Per avere PDA più efficienti / più piccoli e con minor nondeterminismo è necessario

SEMPLIFICARE LE GRAMMATICHE

1) Eliminare le produzioni ε (del tipo $A \to \varepsilon$) che sono inadatte al bottom-up parsing

2) Eliminare le produzioni unitarie (del tipo $A \to B$) che possono creare dei cicli $A \stackrel{+}{\Rightarrow} A$

3) Eliminare simboli inutili, cioè quei terminali e nonterminali che non sono raggiungibili/generabili a partire dal simbolo iniziale S

4) Eliminare la ricorsione sinistra (del tipo $A \to A\alpha$), perché inadatta al top-down parsing

5) Fattorizzare la grammatica, per ottenere grammatiche con meno nondeterminismo nel top-down parsing.

Eliminare le produzioni ε

- Input: G libera con produzioni ε (del tipo $A \to \varepsilon$)
- Output: G' libera, senza produzioni ε, tale che

$$L(G') = L(G) \setminus \{\varepsilon\}$$

Oss: Se $\varepsilon \in L(G)$ e vogliamo una G'' tale che $L(G) = L(G'')$, basta considerare $G' = (NT, T, S, R')$ output dell'algoritmo

e definire $G'' = G' \cup \{S' \to \varepsilon \mid S\}$

$$G'' = (NT \cup \{S'\}, T, S', R' \cup \{S' \to \varepsilon \mid S\})$$

Simbolo annullabile: $A \in NT$ tale che $A \overset{+}{\Rightarrow} \varepsilon$

$N(G) = \{A \in NT \mid A \overset{+}{\Rightarrow} \varepsilon\}$ viene calcolato induttivamente come segue:

$$N_0(G) = \{A \in NT \mid A \to \varepsilon \in R\}$$

$$N_{i+1}(G) = N_i(G) \cup \{B \in NT \mid B \to C_1 \ldots C_k \in R \text{ e } C_1, \ldots, C_k \in N_i(G)\}$$

Oss:
- $N_i(G) \subseteq N_{i+1}(G)$ perché aggiungo qualcosa ad ogni passo
- $\exists i_c$ tale che $N_{i_c}(G) = N_{i_c+1}(G)$ perché NT è finito!

Fatto: L'insieme $N(G) = N_{i_c}(G)$ è esattamente l'insieme di tutti i simboli annullabili

Una volta calcolato $N(G)$ per $G=(NT, T, S, R)$, costruiamo la grammatica $G'=(NT, T, S, R')$ dove per ogni produzione $A \to \alpha \in R$, con $\alpha \neq \varepsilon$, in cui occorrono i simboli annullabili $Z_1, \ldots, Z_k$, mettiamo in R' tutte le produzioni del tipo $A \to \alpha'$ dove α' si ottiene da α cancellando tutti i possibili sottoinsiemi di $Z_1, \ldots, Z_k$ (incluso $\emptyset$), ad eccezione del caso in cui α' risulta ε.

- in G' non mettiamo produzioni $A \to \varepsilon \in R$
- in G' non introduciamo mai produzioni del tipo $A \to \varepsilon$

Proposizione Data una grammatica libera G, la grammatica G' determinata dall'algoritmo sopra non ha ε-produzioni e $L(G') = L(G) \setminus \{\varepsilon\}$

(Come già detto, se vogliamo che la nuova grammatica sia del tutto equivalente a G, allora bisogna ammettere una produzione ε per il nuovo simbolo iniziale:

$$G'' = (NT \cup \{S'\}, T, S', R' \cup \{S' \to \varepsilon \mid S\})$$

)

Esempio:

$$G = \begin{cases} S \to AB \\ A \to aAA \mid \varepsilon \\ B \to bBB \mid \varepsilon \end{cases}$$

$N_0(G) = \{A, B\}$
$N_1(G) = \{A, B, S\} = N(G)$

Consideriamo $S \to AB \in R$. Dobbiamo considerare 4 casi:

$\emptyset \Rightarrow S \to AB$
$\{B\} \Rightarrow S \to A$
$\{A\} \Rightarrow S \to B$
$\{A, B\} \Rightarrow \cancel{S \to \varepsilon}$ non la mettiamo in R'

$\Rightarrow S \to AB \mid A \mid B$

Ora consideriamo $A \to aAA \in R$. Dobbiamo considerare 4 casi:

$\emptyset \Rightarrow A \to aAA$
$\{A\} \Rightarrow A \to aA$ } duplicata
$\{A\} \Rightarrow A \to aA$
$\{A, A\} \Rightarrow A \to a$

$\Rightarrow A \to aAA \mid aA \mid a$

Ora consideriamo $B \to bBB$, e nello stesso modo di prima, otteniamo $B \to bBB \mid bB \mid b$

Le due ε-produzioni $A \to \varepsilon$ e $B \to \varepsilon$ sono ignorate.

$$\Rightarrow G' = \begin{cases} S \to AB \mid A \mid B \\ A \to aAA \mid aA \mid a \\ B \to bBB \mid bB \mid b \end{cases}$$

N.B. $\varepsilon \in L(G)$ $\quad S \Rightarrow AB \Rightarrow B \Rightarrow \varepsilon$
$\varepsilon \notin L(G')$

$$G'' = \begin{cases} S' \to \varepsilon \mid S \\ S \text{ come per } G' \end{cases}$$ è tale che $L(G) = L(G'')$

Eliminare le produzioni unitarie

- produzioni unitarie: $A \to B$ con $A, B \in NT$
- coppie unitarie: (A, B) tale che $A \stackrel{*}{\Rightarrow} B$ usando solo produzioni unitarie

Calcolare le coppie unitarie induttivamente

1) $U_0(G) = \{(A, A) \mid A \in NT\}$

2) $U_{i+1}(G) = U_i(G) \cup \{(A, C) \mid (A, B) \in U_i(G) \text{ e } B \to C \in R\}$

Oss: • $U_i(G) \subseteq U_{i+1}(G)$

• $\exists\, i_c$ tale che $U_{i_c}(G) = U_{i_c+1}(G)$ perché NT è finito

Per definizione, $U(G) = U_{i_c}(G)$, detto insieme di tutte le coppie unitarie.

Algoritmo: Data $G = (NT, T, R, S)$ libera, si definisce $G' = (NT, T, R', S)$ dove, per ogni $(A, B) \in U(G)$, R' contiene tutte le produzioni $A \to \alpha$, dove $B \to \alpha \in R$ e non è unitaria.

Oss: Poiché, per ogni $A \in NT$, la coppia $(A, A) \in U(G)$, R' contiene tutte le produzioni non-unitarie di R e in aggiunta un po' di altre.

Teorema

Sia $G = (NT, T, R, S)$ libera e sia $U(G)$ l'insieme delle sue coppie unitarie. Sia $G' = (NT, T, R', S)$ la grammatica ottenuta dall'algoritmo sopra. Allora G' non ha produzioni unitarie e $L(G) = L(G')$.

Esempio

$$\left.\begin{array}{l} E \to E+T \mid T \\ T \to T*A \mid A \\ A \to a \mid b \mid (E) \end{array}\right\} G$$

- gram. non ambigua per espr. aritmetiche
- però ha prod. unitarie $E \to T$ e $T \to A$

$U_0(G) = \{(E,E), (T,T), (A,A)\}$

$U_1(G) = U_0(G) \cup \{(E,T), (T,A)\}$

$U_2(G) = U_1(G) \cup \{(E,A)\} = U_3(G) = U(G)$

$$G' \left\{\begin{array}{l} E \to E+T \\ T \to T*A \\ A \to a \mid b \mid (E) \end{array}\right.$$

produzioni presenti grazie a $U_0(G)$

in aggiunta

$E \to T*A$ perché $(E,T) \in U_1(G)$

$T \to a \mid b \mid (E)$ perché $(T,A) \in U_1(G)$

$E \to a \mid b \mid (E)$ perché $(E,A) \in U_2(G)$

Quindi, riassumendo, G' è

$$E \to E+T \mid T*A \mid a \mid b \mid (E)$$
$$T \to T*A \mid a \mid b \mid (E)$$
$$A \to a \mid b \mid (E)$$

non contiene produzioni unitarie ed è equivalente a G

Rimuovere i simboli inutili

Def: Un simbolo $X \in T \cup NT$ è

- un generatore sse $\exists w \in T^*$ con $X \Rightarrow^* w$
- raggiungibile sse $S \Rightarrow^* \alpha X \beta$ per qualche $\alpha, \beta \in (T \cup NT)^*$
- utile sse è sia generatore, sia raggiungibile

  ovvero se $S \Rightarrow^* \alpha X \beta \Rightarrow^* z \in L(G)$, cioè X compare in almeno una derivazione di una stringa $z \in L(G)$.

Esempio

$$G \left[\begin{array}{l} S \rightarrow AB \mid a \\ B \rightarrow b \end{array} \right.$$

Poiché

$$\left.\begin{array}{l} a \Rightarrow^* a \\ b \Rightarrow^* b \\ S \Rightarrow^* a \\ B \Rightarrow^* b \end{array}\right\} \Rightarrow \text{generatori} = \{S, B, a, b\}$$

(manca A)

$\Downarrow$

possiamo eliminare tutte le produzioni che includono A

$$G' \left[\begin{array}{l} S \rightarrow a \\ B \rightarrow b \end{array} \right.$$

Ora B non è raggiungibile da S, allora elimino B

$$G'' \left[S \rightarrow a \right.$$

G'' è equivalente a G, ma non contiene simboli inutili

Come calcolare i generatori?

Ricorda: X è un generatore sse $\exists w \in T^*$. $X \Rightarrow^* w$

1) $G_0(G) = T$

se $a \in T$, $a \Rightarrow^* a$!
allora tutti i terminali sono generatori

2) $G_{i+1}(G) = G_i(G) \cup$

$$\{ B \in NT \mid B \to C_1 \dots C_k \in R \text{ e } C_1, \dots, C_k \in G_i(G) \}$$

Oss: Nel passo 2) è contemplato anche il caso $K=0$ (cioè $B \to \varepsilon \in R$): se $B \to \varepsilon$, allora B è un generatore.

Oss:
- $G_i(G) \subseteq G_{i+1}(G)$
- $\exists i_c$ tale che $G_{i_c}(G) = G_{i_c+1}(G)$ perché $T \cup NT$ è finito

Fatto L'insieme $G(G) = G_{i_c}(G)$ è esattamente l'insieme di tutti i simboli generatori di G.

Come calcolare i raggiungibili?

Ricordo: X è raggiungibile sse
$$S \Rightarrow^* \alpha X \beta \quad \text{per qualche } \alpha, \beta \in (N \cup T)^*$$

1) $R_0(G) = \{S\}$

2) $R_{i+1}(G) = R_i(G) \cup \bigcup_{B \in R_i(G),\ B \to X_1 \dots X_k \in R} \{X_1, \dots, X_k\}$

Oss: • $R_i(G) \subseteq R_{i+1}(G)$

• $\exists i_c .\ R_{i_c}(G) = R_{i_c+1}(G)$ perché $T \cup N$ è finito

Fatto L'insieme $R(G) = R_{i_c}(G)$ è esattamente l'insieme di tutti i simboli raggiungibili di G.

Come rimuovere i simboli inutili?

1) Prima elimino tutti i non-generatori (e tutte le produzioni che usano almeno uno di questi)

2) Poi, dalla nuova grammatica, elimino tutti i non-raggiungibili (e tutte le prod. che li usano)

$\Rightarrow$ La grammatica risultante è equivalente all'originale e non contiene simboli inutili

Eliminare i simboli inutili

Teorema Sia $G = (NT, T, R, S)$ una grammatica libera tale che $L(G) \neq \emptyset$.

1) Sia G_1 la grammatica che si ottiene da G eliminando tutti i simboli che non appartengono a $G(G)$, e tutte le produzioni che fanno uso di almeno uno di tali simboli. (*)

2) Sia G_2 la grammatica che si ottiene da G_1 eliminando tutti i simboli che non appartengono a $R(G_1)$, e tutte le produzioni che fanno uso di almeno uno di tali simboli.

Allora G_2 non ha simboli inutili e $L(G_2) = L(G)$.

Dimostrazione

(*) Poiché $L(G) \neq \emptyset$, allora S è un generatore e così rimarrà in G_1, e poi in G_2

- $L(G_2) \subseteq L(G)$ è ovvio, perché G_2 contiene meno produzioni di G.
- $L(G) \subseteq L(G_2)$: dobbiamo dimostrare che se $S \overset{*}{\underset{G}{\Rightarrow}} w$ allora $S \overset{*}{\underset{G_2}{\Rightarrow}} w$. Ma ogni simbolo usato nella derivazione $S \overset{*}{\underset{G}{\Rightarrow}} w$ è ovviamente sia raggiungibile, sia generatore!! Allora quella derivazione è anche una derivazione per G_2.

L'ordine dei 2 passi è importante!

1) prima elimino i non-generatori
2) poi i non-raggiungibili.

Ma se inverto l'ordine, allora può capitare che non elimino tutti i simboli inutili!!

Esempio

$S \to AB \mid a$
$B \to b$

1) Elimino i non-generatori

$S \to a$
$B \to b$

2) Elimino i non-raggiungibili

$S \to a$

Corretto!
tutti i simboli inutili sono stati rimossi

2) Elimino i non-raggiungibili

$S \to AB \mid a$
$B \to b$

Tutti ragg.

1) Elimino i non generatori

$S \to a$
$B \to b$

è rimasto un simbolo inutile: B e la prod. $B \to b$

Esempio

$$G \left[\begin{array}{l} S \to AB \mid aC \\ A \to a \\ B \to bB \\ C \to b \mid AC \\ D \to a \mid aS \end{array} \right.$$

$G(G) = \{a, b, A, C, D, S\}$

solo B non è generatore

$\Downarrow$

$$G_1 \left[\begin{array}{l} S \to aC \\ A \to a \\ C \to b \mid AC \\ D \to a \mid aS \end{array} \right.$$

$R(G_1) = \{S, a, C, b, A\}$

solo D non è raggiungibile

$\Downarrow$

$$\left. \begin{array}{l} S \to aC \\ A \to a \\ C \to b \mid AC \end{array} \right] G_2$$

G_2 è la grammatica semplificata, equivalente a G, senza simboli inutili.

$L(G_2) = \{ab, aab, aaab, \ldots\} = a^+ b$

È possibile trovare una grammatica più semplice di G_2 per il ling. $a^+ b$?

$$S \to aS \mid ab$$

Mettere le cose insieme...

Se, nel semplificare la grammatica G, seguiamo questo ordine:

1) eliminare le ε-produzioni
2) eliminare le produzioni unitarie (e quindi i cicli)
3) eliminare i simboli inutili

allora la grammatica risultante è garantita non avere né ε-produzioni, né produzioni unitarie, né simboli inutili, ed è equivalente a quella di partenza!

N.B. L'ordine è importante poiché alcune delle costruzioni possono interagire tra loro!

Ad es: durante la fase di eliminazione delle ε-produzioni, potremmo introdurre prod. unitarie!

$\implies$ le ε-produzioni vanno eliminate prima di procedere all'eliminazione delle produzioni unitarie

Esempio

$$G\left[\begin{array}{l} S \to aAa \\ A \to C \\ C \to S \mid \varepsilon \end{array}\right.$$

1) Togliere ε-produzioni

$N(G) = \{C, A\}$

$\Downarrow$

$$G' \left[\begin{array}{l} S \to aAa \mid aa \\ A \to C \\ C \to S \end{array}\right.$$

- togo $C \to \varepsilon$
- non metto $A \to \varepsilon$

2) Togliere le prod. unitarie

$U(G') = \{(A,A), (C,C), (S,S), (A,C), (C,S), (A,S)\}$

$$G'' \left[\begin{array}{ll} S \to aAa \mid aa & \text{perché } (S,S) \in U(G') \\ C \to aAa \mid aa & \text{perché } (C,S) \in U(G') \\ A \to aAa \mid aa & \text{perché } (A,S) \in U(G') \end{array}\right.$$

3) Rimuovere i simboli inutili

$G(G'') = \{S, a, A, C\}$ tutti generatori

$R(G'') = \{S, a, A\}$ ma non C!

$\Downarrow$

$$\left.\begin{array}{l} S \to aAa \mid aa \\ A \to aAa \mid aa \end{array}\right] G'''$$

È possibile trovare una grammatica ancora più semplice per $L(G''') = (aa)^+$?

$S \to aSa \mid aa$ o anche

$S \to aaS \mid aa$

Forme Normali

- di Chomsky
- di Greibach

garantiscono proprietà particolari alle grammatiche

Forma normale di Chomsky

Tutte le sue produzioni sono della forma

$$A \to BC$$
$$A \to a$$

(con ε trattato a parte: $S \to \varepsilon \mid BC \dots$)
(S non compare mai a dx in una produzione)

Oss: se G libera è in forma normale di Chomsky, allora
- non ha ε-produzioni
- non ha produzioni unitarie

Oss: Ogni grammatica libera G può essere trasformata in una equivalente G' in forma normale di Chomsky.

Forma Normale di Greibach

Tutte le produzioni sono della forma

$$A \to aBC$$
$$A \to aB$$
$$A \to a$$

(con ε trattato a parte, come sopra)

Oss:
- no ε-produzioni / no prod. unitarie
- non è mai ricorsiva sx!
- ogni prod., applicata in una derivazione, allunga il prefisso di terminali $\Rightarrow$ parser costruito a partire da f.n. di Greibach sono meno nondet.

Oss: Ogni G libera può essere trasformata in una equiv. G' f.n. Greibach

Eliminare la ricorsione sinistra

(problema per parser top-down)

- produzione ricorsiva sx: $A \to A\alpha \in R$
- G è ricorsiva sx: $A \stackrel{+}{\Rightarrow} A\alpha$ per qualche $A \in NT$ e $\alpha \in (T \cup NT)^*$

Come rimuove la ricorsione sx immediata?

$$A \to A\alpha_1 | \dots | A\alpha_n | \beta_1 | \dots | \beta_m$$

(le stringhe β_i non cominciano per A)

Queste produzioni possono essere rimpiazzate da

$$A \to \beta_1 A' | \dots | \beta_m A'$$

$$A' \to \alpha_1 A' | \dots | \alpha_n A' | \varepsilon$$

Se nella gr. originale abbiamo la derivazione

$$A \Rightarrow A\alpha_{i_1} \Rightarrow A\alpha_{i_2}\alpha_{i_1} \Rightarrow \dots \Rightarrow A\alpha_{i_k}\dots\alpha_{i_2}\alpha_{i_1} \Rightarrow \beta_i\alpha_{i_k}\dots\alpha_{i_2}\alpha_{i_1}$$

allora nella nuova grammatica avremo

$$A \Rightarrow \beta_i A' \Rightarrow \beta_i \alpha_{i_k} A' \Rightarrow \dots \Rightarrow \beta_i\alpha_{i_k}\dots\alpha_{i_2}A' \Rightarrow \beta_i\alpha_{i_k}\dots\alpha_{i_2}\alpha_{i_1}A'$$
$$\Rightarrow \beta_i\alpha_{i_k}\dots\alpha_{i_2}\alpha_{i_1}$$

e viceversa

Esempio

$$A \to Aa \mid b$$

$$A \to bA'$$
$$A' \to aA' \mid \varepsilon$$

Esempio

$$A \to Ab \mid Ac \mid d$$

$$A \to dA'$$
$$A' \to bA' \mid cA' \mid \varepsilon$$

Oss: Se $G = [A \to Aa$, non si può applicare l'algoritmo, perché mancano le produzioni di base da cui partire $(A \to \beta_1 \mid .. \mid \beta_m)$

Infatti, $L(G) = \emptyset$ e la grammatica corrispondente non ha produzioni.

Ricorsione sx non-immediata

Consideriamo

$$\left.\begin{array}{l} S \to Ba \mid b \\ B \to Bc \mid Sc \mid d \end{array}\right] G$$

In G c'è ricorsione sx immediata ($B \to Bc$), ma anche non-immediata ($S \Rightarrow Ba \Rightarrow Sca$).

Algoritmo: Input: G libera senza ε-produzioni, senza prod. unitarie, ma con ricorsione sx non-immediata

Output: G' libera, senza ricorsione sx, ma può avere ε-produzioni

Sia $NT = \{A_1, A_2, .., A_n\}$ in un ordine fissato

For i from 1 to n {

1) For j from 1 to $i-1$ {

sostituisci ogni produzione della forma $A_i \to A_j \alpha$ con le produzioni

$A_i \to \beta_1 \alpha \mid \dots \mid \beta_k \alpha$ dove $A_j \to \beta_1 \mid .. \mid \beta_k$ sono le produzioni correnti per A_j

}

2) Elimina la ricorsione immediata su A_i

}

$$S \to Ba \mid b$$
$$B \to Bc \mid Sc \mid d \qquad NT = \{\underset{1}{S}, \underset{2}{B}\}$$

- Per $i=1$ (cioè per S), il ciclo interno (J da 1 a $\emptyset$) non viene eseguito; siccome non c'è ricorsione immediata per S, non faccio niente!

- Per $i=2$ (cioè $A_i = B$), il ciclo interno (J da 1 a 1) si esegue solo per $A_J = A_1 = S$.
  Allora la produzione $B \to Sc$ viene rimpiazzata con
  $$B \to Bac \mid bc$$
  Ora le produzioni complessive per B sono
  $$B \to Bc \mid Bac \mid bc \mid d$$
  dalle quali dobbiamo eliminare la ricorsione immediata!
  Il risultato è
  $$B \to bcB' \mid dB'$$
  $$B' \to cB' \mid acB' \mid \varepsilon$$
  ovvero la grammatica risultante è
  $$S \to Ba \mid b$$
  $$B \to bcB' \mid dB'$$
  $$B' \to cB' \mid acB' \mid \varepsilon$$

Esercizio

$$\left.\begin{aligned} S &\to AB \mid D \\ A &\to \varepsilon \mid Ab \mid aB \\ B &\to b \mid bB \\ C &\to c \\ D &\to aD \end{aligned}\right] G$$

1) Togliere simboli inutili
2) Togliere ricorsione sx
3) Togliere ε-produzioni
4) Togliere prod. unitarie

1) D non generatore
C non raggiungibile $\Longrightarrow$

$$\left.\begin{aligned} S &\to AB \\ A &\to \varepsilon \mid Ab \mid aB \\ B &\to b \mid bB \end{aligned}\right] G'$$

2) Ricorsione sx

$(A \to Ab)$ $\Longrightarrow$

$$\left.\begin{aligned} S &\to AB \\ A &\to A' \mid aBA' \\ A' &\to bA' \mid \varepsilon \end{aligned}\right] G''$$
$$B \to b \mid bB$$

3) Togliere le ε-prod.

$N(G'') = \{A, A'\}$ $\Longrightarrow$

$$\left.\begin{aligned} S &\to AB \mid B \\ A &\to A' \mid aBA' \mid aB \\ A' &\to bA' \mid b \\ B &\to b \mid bB \end{aligned}\right] G'''$$

4) Togliere le prod. unitarie

$U(G''') = Id \cup \{(S,B), (A,A')\}$

$\Longrightarrow$

$$\begin{aligned} S &\to AB \mid \underbrace{b \mid bB} \\ A &\to \underbrace{bA' \mid b} \mid aBA' \mid aB \\ A' &\to bA' \mid b \\ B &\to b \mid bB \end{aligned}$$

Fattorizzazione a Sinistra

(importante per top-down parsing)

$A \to aBbC \mid aBd$

Se, in top-down parsing, sulla pila ho A e leggo in input a, non sono in grado di determinare quale produzione scegliere (nondeterminismo)

$\Rightarrow$ raccolgo la parte comune ("aB") alle 2 produzioni e introduco un nuovo nonterminale per rappresentare il "resto"

$$A \to aBA'$$
$$A' \to bC \mid d$$

Algoritmo generale

- Inizializza $N = NT$
- Ripeti il ciclo presente finché nessuna modifica è più possibile a N o all'insieme delle produzioni {

  for each $A \in N$ {

  Sia α il prefisso più lungo comune alle parti destre di alcune produzioni di A;

  if $\alpha \neq \varepsilon$ {

  - sia A' un nuovo nonterminale; $N := N \cup \{A'\}$;
  - rimpiazza tutte le produzioni per A

  $$A \to \alpha\beta_1 \mid .. \mid \alpha\beta_k \mid \gamma_1 \mid .. \mid \gamma_h$$

  con le produzioni

  $$A \to \alpha A' \mid \gamma_1 \mid .. \mid \gamma_h$$
  $$A' \to \beta_1 \mid .. \mid \beta_k$$

  }

  }

  }

Esempio

$$
\left.\begin{aligned}
E &\to T \mid T+E \mid T-E \\
T &\to A \mid A * T
\end{aligned}\right\}
$$
si può fattorizzare!

$$A \to a \mid b \mid (E)$$

$$
\left.\begin{aligned}
E &\to T E' \\
E' &\to \varepsilon \mid +E \mid -E \\
T &\to A T' \\
T' &\to \varepsilon \mid *T \\
A &\to a \mid b \mid (E)
\end{aligned}\right\}
$$
Versione equivalente fattorizzata