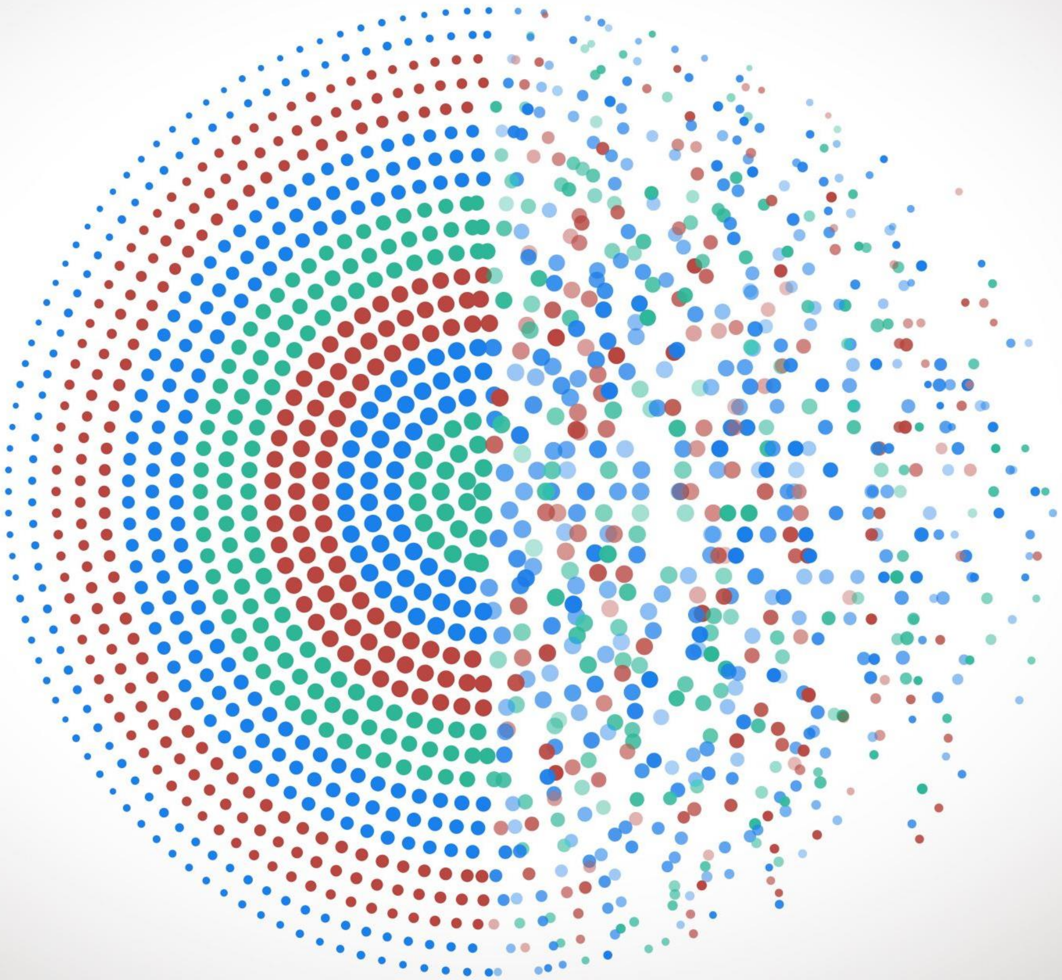
Fondamenti di programmazione

a.a. 2022-23

Antinisca Di Marco
antinisca.dimarco@univaq.it



Riprendiamo i concetti

- Cosa abbiamo visto?

Grammatiche

- Le grammatiche libere da contesto sono un strumento **generativo** (**ricorsivo**) per descrivere linguaggi
- le grammatiche costituiscono una notazione concisa per descrivere la sintassi di un linguaggio di programmazione.

Grammatiche

- Le espressioni aritmetiche possono essere definite ricorsivamente in modo naturale. Consideriamo le espressioni aritmetiche che contengono:
  - i quattro operatori binari $+$, $-$, $*$ e $/$;
  - le parentesi;
  - i numeri come operandi.

Tali espressioni vengono di solito definite in modo induttivo come segue:

Base. Un numero è un'espressione.

Induzione. Se E è un'espressione, lo sono anche:

a) (E)

b) $E + E$

c) $E - E$

d) $E * E$;

e) E/E.

Questa induzione definisce in maniera generativa un linguaggio, ossia un insieme di stringhe

Grammatiche

- Le espressioni aritmetiche possono essere definite ricorsivamente in modo naturale. Consideriamo le espressioni aritmetiche che contengono:
  - i quattro operatori binari +, −, ∗ e /;
  - le parentesi;
  - i numeri come operandi.

Tali espressioni vengono di solito definite

Base. Un numero è un’espressione.

Induzione. Se E è un’espressione, lo sonc

a) (E)

b) E + E

c) E − E

d) E ∗ E;

e) E/E.

Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica

<Espressione> → <Numero>
<Espressione> → (<Espressione>)
<Espressione> → <Espressione> + <Espressione>
<Espressione> → <Espressione> – <Espressione>
<Espressione> → <Espressione> * <Espressione>
<Espressione> → <Espressione> / <Espressione>

Grammatiche

- <Espressione> viene detto categoria sintattica e sta per una qualunque stringa nel linguaggio delle espressioni aritmetiche.
- Il simbolo → significa "può essere composto da": per esempio, la regola <Espressione> → (<Espressione>) dice che un'espressione può essere composta da una parentesi aperta, seguita da una qualunque stringa che sia un'espressione, seguita da una parentesi chiusa.

Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica

<Espressione> → <Numero>
<Espressione> → (<Espressione>)
<Espressione> → <Espressione> + <Espressione>
<Espressione> → <Espressione> - <Espressione>
<Espressione> → <Espressione> * <Espressione>
<Espressione> → <Espressione> / <Espressione>

Grammatiche

- Nella prima regola il simbolo <Numero> a destra della freccia è anch'esso una categoria sintattica, da interpretarsi come un segnaposto per una generica stringa che possa essere interpretata come un numero.
- Allo stato attuale non ci sono regole in cui <Numero> compaia a sinistra della freccia, e quindi non è ancora definito quali stringhe possano essere usate per rappresentare i numeri.

Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica

<Espressione> → <Numero>
<Espressione> → (<Espressione>)
<Espressione> → <Espressione> + <Espressione>
<Espressione> → <Espressione> - <Espressione>
<Espressione> → <Espressione> * <Espressione>
<Espressione> → <Espressione> / <Espressione>

Grammatiche

- Una grammatica è costituita da una o più produzioni : ogni linea della riquadro affianco è una produzione.
- In generale, una produzione `e formata da tre parti:

1. una testa, che è la categoria sintattica a sinistra della freccia;
2. il metasimbolo →;
3. il corpo, costituito da 0 o più categorie sintattiche e/o simboli terminali a destra della freccia.

Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica

<Espressione> → <Numero>
<Espressione> → (<Espressione>)
<Espressione> → <Espressione> + <Espressione>
<Espressione> → <Espressione> - <Espressione>
<Espressione> → <Espressione> * <Espressione>
<Espressione> → <Espressione> / <Espressione>

Convenzioni

- <S> → ε significa che la stringa vuota fa parte del linguaggio della categoria sintattica <S>.
- <S> → B1, <S> → B2, . . . ,<S> → Bn è equivalente a

<S> → B1 | B2 | · · · | Bn

<Espressione> → <Numero>
<Espressione> → (<Espressione>)
<Espressione> → <Espressione> + <Espressione>
<Espressione> → <Espressione> - <Espressione>
<Espressione> → <Espressione> * <Espressione>
<Espressione> → <Espressione> / <Espressione>

<Espressione> → <Numero> |
(<Espressione>) |
<Espressione> + <Espressione> |
<Espressione> - <Espressione> |
<Espressione> * <Espressione> |
<Espressione> / <Espressione>

Completiamo la grammatica delle espressioni

<Espressione> → <Numero> |
(<Espressione>) |
<Espressione> + <Espressione> |
<Espressione> - <Espressione> |
<Espressione> * <Espressione> |
<Espressione> / <Espressione>
<Numero> → ???

Completiamo la grammatica delle espressioni

<Espressione> → <Numero> |
(<Espressione>) |
<Espressione> + <Espressione> |
<Espressione> - <Espressione> |
<Espressione> * <Espressione> |
<Espressione> / <Espressione>

<Numero> → <Cifra> | <Numero> <Cifra>

<Cifra> → 0| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Abbiamo definito la nostra prima Grammatica!

Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

lambda

rho

Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, \mathbf{P} >$$

Λ è un insieme di simboli detto alfabeto

Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

Λ è un insieme di simboli detto alfabeto

V è l'insieme, finito, delle categorie sintattiche, ovvero di variabili che rappresentano sotto-linguaggi

Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

Λ è un insieme di simboli detto alfabeto

V è l'insieme, finito, delle categorie sintattiche, ovvero di variabili che rappresentano sotto-linguaggi

$S \in V$ è la categoria sintattica principale o iniziale

Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

Λ è un insieme di simboli detto alfabeto

V è l'insieme, finito, delle categorie sintattiche, ovvero di variabili che rappresentano sotto-linguaggi

S $\in$ V è la categoria sintattica principale o iniziale

P è un insieme finito di produzioni. Ciascuna produzione, nel caso delle grammatiche libere ha la struttura $A \to \alpha$

$A \in V$

$\alpha \in (\Lambda \cup V)^+$

Definiamo formalmente la nostra grammatica

<Espressione> → <Numero> |
(<Espressione>) |
<Espressione> + <Espressione> |
<Espressione> - <Espressione> |
<Espressione> * <Espressione> |
<Espressione> / <Espressione>

<Numero> → <Cifra> | <Numero> <Cifra>

<Cifra> → 0| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

$G = < \mathbf{\Lambda}, V, S, \mathbf{P} >$

Λ è un insieme di simboli detto alfabeto

V è l'insieme, finito, delle categorie sintattiche, ovvero di variabili che rappresentano sotto-linguaggi

$S \in V$ è la categoria sintattica principale o iniziale

$\Lambda_{espressione}$ = { *,-,+,/, (,) 0, 1, …, 9 }

$V_{espressione}$ = { <Espressione>, <Numero>, <Cifra> }

$S_{espressione}$ = <Espressione>

$P_{espressione}$

Esercizi: Definire una grammatica G per ognuno dei seguenti linguaggi

$L(G1) = \{ a^n \mid n \geq 1 \}$

$L(G2) = \{ a^n \mid n \geq 0 \}$

$L(G3) = \{ a^n b^m \mid n \geq 1, m \geq 0 \}$

$L(G4) = \{ a^n b^n \mid n \geq 1 \}$

$L(G5) = \{ a^n (bc)^m \mid n \geq 0, m \geq 0 \}$

$L(G6) = \{ a^n b^m c^n \mid n \geq 0, m \geq 0 \}$

Linguaggi generati da grammatiche

- Una grammatica è essenzialmente una definizione induttiva che coinvolge insiemi di stringhe
- Spesso una stessa grammatica definisce varie categorie sintattiche contemporaneamente.
- Ad ogni categoria sintattica definita da una grammatica si può associare un linguaggio.

Linguaggi generati da grammatiche

- Per ogni categoria sintattica <S> di una grammatica, definiamo un linguaggio associato L(<S>) nel modo seguente.
- **Passo Base**. Si parte assumendo che, per ogni categoria sintattica <S> della grammatica, il linguaggio L(<S>) sia vuoto.
- **Passo Induttivo**. Supponiamo che la grammatica contenga la produzione $<S> \rightarrow X_1X_2 \cdots X_n$, in cui ogni X_i, per $i = 1, 2, \ldots, n$, è una categoria sintattica o un simbolo terminale.

  Per ciascun $i = 1, 2, \ldots, n$, si seleziona una stringa s_i per X_i nel modo seguente:
  - se X_i è un simbolo terminale, allora la stringa s_i è X_i stesso;
  - se X_i è una categoria sintattica, allora s_i è una qualunque stringa che già sappiamo appartenere a $L(X_i)$. Se la stessa categoria sintattica X_i appare più volte nel corpo della produzione, si può scegliere da $L(X_i)$ una stringa diversa per ogni occorrenza di X_i.
- Allora, la concatenazione $s_1s_2 \cdots s_n$ delle stringhe così selezionate appartiene al linguaggio L(<S>).
- Notiamo che, se $n = 0$, la stringa vuota appartiene al linguaggio.

Linguaggi generati da grammatiche

- Consideriamo la grammatica che contiene alcune per i comandi Pascal.
- Per semplicità, useremo solo le produzioni per i comandi *while*, per il *blocco* e per i *comandi semplici*, oltre alle due produzioni per le sequenze di comandi.
- Inoltre, useremo un'abbreviazione: faremo uso dei simboli terminali w (*while*), c (*condizione*), d (*do*), b (*begin*), e (*end*), s (al posto della categoria sintattica < *ComSemplice* >) e punto e virgola.
- La grammatica utilizza la categoria sintattica <S> per i comandi e la categoria sintattica <L> per sequenze di comandi

1. $<S> \rightarrow \mathbf{w\ c\ d} <S>$
2. $<S> \rightarrow \mathbf{b} <L> \mathbf{e}$
3. $<S> \rightarrow \mathbf{s}$
4. $<L> \rightarrow <L> ; <S>$
5. $<L> \rightarrow <S>$

	S	L
Ciclo 1:	s	
Ciclo 2:	wcds	s
Ciclo 3:	wcdwcds	s;wcds
	bse	s;s
		wcds

Osservazione

- Il linguaggio generato da una grammatica è in generale infinito.
- Se un linguaggio è infinito, non possiamo elencare tutte le sue stringhe: il meglio che possiamo fare è enumerare le stringhe ad ogni ciclo, come avevamo cominciato a fare nell'Esempio precedente.
- L'insieme delle stringhe che vengono a far parte, prima o poi, del linguaggio della categoria sintattica <S> costituisce il linguaggio (infinito) L(<S>).