Azzolini Riccardo 2020-11-23

Grammatiche libere dal contesto

1 Generazione e riconoscimento di linguaggi non regolari

Esistono linguaggi che, pur essendo non regolari, possono essere generati e riconosciuti in modo algoritmico. Ad esempio, il linguaggio dei palindromi L_{pal} può essere generato induttivamente,

- *base*: ϵ, 0 e 1 appartengono a L_{pal};
- *passo*: se $w \in L_{pal}$, allora anche le stringhe $0w0$ e $1w1$ appartengono a L_{pal};
- nessuna altra stringa appartiene a L_{pal};

e può essere riconosciuto con la seguente procedura (qui riportata in linguaggio Java), che data una stringa $w \in \{0,1\}^*$ restituisce `true` se e solo se $w \in L_{pal}$:

```
boolean isPalindrome(String w) {
    for (int i = 0; i < w.length(); i++) {
        if (w.charAt(i) != w.charAt(w.length() - 1 - i)) {
            return false;
        }
    }
    return true;
}
```

Fatte queste osservazioni, viene da chiedersi se sia possibile caratterizzare in generale, tramite generatori e riconoscitori, la classe di linguaggi a cui appartiene L_{pal}. La risposta è sì: in seguito verranno introdotti i formalismi necessari, a partire dal generatore per questa classe di linguaggi.

2 Grammatiche libere dal contesto

Una **grammatica libera dal contesto** (**Context-Free Grammar**, **CFG**) è una quadrupla $G = \langle V, T, \Gamma, S \rangle$ in cui:

- V è un insieme finito i cui elementi sono detti **simboli non-terminali** (o *variabili*, o ancora *categorie sintattiche*);
- T è un insieme finito i cui elementi sono detti **simboli terminali**;

- Γ è l'insieme finito delle **regole di produzione**, ciascuna delle quali è una coppia $A \to \alpha$, dove:
  - $A \in V$ è un simbolo non-terminale, che prende il nome di **testa** o **lato sinistro** della (regola di) produzione;
  - $\alpha \in (V \cup T)^*$ è una sequenza di terminali e non-terminali, chiamata **corpo** o **lato destro** della produzione;
- $S \in V$ è il **simbolo iniziale** della grammatica.

Il significato di questi elementi verrà formalizzato più avanti, ma sostanzialmente:

- i simboli non-terminali rappresentano elementi sintattici del linguaggio;
- i simboli terminali sono quelli che compaiono concretamente nelle stringhe del linguaggio;
- le regole di produzione indicano come sostituire ciascun simbolo non-terminale con sequenze di altri simboli (terminali e non-terminali) per generare le stringhe del linguaggio;
- il simbolo iniziale è il simbolo non-terminale a partire dal quale vengono generate le stringhe del linguaggio.

2.1 Esempio: linguaggio dei palindromi

Come si dimostrerà più avanti, il linguaggio dei palindromi L_{pal} è generato dalla grammatica $G_{pal} = \langle \{P\}, \{0, 1\}, \Gamma, P \rangle$, dove Γ contiene le produzioni:

$$
\begin{aligned}
P &\to \epsilon \\
P &\to 0 \\
P &\to 1 \\
P &\to 0P0 \\
P &\to 1P1
\end{aligned}
$$

Per compattare la notazione, le regole di produzione per uno stesso non-terminale possono essere raggruppate utilizzando il simbolo $|$:

$$P \to \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$$

Informalmente, il processo secondo cui questa grammatica genera, ad esempio, la stringa $0110 \in L_{pal}$ è il seguente:

1. si parte dal simbolo iniziale P;
2. si applica la produzione $P \to 0P0$, ottenendo appunto $0P0$;

3. si applica $P \to 1P1$, ottenendo $01P10$;
4. si applica infine $P \to \epsilon$, ottenendo $01\epsilon10 = 0110$.

2.2 Esempio: linguaggio delle espressioni

Si consideri la grammatica

$$G_{\text{Exp}} = \langle \{E, I\}, \{+, *, (,), a, b, 0, 1\}, \Gamma, E \rangle$$

dove Γ contiene le produzioni:

$$E \to I \mid E + E \mid E * E \mid (E)$$
$$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

Questa CFG genera un semplice linguaggio delle espressioni, composte da identificatori, operatori ($+$, $*$) e parentesi — una versione semplificata delle espressioni che si ritrovano tipicamente nei linguaggi di programmazione. Un esempio di stringa di questo linguaggio è $ab0 + (b1 * a)$.

Il simbolo non-terminale I genera il linguaggio degli identificatori, che iniziano con una lettera e continuano con una sequenza di zero o più lettere e cifre (per semplicità, sono consentite solo le lettere a e b e le cifre 0 e 1, ma si potrebbe facilmente estendere la grammatica in modo da consentire anche altri caratteri). Si osserva che questo è un linguaggio *regolare*: oltre che dalla grammatica appena presentata, esso è generato dall'espressione regolare $(a + b)(a + b + 0 + 1)^*$.

3 Backus-Naur form

Nelle applicazioni pratiche delle CFG è tipicamente necessario rappresentarle in forma testuale, mediante una *sintassi concreta* che sia facilmente manipolabile tramite un editor di testo. Una tale sintassi concreta è la **Backus-Naur form**, **BNF**. Essa è sostanzialmente simile alla rappresentazione usata finora, con due principali differenze:

- i simboli non-terminali sono racchiusi tra parentesi angolari, `<>`;
- la freccia delle regole di produzione è sostituita dal simbolo `::=`.

Ad esempio, la rappresentazione in BNF della grammatica delle espressioni vista prima è:

```
<expression> ::= <identifier> | <expression> + <expression> |
                 <expression> * <expression> | ( <expression> )
<identifier> ::= a | b | <identifier> a | <identifier> b |
                 <identifier> 0 | <identifier> 1
```

4 Carte sintattiche

Un'altra rappresentazione, questa volta grafica, delle CFG sono le **carte sintattiche**, nelle quali:

- ogni regola di produzione è rappresentata da un diagramma;
- i simboli non-terminali nel corpo di ciascuna regola sono rappresentati in dei rettangoli;
- i simboli terminali sono rappresentati in dei rettangoli arrotondati.

Ad esempio, una rappresentazione a carte sintattiche della grammatica delle espressioni è la seguente: