Théorie des Langages
Épisode 4 — Langages Hors-contexte

Thomas Pietrzak

Université Paul Verlaine — Metz

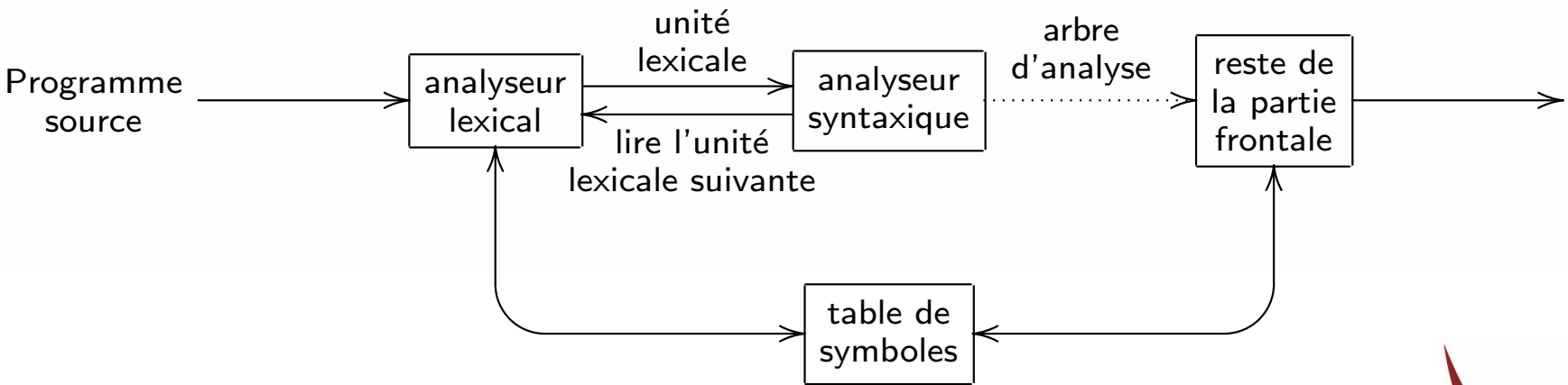
La syntaxe des constructions d'un langage de programmation peut être décrite par des grammaires hors contexte, ou notation **BNF**. Les grammaires hors contexte offrent des avantages significatifs à la fois aux concepteurs de langage de programmation et aux constructeurs de compilateurs.

- Une grammaire hors contexte (**GHC**) donne une spécification syntaxique précise et cependant facile à comprendre d'un langage de programmation.
- Pour certaines classes de **GHC** il est possible de construire automatiquement un analyseur syntaxique efficace.
- Les **GHC** imposent au langage de programmation une structure qui est utile pour la traduction du programme source en du code objet correct et pour la détection d'erreurs.

- Les langages évoluent au cours du temps, acquérant de nouvelles constructions en effectuant des tâches additionnelles. Ces constructions nouvelles peuvent être ajoutées à un langage de programmation facilement.
- Les méthodes pour l'analyse syntaxique sont les méthodes les plus sophistiquées utilisées en compilation.
- Le choix d'une grammaire pour la description d'un langage de programmation doit être fait entre le pouvoir de l'expression ou description, et l'efficacité d'analyse syntaxique.

Interaction entre l'analyseur lexical et l'analyseur syntaxique

- Dans le modèle de compilateur ci-dessous, l'analyseur syntaxique obtient une chaîne d'unités lexicales de l'analyseur lexicale, vérifie que la chaîne peut être engendrée par la grammaire hors contexte du langage source.
- On demande que l'analyseur syntaxique signale chaque erreur de syntaxe de façon intelligible.
- Il doit également supporter les erreurs les plus communes de façon à pouvoir continuer le traitement du texte restant.



Dérivations

- Soit $m_0 \Rightarrow_G m_1 \Rightarrow_G \cdots \Rightarrow_G m_r$ une dérivation au rapport de la grammaire hors contexte $G = (N, T, S, P)$.
- Une dérivation est dite gauche (resp. droite) si à à chaque étape $m_i \Rightarrow_G m_{i+1}, i \in \{0, 1, \ldots, r-1\}$, uniquement le non terminal le plus à gauche (resp. droite) de m_i est remplacé.
- Chaque étape $m_i \Rightarrow_G m_{i+1}$ d'une dérivation gauche peut s'écrire $wA\gamma \Rightarrow_G wa\gamma$ où $w \in T^*$, $(A \rightarrow a) \in P$, et $\gamma \in (N \cup T)^*$.
- Chaque étape $m_i \Rightarrow_G m_{i+1}$ d'une dérivation droite peut s'écrire $\gamma Aw \Rightarrow_G \gamma aw$ où $w \in T^*$, $(A \rightarrow a) \in P$, et $\gamma \in (N \cup T)^*$.
- On note la dérivation gauche $\alpha \Rightarrow_g^* \beta$.
- On note la dérivation droite $\alpha \Rightarrow_d^* \beta$.

proto-phrases

Soit $G = (N, T, S, P)$ une grammaire hors contexte.

- Si $S \Rightarrow_g^* \delta$, on dit que $\delta \in \{N \cup T\}^*$ est une proto-phrase de G.
- δ peut contenir certains non-terminaux.
- On dit que $\delta \in \{N \cup T\}^*$ est une proto-phrase gauche si $S \Rightarrow_g^* \delta$.
- On dit que $\delta \in \{N \cup T\}^*$ est une proto-phrase droite si $S \Rightarrow_d^* \delta$.

Exemple

$E \rightarrow EAE \mid (E) \mid -E \mid id$
$A \rightarrow + \mid - \mid * \mid /$

- $E \Rightarrow EAE \Rightarrow E/E \Rightarrow -E/E \Rightarrow -id/E \Rightarrow -id/id$
- $E \Rightarrow_g EAE \Rightarrow_g -EAE \Rightarrow_g -idAE \Rightarrow_g -id/E \Rightarrow_g -id/id$
- $E \Rightarrow_d EAE \Rightarrow_d EAid \Rightarrow_d E/id \Rightarrow_d -E/id \Rightarrow_d -id/id$

Arbres abstraits syntaxiques

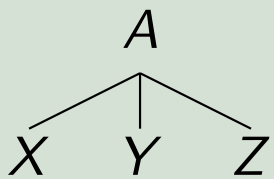
- La tâche principale de l'analyse syntaxique est la construction d'un arbre abstrait syntaxique.
- C'est une représentation particulière d'une dérivation (de la partie principale) du programme source obtenu comme chaîne d'unités lexicales d'analyseur lexical.
- On peut transformer une dérivation facilement dans un arbre abstrait syntaxique.
- À chaque arbre on peut associer une dérivation gauche ou droite unique.

Arbre abstrait syntaxique

Un arbre syntaxique illustre visuellement la manière dont l'axiome d'une grammaire se dérive en une chaîne du langage.

Exemple

Si le non terminal A est défini par la production $A \rightarrow XYZ$, alors un arbre syntaxique peut posséder un noeud intérieur étiqueté A et avoir trois fils étiquetés X, Y et Z, de gauche à droite :



Formellement, étant donné une grammaire hors contexte, un arbre syntaxique est un arbre possédant les propriétés suivantes :

- La racine est étiquetée par l'axiome.
- Chaque feuille est étiquetée par une unité lexicale ou par ϵ.
- Chaque noeud intérieur est étiqueté par un non-terminal
- Si A est le non-terminal étiquetant un noeud intérieur et si les étiquettes des fils de ce noeud sont, de gauche à droite, $X_1, X_2, \ldots, X_n$ alors $A \rightarrow X_1 X_2 \ldots X_n$ est une production. Ici $X_1, X_2, \ldots, X_n$ représentent soit un non-terminal, soir un terminal. Un cas particulier est $A \rightarrow \epsilon$ qui signifie qu'un noeud étiqueté A a un seul fils ϵ.

- Les feuilles d'un arbre syntaxique, lues de gauche à droite, constituent le mot de feuilles, qui est la chaîne engendrée ou dérivée à partir du non-terminal situé à la racine de l'arbre syntaxique.
- Chaque arbre transmet un ordre naturel de gauche à droite à ses feuilles qui s'appuie sur l'idée que si a et b sont deux fils d'un même père, alors tous les descendants de a sont à gauche des descendants de b.
- On peut donner une autre définition du langage engendré par une grammaire : c'est l'ensemble des chaînes qui peuvent être engendrées par un arbre syntaxique quelconque.
- Le processus consistant à trouver un arbre syntaxique correspondant à une chaîne donnée d'unités lexicales s'appelle l'analyse de cette chaîne.

Ambiguïté

On doit être prudent en parlant de la structure d'une chaîne selon une grammaire. En effet, s'il est clair que chaque arbre syntaxique dérive exactement la chaîne constituant son mot des feuilles, une grammaire peut avoir plus d'un arbre syntaxique qui engendrent une chaîne donnée d'unités lexicales.

Une telle grammaire est dite ambigüe. Pour montrer qu'une grammaire est ambigüe, il nous suffit de trouver une chaîne d'unités lexicales qui a plus d'un arbre syntaxique.

Comme une telle chaîne a habituellement plus d'une signification, on a besoin, dans un contexte de compilation, de travailler avec des grammaires non ambigües, ou d'utiliser des grammaires ambigües avec des règles traditionnelles pour résoudre les ambigüités.

Exemple 1

Les listes de chiffres séparés par des signes $+$ ou $-$:

$\mathit{liste} \longrightarrow \mathit{liste} + \mathit{chiffre}$

$\mathit{liste} \longrightarrow \mathit{liste} - \mathit{chiffre}$

$\mathit{liste} \longrightarrow \mathit{chiffre}$

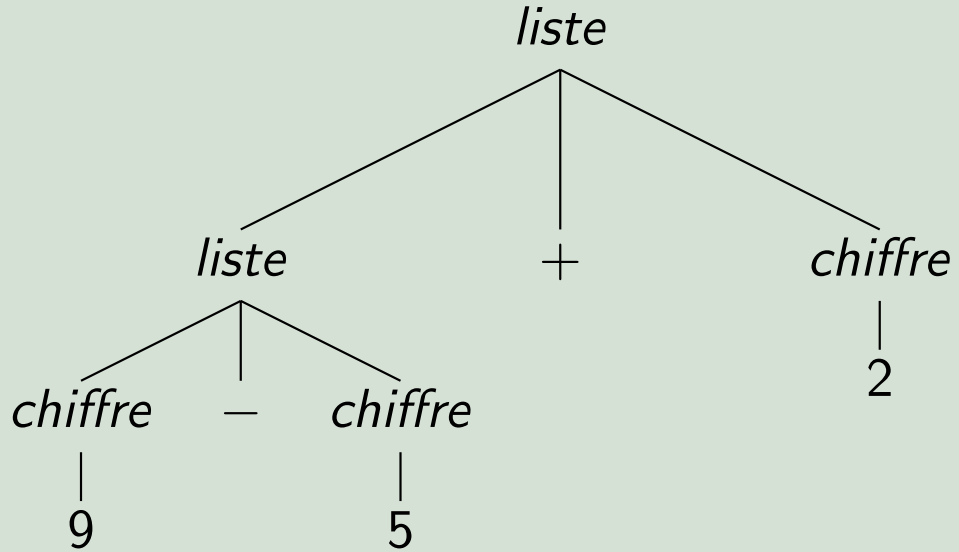
$\mathit{chiffre} \longrightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

ou

$\mathit{liste} \longrightarrow \mathit{liste} + \mathit{chiffre} \mid \mathit{liste} - \mathit{chiffre} \mid \mathit{chiffre}$

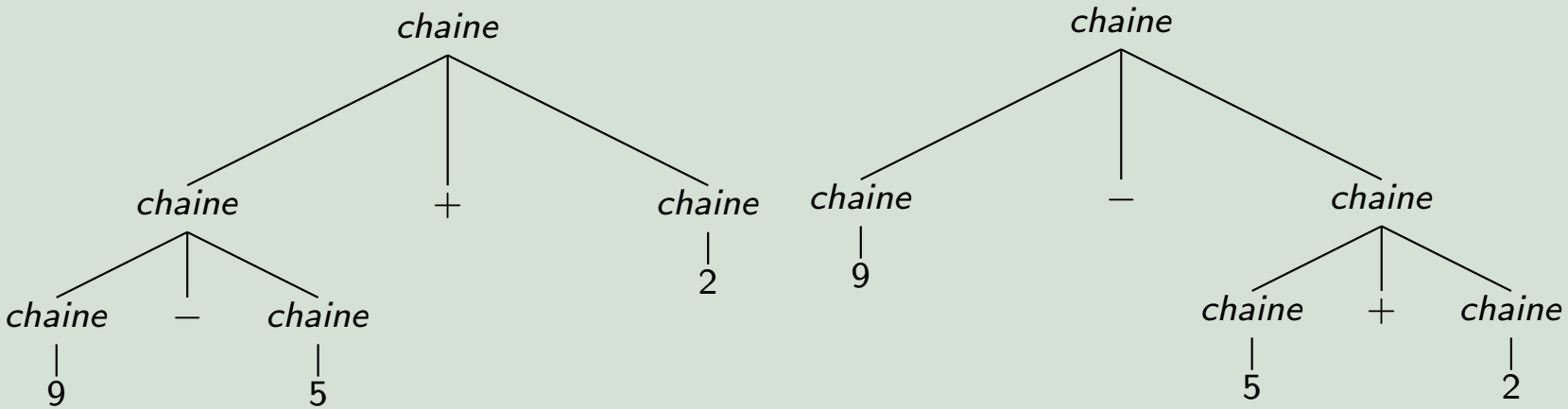
$\mathit{chiffre} \longrightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Arbre syntaxique pour $9 - 5 + 2$:



Exemple 2

Supposons que l'on ne distingue pas les listes des chiffres comme dans l'exemple précédent. On peut écrire la grammaire suivante :
$chaine \longrightarrow chaine + chaine \mid chaine - chaine \mid [0-9]$
Confondre les notions de *chiffre* et de *liste* dans la seule notion *chaine* donne un sens superficiel car un simple *chiffre* est un cas particulier de *liste*.
Ces arbres syntaxiques pour $9-5+2$ montrent que cette grammaire est ambigüe :



Les deux arbres pour $9-5+2$ correspondent aux deux parenthésages de l'expression : $(9-5)+2$ et $9-(5+2)$. La grammaire de l'exemple précédent ne permettait pas cette interprétation.

Élimination des symboles inutiles

La simplification des grammaires hors contexte est importante pour l'analyse syntaxique.

Utile

Soit $G = (N, T, S, P)$ une **GHC**. Un symbole $X \in N \cup T$ est dit utile s'il y a un mot $m \in L(G)$ et est une dérivation $S \Rightarrow_G^* \alpha X \beta \Rightarrow_G^* m$, où $\alpha, \beta \in (N \cup T)^*$. Sinon X est appelé inutile.

Pour chaque grammaire hors contexte il existe une gramaire hors contexte équivalente sans symboles inutiles. On distingue les symboles inutiles en deus sous-catégories : les improductifs et les inaccessibles.

Improductif

$A \in N$ est dit improductif s'il n'y a pas de mot $m \in L(G)$ tel que $A \Rightarrow_G^* m$.

Élimination des symboles inutiles

L'algorithme suivant enlève tous les non terminaux improductifs :

Supposons que $L(G) \neq \emptyset$

- $productifs := \emptyset$
- Répéter ceci jusqu'à ce que *productifs* ne change plus :
  - Pour chaque production $X \longrightarrow p$ telle que $p \in (T \cup productifs)^*$
    - $productifs := productifs \cup \{X\}$
- Supprimer tous les $A \in (N \setminus productifs)$ et toutes les productions qui contiennent un tel $A \in (N \setminus productifs)$ de G.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow eE \mid D$
$E \rightarrow D$

$productifs = \emptyset$

- $A \rightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \rightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \rightarrow b$ convient car $b \in T$
- $C \rightarrow c$ convient car $c \in T$
- $D \rightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow eE \mid D$
$E \rightarrow D$

$productifs = \emptyset$

- $A \rightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \rightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \rightarrow b$ convient car $b \in T$
- $C \rightarrow c$ convient car $c \in T$
- $D \rightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \emptyset$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \longrightarrow b$ convient car $b \in T$
- $C \longrightarrow c$ convient car $c \in T$
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{B\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \longrightarrow b$ convient car $b \in T$
- $C \longrightarrow c$ convient car $c \in T$
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \longrightarrow b$ convient car $b \in T$
- $C \longrightarrow c$ convient car $c \in T$
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow eE \mid D$
$E \rightarrow D$

$productifs = \{B, C\}$

- $A \rightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \rightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \rightarrow b$ convient car $b \in T$
- $C \rightarrow c$ convient car $c \in T$
- $D \rightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow eE \mid D$
$E \rightarrow D$

$productifs = \{B, C\}$

- $A \rightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \rightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \rightarrow b$ convient car $b \in T$
- $C \rightarrow c$ convient car $c \in T$
- $D \rightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow eE \mid D$
$E \rightarrow D$

$productifs = \{B, C\}$

- $A \rightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \rightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \rightarrow b$ convient car $b \in T$
- $C \rightarrow c$ convient car $c \in T$
- $D \rightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow eE \mid D$
$E \rightarrow D$

$productifs = \{B, C\}$

- $A \rightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \rightarrow B$ ne convient pas car $B \notin (T \cup productifs)$
- $B \rightarrow b$ convient car $b \in T$
- $C \rightarrow c$ convient car $c \in T$
- $D \rightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \rightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ ne convient pas car $A \notin (T \cup productifs)$
- $A \longrightarrow B$ convient car $B \in productifs$
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On a modifié *productifs* donc on boucle.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$
$D \longrightarrow eE \mid D$
$E \longrightarrow D$

$productifs = \{A, B, C\}$

- $A \longrightarrow aA$ convient toujours mais a déjà été traité
- $A \longrightarrow B$ convient toujours mais a déjà été traité
- $B \longrightarrow b$ convient toujours mais a déjà été traité
- $C \longrightarrow c$ convient toujours mais a déjà été traité
- $D \longrightarrow eE$ ne convient pas car $E \notin (T \cup productifs)$
- $D \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- $E \longrightarrow D$ ne convient pas car $D \notin (T \cup productifs)$
- On n'a pas modifié *productifs* cette fois-ci.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow eE \mid D$
$E \rightarrow D$

$productifs = \{A, B, C\}$
$D \in N \setminus productifs$ et $E \in N \setminus productifs$ donc ces deux non-terminaux sont supprimés de la grammaires, ainsi que les productions qui lui sont associées. On obtient la grammaire simplifiée suivante :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$

Inaccessible

$X \in N \cup T$ est dit inaccessible s'il n'y a pas de $\alpha, \beta \in (N \cup T)^*$ tels que $S \Rightarrow_G^* \alpha X \beta$.

Élimination des symboles inaccessibles

Soit $G' = (N', T, S, P)$ la grammaire obtenue par la première étape.
L'algorithme suivant enlève tous inaccessibles :

- *accessibles* $:= \{S\}$
- Répéter ceci jusqu'à ce que *accessibles* ne change plus :
  - Pour chaque production $X \to p$ de G' tel que $X \in$ *accessibles*
    - *accessibles* $:=$ *accessibles* $\cup \{Y \in (N' \cup T) \mid p \text{ contient } Y\}$
- Supprimer tous les $B \in ((N' \cup T) \setminus \textit{accessibles})$ et les productions qui contiennent un tel $B \in ((N' \cup T) \setminus \textit{accessibles})$ de G'.

Exemple

Prenons la grammaire simplifiée précédemment :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$

$accessibles = \{A\}$

- $A \longrightarrow aA$: ajout de $a \in T$
- $A \longrightarrow B$: ajout de $B \in N$
- $B \longrightarrow b$: ajout de $b \in T$
- On ne peut plus rien ajouter

La production $C \longrightarrow c$ doit être supprimée. On obtient la grammaire simplifiée :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$

Exemple

Prenons la grammaire simplifiée précédemment :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$

$accessibles = \{A, a\}$

- $A \rightarrow aA$: ajout de $a \in T$
- $A \rightarrow B$: ajout de $B \in N$
- $B \rightarrow b$: ajout de $b \in T$
- On ne peut plus rien ajouter

La production $C \rightarrow c$ doit être supprimée. On obtient la grammaire simplifiée :
$A \rightarrow aA \mid B$
$B \rightarrow b$

Exemple

Prenons la grammaire simplifiée précédemment :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$

$productifs = \{A, a, B\}$

- $A \longrightarrow aA$: ajout de $a \in T$
- $A \longrightarrow B$: ajout de $B \in N$
- $B \longrightarrow b$: ajout de $b \in T$
- On ne peut plus rien ajouter

La production $C \longrightarrow c$ doit être supprimée. On obtient la grammaire simplifiée :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$

Exemple

Prenons la grammaire simplifiée précédemment :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$

$productifs = \{A, a, B, b\}$

- $A \longrightarrow aA$: ajout de $a \in T$
- $A \longrightarrow B$: ajout de $B \in N$
- $B \longrightarrow b$: ajout de $b \in T$
- On ne peut plus rien ajouter

La production $C \longrightarrow c$ doit être supprimée. On obtient la grammaire simplifiée :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$

Exemple

Prenons la grammaire simplifiée précédemment :
$A \rightarrow aA \mid B$
$B \rightarrow b$
$C \rightarrow c$

$productifs = \{A, a, B, b\}$

- $A \rightarrow aA$: ajout de $a \in T$
- $A \rightarrow B$: ajout de $B \in N$
- $B \rightarrow b$: ajout de $b \in T$
- On ne peut plus rien ajouter

La production $C \rightarrow c$ doit être supprimée. On obtient la grammaire simplifiée :
$A \rightarrow aA \mid B$
$B \rightarrow b$

Exemple

Prenons la grammaire simplifiée précédemment :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$
$C \longrightarrow c$

$productifs = \{A, a, B, b\}$

- $A \longrightarrow aA$: ajout de $a \in T$
- $A \longrightarrow B$: ajout de $B \in N$
- $B \longrightarrow b$: ajout de $b \in T$
- On ne peut plus rien ajouter

La production $C \longrightarrow c$ doit être supprimée. On obtient la grammaire simplifiée :
$A \longrightarrow aA \mid B$
$B \longrightarrow b$

ϵ-productions

Une ϵ-production est une production du type $A \rightarrow \epsilon$.

Élimination des ϵ-productions

L'algorithme suivant calcule tous les non-terminaux $A \in N$ tels que $A \Rightarrow_G^* \epsilon$.

- $vides := \{X \in N \mid X \rightarrow \epsilon\}$
- Répéter ceci jusqu'à ce que *vides* ne change plus :
  - $vides := vides \cup \{X \in N \mid \exists X \rightarrow p \in P \text{ où } p \in vides^*\}$
- Supprimer toutes les productions $X \rightarrow \epsilon$ où $X \in vides$.
- Remplacer chaque production $A \rightarrow p$ de P par toutes les productions $A \rightarrow p'$, où on peut obtenir p' à partir de p en enlevant un sous-ensemble quelconque de non-terminaux $B \in vides$ dans p.

Conséquence : si G est une **GHC**, alors G' est une **GHC**.

Exemple

Prenons la grammaire :

$A \rightarrow aA \mid aB \mid a$

$B \rightarrow b \mid CC \mid \epsilon \mid C$

$C \rightarrow c \mid BB \mid B$

$vides = \{B\}$

- On peut ajouter C car $C \rightarrow BB$ et $BB \in vides^*$.
- On ne peut plus rien ajouter.
- On supprime la production $B \rightarrow \epsilon$.
- On remplace la production $A \rightarrow aB$ par $A \rightarrow aB \mid a$.
- On remplace la production $B \rightarrow CC$ par $B \rightarrow CC \mid C$.
- On remplace la production $C \rightarrow BB$ par $C \rightarrow BB \mid B$.

Exemple

Prenons la grammaire :

$A \rightarrow aA \mid aB \mid a$

$B \rightarrow b \mid CC \mid \epsilon \mid C$

$C \rightarrow c \mid BB \mid B$

$vides = \{B, C\}$

- On peut ajouter C car $C \rightarrow BB$ et $BB \in vides^*$.
- On ne peut plus rien ajouter.
- On supprime la production $B \rightarrow \epsilon$.
- On remplace la production $A \rightarrow aB$ par $A \rightarrow aB \mid a$.
- On remplace la production $B \rightarrow CC$ par $B \rightarrow CC \mid C$.
- On remplace la production $C \rightarrow BB$ par $C \rightarrow BB \mid B$.

Exemple

Prenons la grammaire :

$A \rightarrow aA \mid aB \mid a$

$B \rightarrow b \mid CC \mid \epsilon \mid C$

$C \rightarrow c \mid BB \mid B$

$vides = \{B, C\}$

- On peut ajouter C car $C \rightarrow BB$ et $BB \in vides^*$.
- On ne peut plus rien ajouter.
- On supprime la production $B \rightarrow \epsilon$.
- On remplace la production $A \rightarrow aB$ par $A \rightarrow aB \mid a$.
- On remplace la production $B \rightarrow CC$ par $B \rightarrow CC \mid C$.
- On remplace la production $C \rightarrow BB$ par $C \rightarrow BB \mid B$.

Exemple

Prenons la grammaire :

$A \rightarrow aA \mid aB \mid a$

$B \rightarrow b \mid CC \mid \epsilon \mid C$

$C \rightarrow c \mid BB \mid B$

$vides = \{B, C\}$

- On peut ajouter C car $C \rightarrow BB$ et $BB \in vides^*$.
- On ne peut plus rien ajouter.
- On supprime la production $B \rightarrow \epsilon$.
- On remplace la production $A \rightarrow aB$ par $A \rightarrow aB \mid a$.
- On remplace la production $B \rightarrow CC$ par $B \rightarrow CC \mid C$.
- On remplace la production $C \rightarrow BB$ par $C \rightarrow BB \mid B$.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid aB \mid a$
$B \rightarrow b \mid CC \mid \epsilon \mid C$
$C \rightarrow c \mid BB \mid B$

$vides = \{B, C\}$

- On peut ajouter C car $C \rightarrow BB$ et $BB \in vides^*$.
- On ne peut plus rien ajouter.
- On supprime la production $B \rightarrow \epsilon$.
- On remplace la production $A \rightarrow aB$ par $A \rightarrow aB \mid a$.
- On remplace la production $B \rightarrow CC$ par $B \rightarrow CC \mid C$.
- On remplace la production $C \rightarrow BB$ par $C \rightarrow BB \mid B$.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid aB \mid a$
$B \rightarrow b \mid CC \mid \epsilon \mid C$
$C \rightarrow c \mid BB \mid B$

$vides = \{B, C\}$

- On peut ajouter C car $C \rightarrow BB$ et $BB \in vides^*$.
- On ne peut plus rien ajouter.
- On supprime la production $B \rightarrow \epsilon$.
- On remplace la production $A \rightarrow aB$ par $A \rightarrow aB \mid a$.
- On remplace la production $B \rightarrow CC$ par $B \rightarrow CC \mid C$.
- On remplace la production $C \rightarrow BB$ par $C \rightarrow BB \mid B$.

Exemple

Prenons la grammaire :
$A \rightarrow aA \mid aB \mid a$
$B \rightarrow b \mid CC \mid \epsilon \mid C$
$C \rightarrow c \mid BB \mid B$

$vides = \{B, C\}$

- On peut ajouter C car $C \rightarrow BB$ et $BB \in vides^*$.
- On ne peut plus rien ajouter.
- On supprime la production $B \rightarrow \epsilon$.
- On remplace la production $A \rightarrow aB$ par $A \rightarrow aB \mid a$.
- On remplace la production $B \rightarrow CC$ par $B \rightarrow CC \mid C$.
- On remplace la production $C \rightarrow BB$ par $C \rightarrow BB \mid B$.

Production simple

Une production simple est une production du type $A \to B$ où $A, B \in N$.

Élimination des productions simples

Pour chaque non-terminal $A \in N$, on calcule tous les $B \in N$ tels que $A \Rightarrow_G^* B$.
On utilise un graphe orienté $\tilde{G} = (\tilde{V}, \tilde{E})$ où $(A, B) \in \tilde{E}$ si et seulement si $A \to B$ est une production simple de G. Donc $A \Rightarrow_G^* B$ si et seulement si il y a un chemin orienté de A à B dans le graphe G.
On obtient une grammaire G' sans productions simples satisfaisant $L(G) = L(G')$ grâce à cet algorithme :

- soit P' le sous-ensemble des productions non simples de P.
- $P'' := \emptyset$
- Pour chaque couple $(A, B) \in \tilde{V}^2$ où $A \Rightarrow_G^* B$:
  - Ajouter $A \to \beta$ à P'' pour chaque $(B \to \beta) \in P$ avec $\beta \notin \tilde{V}$.

Si G est une **GHC**, alors $G' = (N, T, S, P' \cup P'')$ est une **GHC**.

Exemple

Prenons la grammaire :

$A \to aA \mid B \mid cC \mid d$

$B \to C \mid cC \mid d$

$C \to cC \mid d$

On construit le graphe $\tilde{G} = (\tilde{V}, \tilde{E})$:

- $\tilde{V} = \{A, B, C\}$
- $\tilde{E} = \{(A, B), (B, C)\}$

$P' = \{(A, aA), (C, cC), (C, d)\}$

$P'' = \emptyset$

- Ajoutons $A \to cC$ car $A \Rightarrow_G^* C$ et $C \to cC$
- Ajoutons $A \to d$ car $A \Rightarrow_G^* C$ et $C \to d$
- Ajoutons $B \to cC$ car $B \Rightarrow_G^* C$ et $C \to cC$
- Ajoutons $B \to d$ car $B \Rightarrow_G^* C$ et $C \to cC$

Remarque : il y a des symboles non atteignables. On peut les supprimer.

Exemple

Prenons la grammaire :

$A \longrightarrow aA \mid B \mid cC \mid d$

$B \longrightarrow C \mid cC \mid d$

$C \longrightarrow cC \mid d$

On construit le graphe $\tilde{G} = (\tilde{V}, \tilde{E})$:

- $\tilde{V} = \{A, B, C\}$
- $\tilde{E} = \{(A, B), (B, C)\}$

$P' = \{(A, aA), (C, cC), (C, d)\}$

$P'' = \{(A, cC)\}$

- Ajoutons $A \longrightarrow cC$ car $A \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $A \longrightarrow d$ car $A \Rightarrow_G^* C$ et $C \longrightarrow d$
- Ajoutons $B \longrightarrow cC$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $B \longrightarrow d$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$

Remarque : il y a des symboles non atteignables. On peut les supprimer.

Exemple

Prenons la grammaire :

$A \longrightarrow aA \mid B \mid cC \mid d$

$B \longrightarrow C \mid cC \mid d$

$C \longrightarrow cC \mid d$

On construit le graphe $\tilde{G} = (\tilde{V}, \tilde{E})$:

- $\tilde{V} = \{A, B, C\}$
- $\tilde{E} = \{(A, B), (B, C)\}$

$P' = \{(A, aA), (C, cC), (C, d)\}$

$P'' = \{(A, cC), (A, d)\}$

- Ajoutons $A \longrightarrow cC$ car $A \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $A \longrightarrow d$ car $A \Rightarrow_G^* C$ et $C \longrightarrow d$
- Ajoutons $B \longrightarrow cC$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $B \longrightarrow d$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$

Remarque : il y a des symboles non atteignables. On peut les supprimer.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B \mid cC \mid d$
$B \longrightarrow C \mid cC \mid d$
$C \longrightarrow cC \mid d$

On construit le graphe $\tilde{G} = (\tilde{V}, \tilde{E})$:

- $\tilde{V} = \{A, B, C\}$
- $\tilde{E} = \{(A, B), (B, C)\}$

$P' = \{(A, aA), (C, cC), (C, d)\}$
$P'' = \{(A, cC), (A, d), (B, cC)\}$

- Ajoutons $A \longrightarrow cC$ car $A \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $A \longrightarrow d$ car $A \Rightarrow_G^* C$ et $C \longrightarrow d$
- Ajoutons $B \longrightarrow cC$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $B \longrightarrow d$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$

Remarque : il y a des symboles non atteignables. On peut les supprimer.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B \mid cC \mid d$
$B \longrightarrow C \mid cC \mid d$
$C \longrightarrow cC \mid d$

On construit le graphe $\tilde{G} = (\tilde{V}, \tilde{E})$:

- $\tilde{V} = \{A, B, C\}$
- $\tilde{E} = \{(A, B), (B, C)\}$

$P' = \{(A, aA), (C, cC), (C, d)\}$
$P'' = \{(A, cC), (A, d), (B, cC), (B, d)\}$

- Ajoutons $A \longrightarrow cC$ car $A \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $A \longrightarrow d$ car $A \Rightarrow_G^* C$ et $C \longrightarrow d$
- Ajoutons $B \longrightarrow cC$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $B \longrightarrow d$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$

Remarque : il y a des symboles non atteignables. On peut les supprimer.

Exemple

Prenons la grammaire :
$A \longrightarrow aA \mid B \mid cC \mid d$
$B \longrightarrow C \mid cC \mid d$
$C \longrightarrow cC \mid d$

On construit le graphe $\tilde{G} = (\tilde{V}, \tilde{E})$:

- $\tilde{V} = \{A, B, C\}$
- $\tilde{E} = \{(A, B), (B, C)\}$

$P' = \{(A, aA), (C, cC), (C, d)\}$
$P'' = \{(A, cC), (A, d), (B, cC), (B, d)\}$

- Ajoutons $A \longrightarrow cC$ car $A \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $A \longrightarrow d$ car $A \Rightarrow_G^* C$ et $C \longrightarrow d$
- Ajoutons $B \longrightarrow cC$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$
- Ajoutons $B \longrightarrow d$ car $B \Rightarrow_G^* C$ et $C \longrightarrow cC$

Remarque : il y a des symboles non atteignables. On peut les supprimer.

Corollaire

Pour chaque langage hors contexte L où $\epsilon \notin L$, il y a une grammaire hors contexte G sans symboles inutiles, sans ϵ-productions et sans productions simples telle que $L(G) = L$.

1. Élimination des ϵ-productions
2. Élimination des productions simples
3. Élimination des symboles inutiles

Forme normale de Chomsky

Une grammaire hors contexte $G = (N, T, S, P)$ est en forme normale de Chomsky si toutes les productions de P sont de type $A \rightarrow BC$ ou $A \rightarrow a$, où $A, B, C \in N$ et $a \in T$.

Forme normale de Greibach

Une grammaire hors contexte $G = (N, T, S, P)$ est en forme normale de Greibach si toutes les productions de P sont de type $A \rightarrow a\alpha$ où $A \in N$, $a \in T$ et $\alpha in N^*$.

Théorème

Soit $L \neq \emptyset$ un langage hors contexte tel que $\epsilon \notin L$. Alors, il y a une grammaire en forme normale de Chomsky G' et une grammaire en forme normale de Greibach G'' telles que $L(G') = L(G'') = L$ sans symboles inutiles.

Lemme de pompage (pour les grammaires hors contexte)

Soit $L \subseteq T^*$ un langage hors contexte. Alors il existe une constante $n \in \mathbb{N}$ telle que pour tout $m \in L$ de longueur au moins n, il existe $u, v, w, x, y \in T^*$ satisfaisant :

1. $m = uvwxy$
2. $|vwx| \leq n$
3. $vx \neq \epsilon$
4. $uv^i wx^i y \in L \; \forall i \in \mathbb{N}$

Démonstration

Soit G une grammaire hors-contexte pour un langage $L = L(G)$. Supposons L infini (tout langage fini est hors-contexte), et que $\epsilon \notin L$ (il suffit de vérifier pour $L \setminus \{\epsilon\}$). Supposons que la grammaire est sans productions inutiles, sans productions unitaires et sans ϵ-productions (on la mettra en forme normale de Chomski par exemple).

- Le seul moyen de produire des mots de taille arbitraire est d'avoir un non-terminal récursif.
- En effet si on n'a pas de non terminal non-récursif, on ne peut produire que des chaînes de longueur fixée à l'avance.
- En partant de S, on peut réduire vers une suite de terminaux et de non terminaux (mais pas S).
- A chaque non-terminal, on réduit en des terminaux et des non-terminaux pas encore utilisés.
- Comme le nombre de non-terminaux est fini, la longueur des chaînes productibles est donc finie.

Démonstration (suite)

- Soit $m \in L$ plus grand que tous les mots générables sans récursivité. Sa dérivation contient donc un non-terminal récursif R. Il y a donc une réduction du type : $S \Rightarrow^* uRy$, avec u et y des sous-chaînes de L.
- Puisque R est utilisé récursivement, il y a une chaîne de dérivation de type $R \Rightarrow^* vRx$ et donc $S \Rightarrow^* uRy \Rightarrow^* uvRxy$.
- Comme le mot appartient au langage, on arrive à une chaîne de terminaux : $S \Rightarrow^* uRy \Rightarrow^* uvRxy \Rightarrow^* uvwxy$, avec $m = uvwxy$.
- Le mot étant fini, il y a un nombre fini de réductions récursives. Il existe donc un entier n tel que $|vwx| \leq n$ si on a choisi R comme étant le dernier non terminal récursif réduit dans la dérivation de vRx. Cet entier peut être par exemple la longueur du plus grand mot générable sans récursivité. Pour produire un mot plus grand, il faut utiliser une règle récursive et s'étendre sur v et x.
- Comme la grammaire ne contient pas de productions unitaires et de ϵ-productions par hypothèse, chaque étape de dérivation introduit au moins un terminal et allonge donc la phrase dérivée. Comme $R \Rightarrow^* vRx$, on a donc $vx \neq \epsilon$.
- Des dérivations $S \Rightarrow^* uvRxy$, $R \rightarrow^* vRx$ et $R \rightarrow^* w$, on peut obtenir $S \Rightarrow^* uv^iwx^iy \ \forall i \in \mathbb{N}$, donc $uv^iwx^iy \in L \ \forall i \in \mathbb{N}$

Exemple

Considérons le langage abstrait $L = \{wcw \mid w \in (a \mid b)^*\}$. L est constitué de tous les mots composés par deux chaînes de a et de b, séparées par un c. Par exemple *abacaba*.
Montrons par l'absurde que ce langage n'est pas hors contexte. En supposant que ce langage est hors contexte, il existe un $n \in \mathbb{N}$ tel que pour tout $m \in L$ de longueur au moins n on ait les propriétés précédemment citées.
Prenons $m = a^n b^n c a^n b^n$. On a bien $|m| > n$.

- Soient $u, v, w, x, y \in T^*$ tels que $m = vwxy$
- D'après la propriété 2 on a $|vwx| \leq n$, donc les cas suivants sont possibles :
  - $vwx = a^i$ avec $i \leq n$. Or dans ce cas
    $uv^i wx^i y = uwy = a^j b^n c a^n b^n (j < n \text{ car } vx \neq \epsilon) \notin L$.
  - $vwx = a^i b^j$ avec $i + j \leq n$. Or dans ce cas
    $uwy = a^k b^l c a^n b^n (k, l < n \text{ car } vx \neq \epsilon) \notin L$.
  - $vwx = b^i$ avec $i \leq n$. Or dans ce cas
    $uwy = a^n b^j c a^n b^n (j < n \text{ car } vx \neq \epsilon) \notin L$.
  - $vwx = b^i c a^j$ avec $i + j \leq n$. Or dans ce cas
    $uwy = a^n b^k c ? a^l b^n (k, l < n \text{ car } vx \neq \epsilon) \notin L$.
  - $vwx = a^i$ avec $i \leq n$. Or dans ce cas
    $uv^i wx^i y = uwy = a^n b^n c a^j b^n (j < n \text{ car } vx \neq \epsilon) \notin L$.
  - $vwx = a^i b^j$ avec $i + j \leq n$. Or dans ce cas
    $uwy = a^n b^n c a^k b^l (k, l < n \text{ car } vx \neq \epsilon) \notin L$.
  - $vwx = b^i$ avec $i \leq n$. Or dans ce cas
    $uwy = a^n b^n c a^n b^j (j < n \text{ car } vx \neq \epsilon) \notin L$.

- Le langage de l'exemple met en avant le problème du contrôle que dans dans un programme les identificateurs sont déclarés avant d'être utilisés.
- Le premier w de wcw représente la déclaration d'un identificateur w.
- Le second w représente son utilisation.
- La dépendance de L vis à vis du contexte implique directement la dépendance vis-à-vis du contexte des langages de programmation comme Pascal ou C, qui nécessitent la déclaration des identificateurs avant leur utilisation etqui permettent des identificateurs de longueur arbitraire.
- Pour cette raison, une grammaire décrivant la syntaxe de Pascal ou C ne spécifie pas les caractères dans un identificateur. À la place, tous les identificateurs sont représentés dans la grammaire par une unité lexicale telle que id.
- Dans un compilateur pour un langage de ce genre, c'est la phase d'analyse sémantique qui contrôle que les identificateurs ont été déclarés avant d'être utilisés.