# Semplificazione di una CFG per ottenere la CNF

Nicola Gulmini

Sia $G = (V, T, P, S)$ una CFG. Vogliamo semplificarla. In ordine, eseguiamo le seguenti operazioni.

## 0.1 1. Eliminazione delle $\varepsilon$-produzioni

Supponiamo che $L(G) = L(G)/\{\varepsilon\} \neq \varnothing$. Se $\varepsilon \in L(G)$ vedremo alla fine come fare: tale procedura non considera il caso in cui la stringa vuota appartenga al linguaggio.

Per eliminare le epsilon produzioni, calcoliamo le variabili annullabili.

**Definizione 1** (Variabile annullabile)**.** $A \in V$ è annullabile se $A$ produce $\varepsilon$.

L'idea è la seguente: se $A$ è annullabile e $B \to CAD$, con $B, C, D \in V$, allora facciamo in modo che

- $B \to CD$ quando $A$ genera $\varepsilon$;
- $B \to CAD$ quando $A$ non genera $\varepsilon$, caso garantito eliminando la produzione $A \to \varepsilon$.

**Algoritmo 1** (Calcolo delle variabili annullabili)**.** Base: le variabili che producono $\varepsilon$ sono annullabili. Induzione: se $A \to B_1 B_2 \dots B_k$ con $B_1, \dots, B_k$ tutte annullabili, allora anche $A$ è annullabile.

Ora possiamo eliminare le epsilon produzioni.

**Algoritmo 2** (Eliminazione delle epsilon produzioni)**.** Togliamo tutte le produzioni di epsilon. Sia $A \to X_1 \dots X_{n \geq 1}$ una produzione della grammatica. Allora si considerano tutte le $2^n$ combinazioni possibili che si ottengono eliminando le $X_i$ annullabili tra quelle prodotte. L'unica eccezione è il caso in cui tutte le $X_i$ sono annullabili: non si considera la rimozione di tutte altrimenti si otterrebbe la produzione $A \to \varepsilon$, per cui le combinazioni diventano $2^n - 1$.

Notiamo che dopo aver eliminato le epsilon produzioni possiamo avere dei simboli *inutili*, che rimuoveremo con le prossime procedure.

**Esempio 1.** Sia $P$ l'insieme delle produzioni

$$A \to \varepsilon, B \to CDE, C \to A, E \to \varepsilon, D \to a \in T.$$

Al primo passo della procedura troviamo $A$ e $E$ annullabili perché producono $\varepsilon$. Poi troviamo anche $C$ perché produce una variabile annullabile. Ora eliminiamo $A \to \varepsilon$ e $E \to \varepsilon$, ottenendo

$$P = \{B \to CDE, C \to A, D \to a\}$$

notando che $C \to A$ va lasciata (anche se non porta a niente) mentre $B \to CDE$ va sostituita con le seguenti produzioni, ottenute rimuovendo le possibili combinazioni di variabili annullabili:

$$B \to CD|CE|C|CDE$$

che sono $2^m$ con $m = 2$ dato che i simboli annullabili nella produzione sono 2: $C, E$.

## 0.2 2. Eliminazione delle produzioni unitarie

Nell'esempio appena fatto rimane la produzione $C \to A$, inutile ai fini generativi della grammatica, il che è sufficiente per motivarne la rimozione. Questo spiega anche perché le operazioni vadano effettuate seguendo questo preciso ordine, dato che da ciascuna si possono produrre casi da trattare nella successiva.

**Definizione 2** (Produzione unitaria)**.** Siano $A, B \in V$ e $a \in T$. La produzione $A \to B$ è unitaria, mentre produzioni del tipo $A \to a$ o $A \to \varepsilon$ non lo sono.

Un buon effetto della la rimozione delle produzioni unitarie è anche l'eliminazione collaterale dell'ambiguità, come si può notare dall'albero di derivazione, per esempio, delle seguenti produzioni $\{A \to A|a\}$ e $\{A \to a\}$.

Ci sono due metodi per togliere le produzioni unitarie:

- **espansioni successive**, che non funziona nel caso di cicli, per esempio $\{A \to B, B \to C, C \to A\}$;
- **coppie unitarie**, che ora vediamo.

**Definizione 3** (Coppia unitaria)**.** Siano $A, B \in V$. $(A, B)$ è una coppia unitaria se $A$ produce $B$ in un numero arbitrario di sole produzioni unitarie: $A \to B_1, B_1 \to B_2, \ldots, B_n \to B$.

**Algoritmo 3** (Calcolo delle coppie unitarie)**.** Base: $A \in V$. $A$ produce $A$ in 0 passi, quindi $(A, A)$ è una coppia unitaria. Induzione: sia $(A, B)$ una coppia unitaria, con $B \in V$. Se $B \to C$ è una produzione, allora $(A, C)$ è una nuova coppia unitaria.

Ora possiamo usare le coppie unitarie trovate con questo algoritmo per rimuovere le produzioni unitarie.

**Algoritmo 4** (Eliminazione di produzioni unitarie)**.** Calcoliamo tutte le coppie unitarie con l'algoritmo appena visto. Siano del tipo $(A, B)$ le coppie trovate. Se $B \to a \in T^*$ è una produzione, allora consideriamo come produzione valida solo $A \to a$. Con $A = B$ stiamo considerando tutte le produzioni non unitarie.

## 0.3 3. Eliminazione dei simboli inutili

Siano $X \in V \cup T$, $\alpha, \beta \in (V \cup T)^*$, $S$ simbolo iniziale. Quando un simbolo è inutile?

**Definizione 4** (Simbolo raggiungibile)**.** $X$ è raggiungibile se esiste una derivazione che parte da $S$ e arriva a una forma sentenziale del tipo $\alpha X \beta$ per qualche $\alpha, \beta$.

**Definizione 5** (Simbolo generatore)**.** Un simbolo $X$ è generatore se esiste una derivazione che produce $w$ da $X$, per $w \in T^*$.

**Definizione 6** (Simbolo utile)**.** $X$ è utile $\iff$ è generatore e raggiungibile. In caso contrario è inutile.

Per eliminare i simboli inutili dobbiamo prima sapere quali sono utili, quindi calcolare generatori e raggiungibili.

**Algoritmo 5** (Calcolo dei simboli generatori)**.** Base: $a \in T$, allora $a$ produce $a$ in 0 passi. Induzione: se $A \to X_1 \ldots X_n$ tutti generatori, allora $A$ è generatore.

**Algoritmo 6** (Calcolo dei simboli raggiungibili)**.** Sia $A \to X_1 \ldots X_n$ una produzione. Se $A$ è raggiungibile, allora anche $X_1, \ldots, X_n$ lo sono, partendo da $S$ che lo è per definizione.

**Algoritmo 7** (Eliminazione dei simboli inutili)**.** In ordine:

1. calcoliamo i simboli generatori e raggiungibili;
2. eliminiamo le produzioni che coinvolgono simboli non generatori;
3. ripetiamo anche con i non raggiungibili.

In questo modo otteniamo una grammatica priva di simboli inutili.

## 0.4 4. CNF

Dopo aver

1. rimosso le epsilon produzioni, mediante i simboli annullabili;
2. rimosso le produzioni unitarie, mediante le coppie unitarie;
3. rimosso i simboli inutili, mediante generatori e raggiungibili;

*semplifichiamo* la grammatica, ma la sua forma non è ancora quella desiderata.

**Definizione 7** (Chomsky Normal Form: CNF)**.** Una CFG si trova in forma normale di Chomsky se le produzioni hanno la seguente forma:

- $A \to BC$, per $A, B, C \in V$;
- $A \to a$, per $A \in V$, $a \in T$;

ed è priva di simboli inutili.

Nota: possiamo sempre ottenere la CNF di $G$ se $\varepsilon \notin L(G)$, ma in caso contrario, si può considerare il linguaggio $L(G)/\{\varepsilon\}$ e alla fine inserire una produzione eccezionale del tipo $S \to \varepsilon$.

Nota: l'albero di derivazione ottenuto da una grammatica in CNF è binario, con i terminali alle foglie.

Cosa manca per ottenere la CNF a partire da una grammatica semplificata con gli algoritmi visti? I corpi possono essere di lunghezza maggiore di 2, quindi devono essere scomposti in catene di produzioni con corpo costituito di due sole variabili. Inoltre, tali corpi devono essere costituiti di sole variabili.

**Algoritmo 8** (CNF partendo da una grammatica semplificata)**.** Fissata la produzione $A \to \alpha \in (V \cup T)^*$ e $\|\alpha\| \geq 2$, se in $\alpha$ c'è un terminale va sostituito con un non-terminale, in questo modo:

$$A \to \alpha = X_1 X_2 \dots a \dots X_n, a \in T \Rightarrow \begin{cases} A \to X_1 X_2 \dots B \dots X_n \\ B \to a. \end{cases}$$

Se, invece, si ha $A \to B_1 B_2 \dots B_k$ con $k \geq 3$ (perché $k = 2$ va bene, $k = 1$ corrisponde a produzioni unitarie che sono state rimosse) va accorciata definendo $C_1, \dots, C_{k-2}$:

$$\begin{cases} A \to B_1 C_1 \\ C_1 \to B_2 C_2 \\ \vdots \\ C_{k-2} \to B_{k-1} B_k. \end{cases}$$