, alfa e Alfa sono identificatori diversi**

Sintassi per gli identificatori

identificatore ::=
    carattere-alfabetico |
    carattere-alfabetico {*carattere-alfanumerico*}
carattere-alfanumerico ::=
    carattere-alfabetico | *cifra*
carattere-alfabetico ::=
    uno di **a à b c ... x y z A À B C ... X Y Z … _ …**
cifra ::=
    uno di **0 1 2 ... 8 9**
parola-chiave ::=
    una di **abstract boolean char class continue do double else final float for if import instanceof int interface long new package private protected public return short static this void while *e altre ancora . . .***

Semantica di Java

- In questo corso la semantica di Java viene descritta in modo informale

- Una istruzione valida
  - System.out.println("ciao a tutti");
- Una istruzione ben formata – ma non valida
  - System.out.stampa("ciao a tutti");

    . . . Non esiste il metodo stampa
- Una istruzione che è non ben formata
  - System.out.println("ciao a tutti";

    . . . Manca una parentesi tonda chiusa

Semantica di una frase

- I tipi rivestono un ruolo importante nel discriminare tra frasi valide e no
- La semantica di una frase dipende anche dal tipo della frase
  - la semantica di una espressione viene data in termini di un tipo e di un valore
  - la semantica di una istruzione viene data in termini dell'effetto dell'esecuzione dell'istruzione
    - in modo diverso per istruzioni semplici e istruzioni strutturate (composte)

Sintassi, semantica ed errori

- **Possibili errori di programmazione**
  - **la frase non è ben formata**
    - **errori sintattici o grammaticali**
  - **la frase è ben formata ma non è valida**
    - **errori semantici**
    - **errori di semantica statica ed errori di semantica dinamica**
  - **la frase è valida – ma il suo significato è diverso da quello voluto**
    - **errori logici**

Esempi di errori

- **Errori sintattici**

```
System.out.println("Ciao a tutti"];
System.outbprintln("Ciao a tutti");
```

- **Errori semantici**

```
System.out.stampa("Ciao a tutti");
system.out.println("Ciao a tutti");
```

  - Questi sono errori di **semantica statica**

- **Errori logici**

```
System.out.println("Ciao a totti");
```

Cosa abbiamo visto finora

- Come si definisce un linguaggio
- Cosa sono la sintassi e la semantica
- Come si definiscono i linguaggi di programmazione
- Come si definiscono le grammatiche
- Cosa è il meta-linguaggio BNF
- Come si definisce la sintassi dei linguaggi di programmazione
- Quali sono la sintassi e la semantica del linguaggio Java
- Che relazione c'è tra sintassi, semantica ed errori

Riferimenti al libro di testo

- **Per lo studio della sintassi e della semantica del linguaggio Java si fa riferimento al libro di testo, e in particolare al capitolo**
  - **6 – Le basi del linguaggio Java**