5. Introduzione all'analisi sintattica

Il parser, come detto, esamina il flusso di token (prodotti dal lexer) per verificarne la correttezza grammaticale. Stabilisce quindi se l'input è sintatticamente ben formato, costruendo una rappresentazione intermedia (IR) del codice.

Tuttavia, il processo di individuazione di una derivazione per una certa frase richiede un *modello* della sintassi, ovvero una grammatica G. Inoltre, è necessario disporre di un algoritmo in grado di verificare l'appartenenza a $L(G)$.

Grammatiche libere dal contesto

Una grammatica libera dal contesto (CFG) è una quadrupla $G = \langle S, N, T, P \rangle$ dove:

- N è un insieme di variabili, anche dette **simboli non terminali**
- T è un insieme di **simboli terminali** (tali che $V \cap T = \emptyset$)
- P è un insieme finito di **produzioni** o **regole di riscrittura**; ogni produzione è della forma $A \to \alpha$, con $A \in V$ e $\alpha \in (V \cup T)^*$
- S è una variabile, detta **simbolo iniziale**

Esempio. La grammatica

$$G = \{\{E\}, \{or, and, not, (,), 0, 1\}, P, E\}$$

ove P è costituito da

$$\begin{aligned} \mathtt{E} &\mapsto \mathtt{0} \\ \mathtt{E} &\mapsto \mathtt{1} \\ \mathtt{E} &\mapsto \mathtt{(E\ or\ E)} \\ \mathtt{E} &\mapsto \mathtt{(E\ and\ E)} \\ \mathtt{E} &\mapsto \mathtt{(not\ E)} \end{aligned}$$

genera le possibili espressioni booleane.

Nota. La grammatica G può essere riscritta come segue

$$\mathtt{E} \mapsto \mathtt{0}|\mathtt{1}|(\mathtt{E}\ \mathtt{or}\ \mathtt{E})|(\mathtt{E}\ \mathtt{and}\ \mathtt{E})|(\mathtt{not}\ \mathtt{E})\,.$$

Linguaggio generato

Le grammatiche servono a **generare** un linguaggio.

Definiremo le relazioni $\overset{G}{\Longrightarrow}, \overset{G}{\Longrightarrow}_i, \overset{G}{\Longrightarrow}_* \subseteq (V \cup T)^* \times (V \cup T)^*$ nel seguente modo:

- se $A \to \beta \in P$ e $\alpha, \gamma \in (V \cup T)^*$, allora $\alpha A \gamma \overset{G}{\Longrightarrow} \alpha\beta\gamma$. Diremo in questo caso che da $\alpha A \gamma$ deriva immediatamente $\alpha\beta\gamma$;
- se $\alpha_1, \ldots, \alpha_i \in (V \cup T)^*$, con $i \geq 1$

  $$\bigwedge_{j=1}^{i-1} \alpha_j \overset{G}{\Longrightarrow} \alpha_{j+1}$$

  allora $\alpha_1 \overset{G}{\Longrightarrow}_i \alpha_m$. In questo caso diremo che da α_1 deriva α_m in i passi. Per ogni α, diremo anche che $\alpha \overset{G}{\Longrightarrow}_0 \alpha$;
- se esiste un i tale per cui $\alpha \overset{G}{\Longrightarrow}_i \beta$, allora $\alpha \overset{G}{\Longrightarrow}_* \beta$. Diremo in tal caso che da α deriva β.

Si noti che $\overset{G}{\Longrightarrow}_*$ è la chiusura transitiva e riflessiva della relazione $\overset{G}{\Longrightarrow}$. In particolare, vale che $\alpha \overset{G}{\Longrightarrow}_* \alpha$ per ogni $\alpha \in (V \cup T)^*$.

Il linguaggio generato da G è:

$$L(G) = \{w \in T^* : S \overset{G}{\Longrightarrow}_* w\}$$

L è un linguaggio libero dal contesto (CF) se esiste una grammatica CF G tale che $L = L(G)$.

Nota. Due grammatiche G_1 e G_2 sono equivalenti se $L(G_1) = L(G_2)$.

Alberi di derivazione

Le derivazioni generate da una grammatica vengono ben visualizzate mediante l'ausilio di strutture dati ad albero.

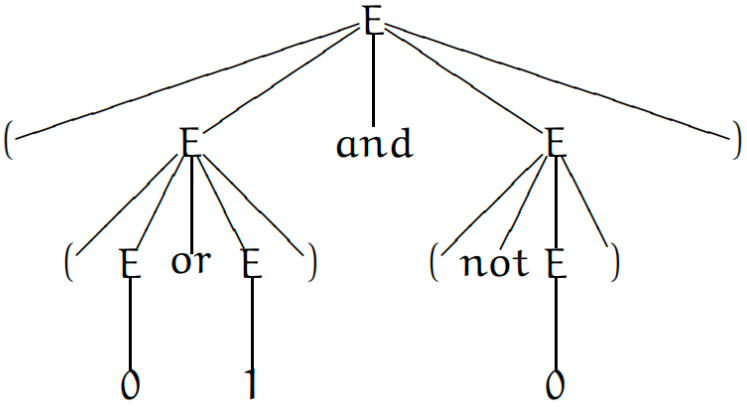
Sia $G = \langle V, T, P, S \rangle$ una grammatica CF. Un albero è un albero di derivazione (**parse tree**) per G se:

1. ogni vertice ha una etichetta, presa tra $V \cup T \cup \{\epsilon\}$;
2. l'etichetta della radice appartiene a V;
3. ogni vertice interno (ovvero, non una foglia) ha etichetta appartenente a V;
4. se un vertice n è etichettato con A e $n_1, \ldots, n_k$ sono (ordinatamente, da sinistra a destra) i vertici figli di A etichettati con $X_1, \ldots, X_k$, allora $A \to X_1 \cdot X_2 \ldots X_k \in P$;
5. se un vertice n ha etichetta ϵ, allora n è una foglia ed è l'unico figlio di suo padre.

Si noti che la definizione di albero non impone che tutte le foglie siano etichettate con simboli terminali. L'albero avente un solo nodo etichettato da S è un caso particolare di albero di derivazione. Gli alberi verranno utilizzati per dare una controparte grafica delle derivazioni. Se vi sono foglie etichettate con simboli non terminali, allora l'albero rappresenta una derivazione parziale.

Diremo che un albero descrive una stringa $\alpha \in (V \cup T)^*$ se α è proprio la stringa che possiamo leggere dalle etichette delle foglie da sinistra a destra.

Esempio. La grammatica dell'esempio genera, in particolare, la stringa `w = ((0 or 1) and (not 0))`. La derivazione per w è rappresentata dall'albero seguente



Dato un albero ed un suo nodo n, il sottoalbero identificato da quel nodo è costituito da quel nodo più tutti i suoi discendenti (figli, nipoti, etc.), nonché le varie etichette associate ad essi.

Sia $G = \langle V, T, P, S \rangle$ una grammatica CF. Allora $S \overset{G}{\Longrightarrow}_* \alpha$ se e solo se esiste un albero di derivazione con radice etichettata S per G che descrive α.

Ambiguità delle derivazioni

Se ad ogni passo di una derivazione la produzione è applicata al simbolo non terminale più a sinistra, allora la derivazione è detta **sinistra** (leftmost). Similmente, se viene applicata

sempre a quello a destra, allora è detta **destra** (rightmost).

Se $w \in L(G)$, sappiamo che per essa esiste almeno un albero di derivazione con radice etichettata S. Si può dimostrare che, fissato un albero di derivazione con radice etichettata S , esattamente una derivazione sinistra (suggerimento: si visiti l'albero in preordine) ed una derivazione destra (simmetricamente) possono essere desunte.

Un albero di derivazione rappresenta dunque un certo insieme di possibili derivazioni distinte di una stessa stringa. Ci possono tuttavia essere più alberi di derivazione per la stessa parola.

Esempio. Si consideri la grammatica

$$E \to E + E \mid E * E \mid 0 \mid 1 \mid 2$$

Una derivazione sinistra è

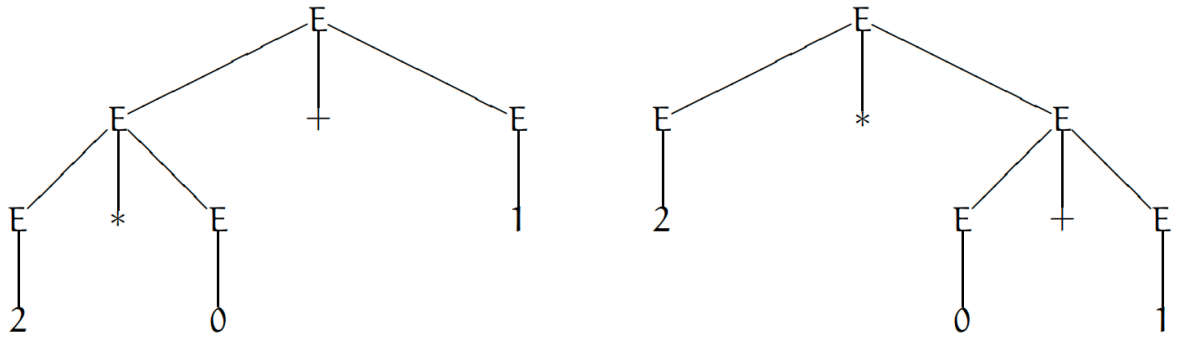
$$E \overset{G}{\Longrightarrow} E + E \overset{G}{\Longrightarrow} E * E + E \overset{G}{\Longrightarrow}_3 2 * 0 + 1$$

In questo esempio, la derivazione procede applicando le regole della grammatica seguendo un approccio sinistro (leftmost), trasformando progressivamente E fino ad ottenere la stringa finale $2 * 0 + 1$.

Un'altra derivazione sinistra per la stessa stringa è:

$E \overset{G}{\Longrightarrow} E * E \overset{G}{\Longrightarrow} 2 * E \overset{G}{\Longrightarrow} 2 * E + E \overset{G}{\Longrightarrow}_2 2 * 0 + 1.$

Tuttavia, gli alberi associati alle due derivazioni (che potrebbero essere visti come alberi per la computazione dell'espressione associata alla stringa generata) sono diversi



Il primo è intuitivamente associato a $(2 * 0) + 1 = 1$, mentre il secondo, invece, a $2 * (0 + 1) = 2$.

Una grammatica CF tale per cui esiste una parola con più di un albero di derivazione con radice etichettata S è detta *ambigua*. Un linguaggio CF per cui ogni grammatica che lo

genera è ambigua è detto essere *inerentemente ambiguo*.

L'ambiguità di una grammatica si può risolvere introducendo dei **simboli** che servono a disambiguarla (e.g. le parentesi) o delle **regole di precedenza**

Automa a pila

Descriviamo una macchina astratta per il riconoscimento dei linguaggi CF. Questa macchina corrisponde ad un arricchimento degli automi già visti per il caso dei linguaggi regolari, mediante una struttura dati opportuna per memorizzare informazione: la pila.

Dal punto di vista architetturale, un automa a pila è una macchina costituita da un **controllo**, che è un automa a stati finiti (NFA o DFA), un **nastro di input**, che, come negli NFA e DFA, è di sola lettura e da una **pila**, sulla quale è possibile eseguire le seguenti operazioni:

- `push(X)` – Sposta una stringa X in testa alla pila
- `pop(X)` – Preleva il simbolo in testa alla pila
- `empty` – Controlla se la pila è vuota e restituisce un valore booleano

Le azioni possibili di un automa a pila sono:

Leggere dal nastro e dalla pila (pop) contemporaneamente

- Avanzare la testina a destra sul nastro.
- Cambiare stato nel controllo.
- Inserire una stringa (anche vuota) nella pila (push).

Leggere solo dalla pila (pop)

- Cambiare stato nel controllo.
- Inserire una stringa (anche vuota) nella pila (push).

Automa a Pila Non-Deterministico (APND)

Un automa a pila non-deterministico (APND) M è una 7-upla $M = \langle Q, \Sigma, R, \delta, q_0, Z_0, F \rangle$ dove:

- Q è un insieme finito di stati.
- Σ è un alfabeto finito (alfabeto del nastro).
- R è l'alfabeto finito della pila.
- $q_0 \in Q$ è lo stato iniziale.
- $Z_0 \in R$ è il simbolo iniziale sulla pila.

- $F \subseteq Q$ è l'insieme degli stati finali.
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times R \to \mathcal{P}_f(Q \times R^*)$ è la funzione di transizione.

L'evoluzione di un automa a pila è determinata dall'evoluzione delle sue **descrizioni istantanee**. Una descrizione istantanea rappresenta la "fotografia" dello stato (stato + nastro + pila) della macchina in un dato istante.

Una **descrizione istantanea** per un automa a pila non-deterministico (APND) è una tripla (q, x, γ) dove:

- $q \in Q$ è lo **stato corrente**.
- $x \in \Sigma^*$ è la **stringa ancora da leggere** sul nastro.
- $\gamma \in R^*$ è la **sequenza di simboli** contenuti sulla pila.

Se $(p, \gamma) \in \delta(q, a, Z)$, allora è definito un **passo di derivazione** dell'APND

$$(q, aw, Z\alpha) \to_M (p, w, \gamma\alpha)$$

Il linguaggio $L(M)$ riconosciuto da un APND M si può definire in due modi diversi

- **Per pila vuota** $L_p(M) = \{x \in \Sigma^* : (q_0, x, Z_0) \to_M^* (q, \epsilon, \epsilon), q \in Q\}$
- **Per stato finale** $L_F(M) = \{x \in \Sigma^* : (q_0, x, Z_0) \to_M^* (q, \epsilon, \gamma), \gamma \in R^*, q \in F\}$

Per ogni APND M, esiste un APND M_0 tale che $L_F(M) = L_p(M_0)$

...
...
...

Grammatiche Regolari

Una grammatica **context-free (CF)** si dice **lineare destra** se ogni produzione è della forma:

- $A \to wB$, con $w \in T^+$, oppure
- $A \to w$, con $w \in T^+$,

più eventualmente $S \to \epsilon$.

Se invece tutte le produzioni sono della forma:

- $A \to Bw$, con $w \in T^+$, oppure
- $A \to w$, con $w \in T^+$,

più eventualmente $S \to \epsilon$, si dice **lineare sinistra**.

Data una grammatica lineare destra G, esiste una grammatica G_0 equivalente che può essere trasformata nella **forma normale di Greibach**. In questa forma, ogni produzione deve essere della seguente forma:

- $A \to aB$, dove $a \in T$ (cioè, a è un simbolo terminale) e B è un simbolo non terminale, oppure
- $A \to a$, dove $a \in T$ (ossia, una produzione che termina direttamente con un simbolo terminale), con l'eventuale eccezione della produzione $S \to \varepsilon$, che può apparire solo per il simbolo iniziale S se la grammatica genera la stringa vuota.

Processo di conversione

1. **Rimozione di ricorsioni a sinistra**: Nella forma normale di Greibach, ogni produzione deve iniziare con un simbolo terminale, quindi si devono eliminare eventuali ricorsioni a sinistra.
2. **Normalizzazione delle produzioni**: Le produzioni della grammatica originale che non soddisfano la forma $A \to aB$ o $A \to a$ devono essere trasformate attraverso una serie di riscritture.
3. **Ciclo di riscrittura**: Si sostituiscono le produzioni che non rispettano le regole della GNF, finché tutte le produzioni sono nella forma corretta.

Alla fine di questo processo, si ottiene una grammatica G_0 in **forma normale di Greibach** con produzioni che seguono il formato indicato:

1. $A \to aB$, dove $a \in T$ e $B \in N$.
2. $A \to a$, dove $a \in T$.
3. $S \to \varepsilon$ se necessario.
