LF – Théorie des langages formels

Sylvain Brandel

2025 – 2026

sylvain.brandel@univ-lyon1.fr

Partie 3

LANGAGES ALGÉBRIQUES

Grammaires

Analyse ascendante

Automates à pile, algébricité

Grammaires

- Exemple
  - $L = a(ab \cup ba)^*b$.
  - Mots générés : un *a* suivi de *ab* ou *ba* un certain nombre de fois suivi d'un *b*
  - Un mot de L peut naturellement être décomposé en un début, un milieu et une fin

$$S \to aE$$
$$E \to abE$$
$$E \to baE$$
$$E \to b$$

  - Génération

$$S \to aE \to aabE \to aabbaE \to aabbab$$
$$S \to aE \to abaE \to ababaE \to ababaabE \to ababaabb$$

Grammaires

- Grammaire algébrique, ou hors-contexte (ang. *Context-free*)
  - Un ensemble de symboles terminaux à partir desquels sont construits les mots du langage (*a* et *b* dans l'exemple)
  - Un ensemble de symboles non terminaux (S et E dans l'exemple) parmi lesquels on distingue un symbole particulier (souvent S pour *Start*)
  - Un ensemble fini de règles ou production de la forme

    symbole non terminal $\rightarrow$ suite finie de symboles terminaux et / ou non terminaux

- Grammaire contextuelle (ang. *Context-sensitive*)
  - Dans les règles, le symbole non terminal est entouré de deux mots appelés le contexte

- Grammaire générale
  - Pas de restriction sur les règles

Grammaires

- Grammaire algébrique : quadruplet $G = (V, \Sigma, R, S)$ où
  - Σ est un ensemble fini de symboles terminaux appelé alphabet
  - V est un ensemble fini de symboles non terminaux tels que $V \cap \Sigma = \emptyset$
  - $S \in V$ est le symbole initial
  - $R \subset V \times (V \cup \Sigma)^*$ de la forme $A \rightarrow w$

Les éléments de R sont appelés règles ou productions

- Grammaire contextuelle
  - $R \subset (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ de la forme $uAv \rightarrow uwv$

- Grammaire générale
  - $R \subset (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ de la forme $z \rightarrow w$

Grammaires algébriques

- Dérivation directe

  Soient u et v $\in (V \cup \Sigma)^*$

  On dit que v dérive directement de u, et on note $u \Rightarrow_G v$,

  ssi $\exists$ x, y, w $\in (V \cup \Sigma)^*$, $\exists A \in V$ tels que

  $u = xAy$ et $v = xwy$ et $A \rightarrow w \in R$

- Exemple

  En utilisant la grammaire G définie par les règles suivantes :

  $S \rightarrow aE$

  $E \rightarrow abE \mid baE \mid b$

  on obtient $aabE \Rightarrow_G aabbaE$ par application de la règle $E \rightarrow baE$

- Dérivation

  La relation $\Rightarrow_G^*$ est la fermeture réflexive transitive de la relation $\Rightarrow_G$

Grammaires algébriques

- Dérivation

Soient u et v $\in (V \cup \Sigma)^*$

On dit que v dérive de u, et on note $u \Rightarrow_G^* v$

ssi $\exists\, w_0, \ldots, w_n \in (V \cup \Sigma)^*$ tels que

$u = w_0$ et $v = w_n$ et $w_i \Rightarrow_G w_{i+1}\ \forall\, i < n$

La suite $w_0 \Rightarrow_G w_1 \Rightarrow_G \ldots \Rightarrow_G w_n$ est appelée une dérivation

(lorsqu'il n'y a pas d'ambiguïté, on peut omettre la référence à la grammaire G dans les symboles $\Rightarrow_G$ et $\Rightarrow_G^*$)

La valeur de n ($n \geq 0$) est la longueur de la dérivation

Grammaires algébriques

- Langage engendré

  Soit $G = (V, \Sigma, R, S)$ une grammaire algébrique

  Le langage engendré par G, noté L(G), est :

  $$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow_G^* w \}$$

- Langage algébrique
  - Un langage est dit algébrique s'il peut être engendré par une grammaire algébrique

Grammaires algébriques

- Exemple

  G = (V, Σ, R, Phrase) avec :

  – V = { Phrase, Sujet, Verbe, Complément, Article, Nom, Adjectif }

  – Σ = { il, elle, nous, je, mange, voit, suis, le, la, les, frites, montagne, bleue, dorées, déprimé, " " }

  – R = { Phrase $\rightarrow$ Sujet Verbe Complément,

    Sujet $\rightarrow$ il | elle | nous | je,

    Verbe $\rightarrow$ mange | voit | suis,

    Complément $\rightarrow$ Article Nom Adjectif,

    Article $\rightarrow$ le | la | les,

    Nom $\rightarrow$ frites | montagne,

    Adjectif $\rightarrow$ bleue | dorées | déprimé }

Grammaires algébriques

- Exemple

  G = (V, Σ, R, Paragraphe) avec :
  - V = { Paragraphe, Phrase, Sujet, Verbe, Complément, Article, Nom, Adjectif }
  - Σ = { il, elle, nous, je, mange, voit, suis, le, la, les, frites, montagne, bleue, dorées, déprimé, . }
  - R = { Paragraphe → Phrase Paragraphe | Phrase,
    Phrase → Sujet Verbe Complément.,
    Sujet → il | elle | nous | je,
    Verbe → mange | voit | suis,
    Complément → Article Nom Adjectif,
    Article → le | la | les,
    Nom → frites | montagne,
    Adjectif → bleue | dorées | déprimé }

Grammaires algébriques

- Exemple

  $G = (V, \Sigma, R, \text{Chapitre})$ avec :

  – V = { Chapitre, Paragraphe, Phrase, Sujet, Verbe, Complément, Article, Nom, Adjectif }

  – Σ = { il, elle, nous, je, mange, voit, suis, le, la, les, frites, montagne, bleue, dorées, déprimé, **.** }

  – R = { Chapitre $\rightarrow$ Paragraphe Chapitre | Paragraphe,
  Paragraphe $\rightarrow$ Phrase Paragraphe | Phrase,
  Phrase $\rightarrow$ Sujet Verbe Complément**.**,
  Sujet $\rightarrow$ il | elle | nous | je,
  Verbe $\rightarrow$ mange | voit | suis,
  Complément $\rightarrow$ Article Nom Adjectif,
  Article $\rightarrow$ le | la | les,
  Nom $\rightarrow$ frites | montagne,
  Adjectif $\rightarrow$ bleue | dorées | déprimé }

Grammaires algébriques

- Exemple

  $G = (V, \Sigma, R, \text{Livre})$ avec :

  - V = { Livre, Chapitre, Paragraphe, Phrase, Sujet, Verbe, Complément, Article, Nom, Adjectif }
  - Σ = { il, elle, nous, je, mange, voit, suis, le, la, les, frites, montagne, bleue, dorées, déprimé, **.** }
  - R = { Livre $\to$ Chapitre Livre | Chapitre,

    Chapitre $\to$ Paragraphe Chapitre | Paragraphe,

    Paragraphe $\to$ Phrase Paragraphe | Phrase,

    Phrase $\to$ Sujet Verbe Complément**.**,

    Sujet $\to$ il | elle | nous | je,

    Verbe $\to$ mange | voit | suis,

    Complément $\to$ Article Nom Adjectif,

    Article $\to$ le | la | les,

    Nom $\to$ frites | montagne,

    Adjectif $\to$ bleue | dorées | déprimé }

Grammaires algébriques

- Exemple

  $G = (V, \Sigma, R, S)$ avec :
  - $V = \{ S, E \}$
  - $\Sigma = \{ a, b \}$
  - $R = \{ S \rightarrow EE,\ E \rightarrow EEE \mid bE \mid Eb \mid a \}$

- La chaîne ababaa peut être dérivée de plusieurs façons :

(a)	(b)	(c)	(d)
$S \Rightarrow EE$	$S \Rightarrow EE$	$S \Rightarrow EE$	$S \Rightarrow EE$
$\Rightarrow EEEE$	$\Rightarrow aE$	$\Rightarrow Ea$	$\Rightarrow aE$
$\Rightarrow aEEE$	$\Rightarrow aEEE$	$\Rightarrow EEEa$	$\Rightarrow aEEE$
$\Rightarrow abEEE$	$\Rightarrow abEEE$	$\Rightarrow EEbEa$	$\Rightarrow aEEa$
$\Rightarrow abaEE$	$\Rightarrow abaEE$	$\Rightarrow EEbaa$	$\Rightarrow abEEa$
$\Rightarrow ababEE$	$\Rightarrow ababEE$	$\Rightarrow EbEbaa$	$\Rightarrow abEbEa$
$\Rightarrow ababaE$	$\Rightarrow ababaE$	$\Rightarrow Ebabaa$	$\Rightarrow ababEa$
$\Rightarrow ababaa$	$\Rightarrow ababaa$	$\Rightarrow ababaa$	$\Rightarrow ababaa$

Grammaires

- Types de dérivations
  - (a) et (b) : chaque étape de la dérivation consiste à transformer le non-terminal le plus à gauche. On appelle ce genre de dérivation une dérivation la-plus-à-gauche (ang. *left-most derivation*)
  - (c) : dérivation la-plus-à-droite (ang. *right-most derivation*) où le symbole transformé à chaque étape est le non-terminal le plus à droite
  - (d) : dérivation ni plus-à-droite, ni plus-à-gauche

- Une suite de dérivations peut être visualisée par un arbre de dérivation ou arbre syntaxique (ang. *parse tree*)
  - Un tel arbre indique quelles sont les règles appliquées aux non-terminaux
  - Il n'indique pas l'ordre d'application des règles
  - Les feuilles de l'arbre représentent la chaîne dérivée

Grammaires algébriques

- Exemple
  G = (V, Σ, R, E) avec :
  - V = { E, T, F }
  - Σ = {x, y, +, $\times$, (,) }
  - R = {E $\rightarrow$ E + T | T, T $\rightarrow$ T $\times$ F | F, F $\rightarrow$ (E) | x | y }

- Chaîne $(x + y \times y) \times (x + y)$ avec la dérivation la plus-à-gauche :

$$
\begin{aligned}
E &\Rightarrow T \\
&\Rightarrow T \times F \\
&\Rightarrow F \times F \\
&\Rightarrow (E) \times F \\
&\Rightarrow (E + T) \times F \\
&\Rightarrow (T + T) \times F \\
&\Rightarrow (F + T) \times F \\
&\Rightarrow (x + T) \times F \\
&\Rightarrow (x + T \times F) \times F \\
&\Rightarrow (x + F \times F) \times F \\
&\Rightarrow (x + y \times F) \times F \\
&\Rightarrow (x + y \times y) \times F \\
&\Rightarrow (x + y \times y) \times (E) \\
&\Rightarrow (x + y \times y) \times (E + T) \\
&\Rightarrow (x + y \times y) \times (T + T) \\
&\Rightarrow (x + y \times y) \times (F + T) \\
&\Rightarrow (x + y \times y) \times (x + T) \\
&\Rightarrow (x + y \times y) \times (x + F) \\
&\Rightarrow (x + y \times y) \times (x + y)
\end{aligned}
$$

Grammaires

- Grammaire ambiguë
  - Lorsqu'une grammaire peut produire plusieurs arbres distincts associés à un même mot, ont dit que la grammaire est ambiguë

- Grammaires équivalentes
  - Deux grammaires qui engendrent le même langage sont dites équivalentes

Langages rationnels et grammaires algébriques

- Il existe des langages algébriques qui ne sont pas rationnels

cf. preuves de *non-rationalité*

- Grammaire linéaire à droite
  - $G = (V, \Sigma, R, S)$ telle que $R \subseteq V \times \Sigma^*(V \cup \{\varepsilon\})$

  (rappel : dans une grammaire algébrique (non régulière),$R \subseteq V \times (V \cup \Sigma)^*$)

- Grammaire linéaire à gauche
  - $G = (V, \Sigma, R, S)$ telle que $R \subseteq V \times (V \cup \{\varepsilon\})\ \Sigma^*$

- Grammaire régulière
  - Grammaire linéaire à droite ou linéaire à gauche

Hiérarchie de Chomsky

- Type 3
  - Grammaires régulières — Langages réguliers
  - Automates à états finis
- Type 2
  - Grammaires algébriques — Langages algébriques
  - Automates à pile
- Type 1
  - Grammaires contextuelles
  - Machines de Turing à mémoire linéairement bornée
- Type 0
  - Grammaires générales — Langages récursivement énumérables
  - Machines de Turing
- Inclusion
  - $T3 \subset T2 \subset T1 \subset T0$

Langages rationnels et grammaires algébriques

- Exemple

  $G = (V, \Sigma, R, S)$ avec :

  – $V = \{ S \}$

  – $\Sigma = \{ a, b \}$

  – $R = \{ S \rightarrow aaS \mid bbS \mid \varepsilon \}$

  grammaire régulière : $L(G) = (aa \cup bb)^*$

Langages rationnels et grammaires algébriques

- Théorème

  Un langage est rationnel si et seulement si il est engendré par une grammaire régulière.

- Ce théorème exprime que tout langage rationnel est algébrique (et non l'inverse).

- On peut utiliser la preuve de ce théorème pour :
  - passer d'un automate (déterministe ou non) à une grammaire,
  - passer d'une grammaire à un automate.

Langages rationnels et grammaires algébriques

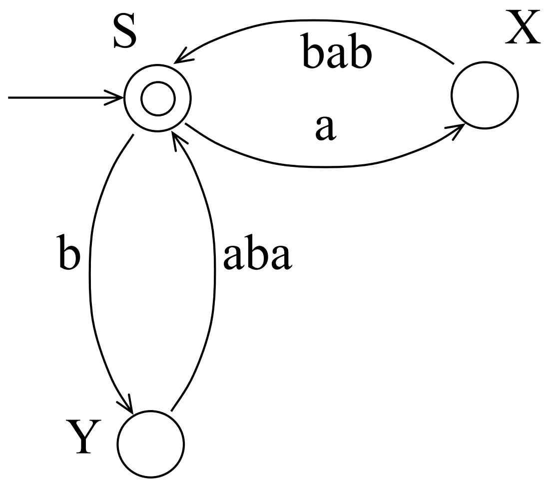
- Exemple ($G \rightarrow M$)

  $G = (V, \Sigma, R, S)$ une grammaire régulière avec :
  - $V = \{ S, X, Y \}$
  - $\Sigma = \{ a, b \}$
  - $R = \{ S \rightarrow aX \mid bY, X \rightarrow aS \mid a, Y \rightarrow bS \mid b \}$

- Soit M tel que $L(M) = L(G)$. Pour chaque non-terminal de G on crée un état dans M de la façon suivante :
  - Si $A \rightarrow wB \in R$ alors on crée dans M une transition de l'état A vers l'état B étiquetée par w
  - Si $A \rightarrow w \in R$ alors on crée dans M une transition de l'état A vers l'état F, où F est le seul état introduit dans M qui ne correspond à aucun non-terminal de G

Langages rationnels et grammaires algébriques

- Exemple ($M \rightarrow G$)
  Soit l'automate :



- Soit G une grammaire régulière telle que $L(G) = L(M)$. Pour chaque transition de M on crée une règle dans G de la façon suivante :
  - Pour toute transition de l'état p vers l'état q étiquetée par w, on crée la règle correspondante dans G : $P \rightarrow wQ$
  - Pour tout état final f de M, on crée dans G une règle d'effacement : $F \rightarrow \varepsilon$

Automates finis

- Transducteurs rationnels
  - Automates finis avec transitions étiquetées par une sorte

    ➔ génération de tokens

- Erreurs jusqu'ici LEXICALES

Grammaires

- Type 0
  - Grammaires générales Pas de restrictions
- Type 1
  - Grammaires contextuelles $uAv \rightarrow uwv$
  - Grammaires croissantes Si $w \rightarrow w' \in R$ alors $|w| \leq |w'|$
- Type 2
  - Grammaires hors contexte Si $w \rightarrow w' \in R$ alors $w \in V$
- Type 3
  - Grammaires régulières

Grammaires

- Grammaires croissantes (contextuelle)

$S \to ABCS$	$S \to aSTc$	$S \to aSBc$	$S \to aSBC$
$S \to T_c$	$S \to aTc$	$S \to abc$	$S \to aBC$
$CA \to AC$	$cT \to Tc$	$cB \to Bc$	$CB \to HB$
$BA \to AB$	$T \to b$	$bB \to bb$	$HB \to HC$
$CB \to BC$			$HC \to BC$
$CT_c \to T_c c$			$aB \to ab$
$CT_c \to T_b c$			$bB \to bb$
$BT_b \to T_b b$			$bC \to bc$
$BT_b \to T_a b$			$cC \to cc$
$AT_a \to T_a a$			
$T_a \to \varepsilon$			

- Grammaire croissante ➔ grammaire contextuelle

Grammaires

$S \rightarrow ABCS$
$S \rightarrow T_c$
$CA \rightarrow AC$
$BA \rightarrow AB$
$CB \rightarrow BC$
$CT_c \rightarrow T_c c$
$CT_c \rightarrow T_b c$
$BT_b \rightarrow T_b b$
$BT_b \rightarrow T_a b$
$AT_a \rightarrow T_a a$
$T_a \rightarrow \varepsilon$

$S \Rightarrow ABCS$
$\Rightarrow ABCABCS$
$\Rightarrow ABCABCT_c$
$\Rightarrow ABACBCT_c$
$\Rightarrow ABABCCT_c$
$\Rightarrow AABBCCT_c$
$\Rightarrow AABBCT_c c$
$\Rightarrow AABBT_b cc$
$\Rightarrow AABT_b bcc$
$\Rightarrow AAT_a bbcc$
$\Rightarrow AT_a abbcc$
$\Rightarrow T_a aabbcc$
$\Rightarrow aabbcc$

Dérivations

- Expressions arithmétiques sur $\{+, \times, (,), id, cte\}$ tokens
- $E \rightarrow id \mid cte \mid E + E \mid E \times E \mid (E)$

$$x + 2.5 \times 4 + (y + z)$$

$$
\begin{aligned}
E \quad & \Rightarrow E + E \\
& \Rightarrow id + E \\
& \Rightarrow id + E \times E \\
& \Rightarrow id + cte \times E \\
& \Rightarrow id + cte \times E + E \\
& \Rightarrow id + cte \times cte + E \\
& \Rightarrow id + cte \times cte + (E) \\
& \Rightarrow id + cte \times cte + (E + E) \\
& \Rightarrow id + cte \times cte + (id + E) \\
& \Rightarrow id + cte \times cte + (id + id)
\end{aligned}
$$

- Dérivation : gauche
- Parenthésage implicite : $(x + (2.5 \times (4 + (y + z))))$

Dérivations

- Expressions arithmétiques sur $\{+, \times, (,), \text{id}, \text{cte}\}$
- $E \rightarrow \text{id} \mid \text{cte} \mid E + E \mid E \times E \mid (E)$

$$x + 2.5 \times 4 + (y + z)$$

$$
\begin{aligned}
E \quad & \Rightarrow E + E \\
& \Rightarrow E + (E) \\
& \Rightarrow E + (E + E) \\
& \Rightarrow E + (E + \text{id}) \\
& \Rightarrow E + (\text{id} + \text{id}) \\
& \Rightarrow E \times E + (\text{id} + \text{id}) \\
& \Rightarrow E \times \text{cte} + (\text{id} + \text{id}) \\
& \Rightarrow E + E \times \text{cte} + (\text{id} + \text{id}) \\
& \Rightarrow E + \text{cte} \times \text{cte} + (\text{id} + \text{id}) \\
& \Rightarrow \text{id} + \text{cte} \times \text{cte} + (\text{id} + \text{id})
\end{aligned}
$$

- Dérivation droite
- Parenthésage implicite : $(((x + 2.5) \times 4) + (y + z))$

Dérivations

- $(((x + 2.5) \times 4) + (y + z)) \neq (x + (2.5 \times (4 + (y + z))))$
- Dérivation droite avec $((x + (2.5 \times 4)) + (y + z))$?
- G ambiguë si $\exists$ w $\in$ L(G) tq plusieurs dérivations droite pour w.
- Lever l'ambiguïté
  - Pas toujours faisable
  - Cas faciles : 1) Nouveau non terminal par niveau de priorité
    2) Récursif gauche si associatif gauche (resp. droite)

Dérivations

- Expressions arithmétiques sur $\{+, \times, (,), id, cte\}$
- $E \rightarrow id \mid cte \mid E + E \mid E \times E \mid (E)$
- $+ < \times$ + moins prioritaire que $\times$
- + et $\times$: associatifs à gauche x+y+z = (x+y)+z

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T \times F \mid F$$
$$F \rightarrow id \mid cte \mid (E)$$

- Unique derivation gauche ou droite
- Un peu plus long et complexe mais univoque

Arbre syntaxique

- Plusieurs dérivations pour un même résultat (permutations, etc.)

  ➔ Représentation invariante

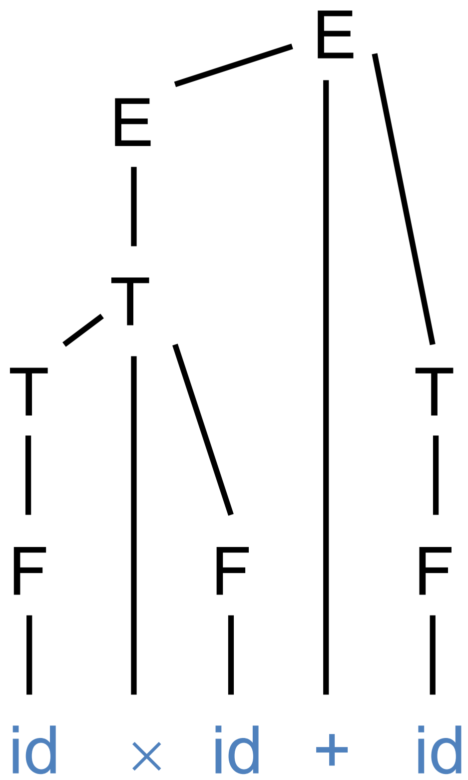
  ➔ Représentation unique lorsque G non ambiguë

- Soit $G = (V, \Sigma, R, S)$, arbres de syntaxe de G :
  - Nœuds internes étiquetés par V
  - Feuilles étiquetées par Σ
  - Si nœud interne N a k fils $a_1, \dots, a_k$ alors $N \to a_1 \dots a_k \in R$

- Arbre de dérivation : $\Lambda = S$ feuilles $\in \Sigma$

Arbre syntaxique

- Exemple



- Arbre de la dérivation $E \Rightarrow^* id \times id + id$

Arbre syntaxique

- Un arbre de dérivation = plusieurs dérivations

  ➔ Stratégies de parcours (parent avant fils)

- Pour arbre de dérivation : mot des feuilles $\in L(G)$

- Réciproque : produire un arbre, récurrence sur longueur de dérivation
  - Nulle : arbre = feuille a
  - $N \Rightarrow^n w_1Mw_2 \Rightarrow w_1a_1 \ldots a_kw_2$:

    ajout de k fils au nœud M de l'arbre de $N \Rightarrow^n w_1Mw_2$

- G ambiguë si $\exists\, w \in L(G)$ avec deux arbres de dérivation distincts

- Construction de l'arbre vers le haut ou vers le bas ?

Analyse descendante

- Exemple : grammaire de Dick

  (1) $S \rightarrow (S)S$

  (2) $S \rightarrow \varepsilon$

- $((()())()) \in L(G)$?

- Construction de l'arbre :
  - Lecture à partir de la gauche — Left
  - Construction dérivation gauche — Left
  - Règle déterminée par 1 caractère à produire — 1

  ➔ LL(1)

Analyse descendante

- Efficace
- Simple

- Pas toutes LL(1)
- Si récursif à gauche → échec

- Par exemple $E \to E + T \mid T$

  $T \to T \times F \mid F$

  $F \to id \mid cte \mid (E)$ pas faisable

Analyse ascendante

$E \rightarrow E + T \mid T$

$T \rightarrow T \times F \mid F$

$F \rightarrow id \mid cte \mid (E)$

- Récursive gauche …
- $E \rightarrow T$ ou $E \rightarrow E + T$?
  - ➔ Analyse ascendante
- Construction de l'arbre
  - Lecture à partir de gauche
  - Dérivation droite à l'envers
  - ➔ LR
- En particulier construction d'une forêt

Analyse ascendante

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T \times F \mid F$$
$$F \rightarrow id \mid cte \mid (E)$$

- Racine de forêt : suite des racines des arbres la constituant
- Opérations :
  - 1. Lecture — Shift
  - 2. Enracinement — Reduce

    Sur juxtaposition f f' construction de fN si $N \rightarrow \Lambda$(f')

$id \times id + id$?

Syntaxe abstraite

- Type et signification : depuis l’arbre de syntaxe
- Utilisateur : ce qu’on écrit
- Concrète : presque comme on écrit   Impropre à bonne compréhension

➔ niveau intermédiaire : syntaxe abstraite dépolluée

- On veut :
  – Objectif 1 : sans ambiguïté
  – Objectif 2 : sans scories
  – Objectif 3 : structure et valeurs

Syntaxe abstraite

- T ensemble d'étiquettes abstraites
- $\Sigma = T \cup \{(;\ );\ ,\}$
- $G = (V, \Sigma, R, S)$ abstraite si

  1. Règles : $N \to t$

     ou $N \to t(N_1, \ldots, N_k)$ pour $N_i \in V$, $t \in T$
  2. Occurrence t unique dans G

- Mot généré : expression abstraite

$T = \{p, m, cte\} \quad E \to p(E,E) \mid m(E,E) \mid cte$

$T = \{+, \times, cte\} \quad E \to +(E,E) \mid \times(E,E) \mid cte$

Syntaxe abstraite

- G abstraite nécessairement non ambiguë   Objectif 1   OK
- Représentation arborescente :
  - facile grâce à la forme des règles

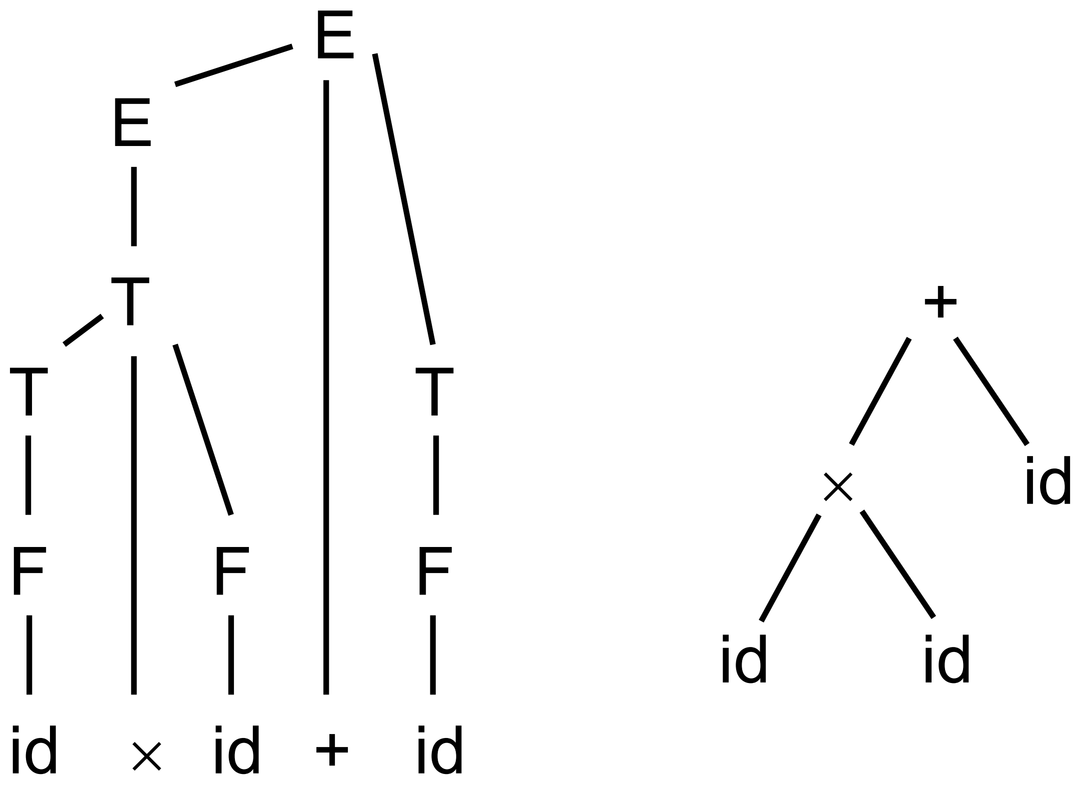
$$\begin{array}{c} t \\ / \quad \backslash \\ a_1 \ldots\; a_k \end{array}$$

- Arbre de syntaxe abstraite A de sorte N : nœuds dans T et
  - $N \rightarrow t \in R$ et $A = t$

    ou
  - $N \rightarrow t(N_1, \ldots, N_k) \in R$ et $\Lambda(A) = t$
- A a alors k fils ASA de sortes respectives $N_1 \ldots N_k$

Syntaxe abstraite

- Comparaison AS / ASA



- Objectif 2 OK

Syntaxe abstraite

- Pour valeurs : étiquette = langage rationnel

$$E \to +(E,E) \mid \times (E,E) \mid cte(Nat)$$

$$Nat \in (0|1| \cdot \cdot \cdot |9)+$$

- Objectif 3 OK

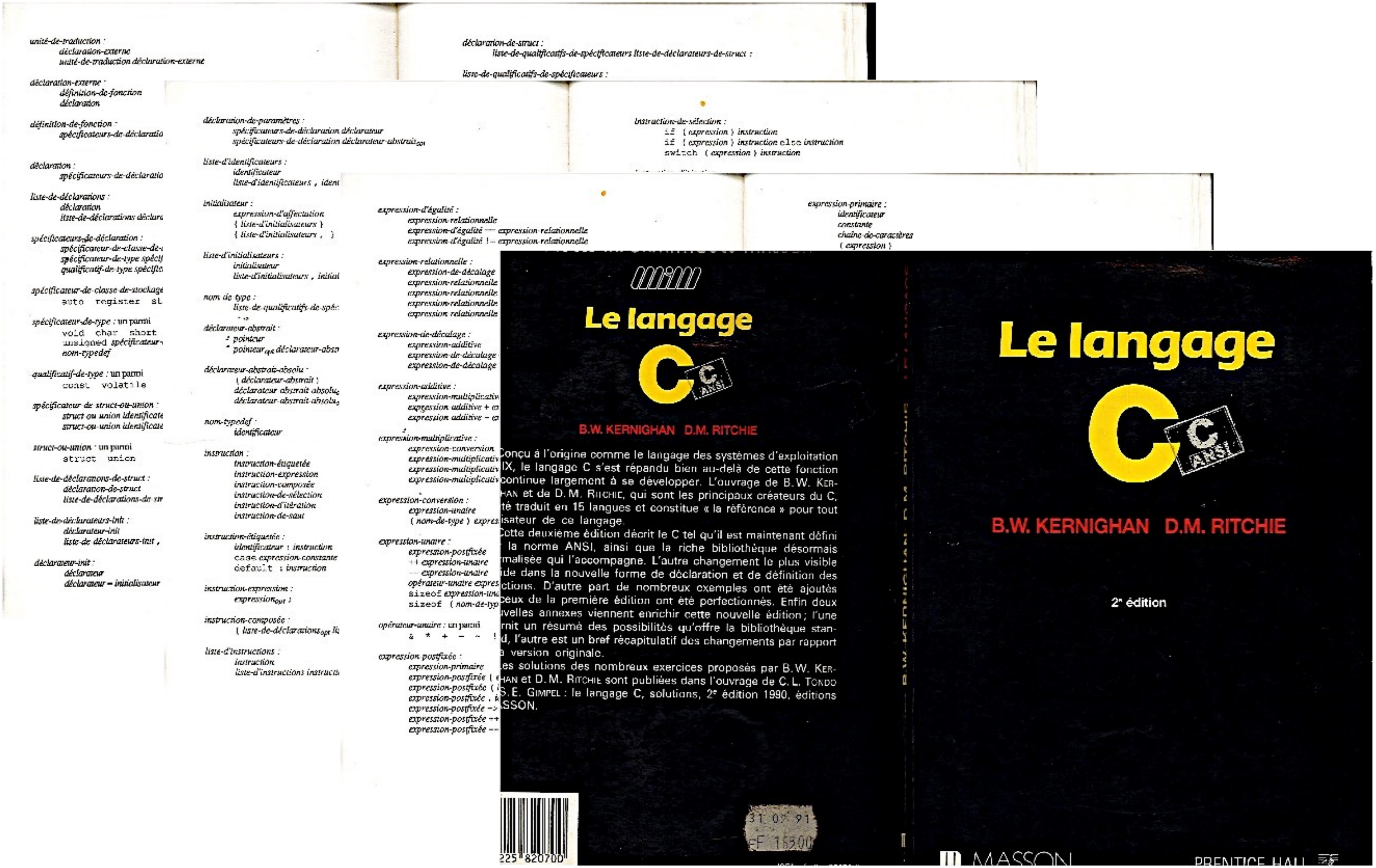
Syntaxe abstraite

- Actions dans une grammaire

$E \to E + T$	$\{ +(E_1, T_1) \}$
$E \to T$	$\{ T_1 \}$
$T \to T \times F$	$\{ \times(T_1, F_1) \}$
$T \to F$	$\{ F_1 \}$
$F \to Nat$	$\{ cte(val(Nat_1)) \}$
$F \to (E)$	$\{ E_1 \}$

- Erreurs ici SYNTAXIQUES

Grammaire du C



Automates à pile

- Un automate à pile est un sextuplet $M = (K, \Sigma, \Gamma, \delta, s, F)$ avec :
  - K : ensemble fini d'états
  - Σ : ensemble fini de symboles d'entrée (alphabet)
  - Γ : ensemble fini de symboles de la pile
  - $s \in K$: état initial
  - $F \subset K$: ensemble des états finaux (acceptants)
  - $\Delta \subset (\ K \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})\ ) \times (\ K \times (\Gamma \cup \{\varepsilon\})\ )$: relation de transition

Automates à pile

- Une transition $((q, \sigma, \gamma), (q', \gamma')) \in \Delta$ où :
  - q est l'état courant
  - σ est le symbole d'entrée courant
  - γ est le symbole sommet de la pile
  - q' est le nouvel état
  - γ' est le nouveau symbole en sommet de pile

  a pour effet :

  (1) De passer de l'état q à l'état q'

  (2) D'avancer la tête de lecture après σ

  (3) De dépiler γ du sommet de la pile

  (4) D'empiler γ' sur la pile

Automates à pile

- Soit $M = (K, \Sigma, \Gamma, \delta, s, F)$ un automate à pile. Une configuration de M est définie par un triplet $(q, w, z) \in K \times \Sigma^* \times \Gamma^*$ où :
  - q est l'état courant de M
  - w est la partie de la chaîne restant à analyser
  - z est le contenu de la pile

- Soient (q, w, z) et (q', w', z') deux configurations d'un automate à pile M. On dit qu'on passe de (q, w, z) à (q', w', z') en une étape

  ssi $\exists\ \sigma \in \Sigma \cup \{\varepsilon\}$ et $\gamma, \gamma' \in \Gamma \cup \{\varepsilon\}$

  tels que $w = \sigma w'$

  et $z = \gamma z'', z' = \gamma' z''$ avec $z'' \in \Gamma^*$

  et $((q, \sigma, \gamma), (q', \gamma')) \in \Delta$

  On note $(q, w, z) \vdash_M (q', w', z')$

Automates à pile

- La relation $\vdash_M^*$ est la fermeture réflexive transitive de $\vdash_M$

- Soit $M = (K, \Sigma, \Gamma, \delta, s, F)$ un automate à pile

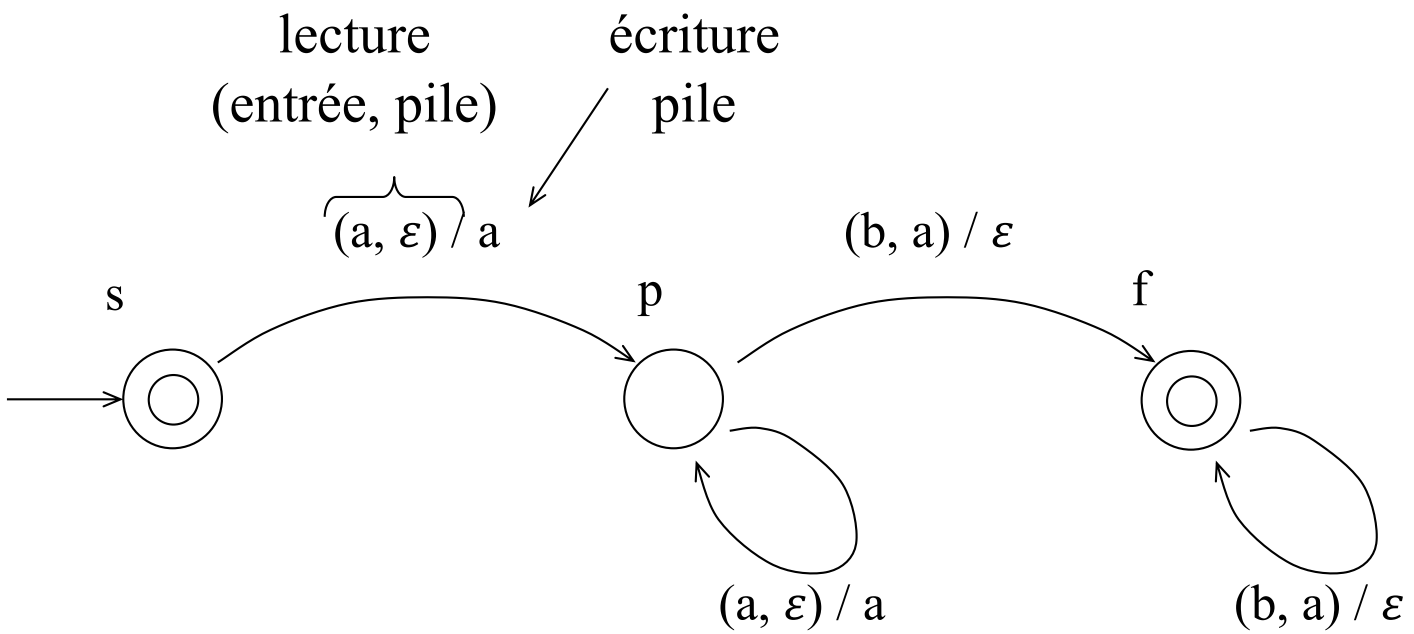
$$\boxed{\text{Un mot } w \in \Sigma^* \text{est accepté par } M \quad \text{ssi } (s, w, \varepsilon) \vdash_M^* (f, \varepsilon, \varepsilon) \text{ avec } f \in F}$$

- Le langage accepté par M, noté $L(M)$, est l'ensemble des mots acceptés par M

Automates à pile

- Exemple : Soit $M = (K, \Sigma, \Gamma, \delta, s, F)$ avec :
  - K = {s, p, f}
  - Σ = {a, b}
  - Γ = {a, b}
  - F = {s, f}

Δ = { ((s, a, ε), (p, a)),
((p, a, ε), (p, a)),
((p, b, a), (f, ε)),
((f, b, a), (f, ε)) }



Automates à pile

- Un automate à pile est déterministe s'il y a au plus une transition applicable pour tout triplet de la forme

  (État courant, symbole d'entrée, sommet de pile).

- Les automates à pile non déterministes reconnaissent plus de langages que les automates à pile déterministes

Automates à pile et grammaires algébriques

- Théorème

  La classe des langages acceptés par les automates à pile est égale à la classe des langages engendrés par les grammaires algébriques

- Un automate à pile est dit simple ssi quelle que soit la transition $((q, \sigma, \gamma), (q', \gamma')) \in \Delta$, on a :

  $\gamma \in \Gamma$ (sauf pour $q = s$ où on ne dépile rien)

  et $|\gamma'| \leq 2$

- Proposition

  On peut transformer tout automate à pile en un automate simple équivalent

Propriétés des langages algébriques

Preuve d’algébricité

- Pour montrer qu'un langage est algébrique, on peut :
  - Soit définir une grammaire algébrique qui engendre ce langage
  - Soit définir un automate à pile qui l'accepte

- Il est également possible d'utiliser les propriétés de stabilité de la classe des langages algébriques

Propriétés des langages algébriques
Propriétés de stabilité

- Théorème

  La classe des langages algébriques est stable par les opérations d'union, de concaténation et d'étoile de Kleene

- Preuve

  Soient deux grammaires $G_1 = (V_1, \Sigma_1, R_1, S_1)$ et $G_2 = (V_2, \Sigma_2, R_2, S_2)$, avec $V_1 \cap V_2 = \emptyset$ (on renomme éventuellement les non-terminaux)

  La preuve (constructive) consiste à :

  - Construire une grammaire G à partir de G_1 et G_2 validant les propriétés de stabilité
  - Montrer que $L(G) = L(G_1)$ op $L(G_2)$ (op $\in \{\cup, .\}$) et $L(G) = L(G_1)^*$

Propriétés des langages algébriques
Propriétés de stabilité

- Preuve

(a) Union
Soit $G = (V, \Sigma, R, S)$ avec :
- $V = V_1 \cup V_2 \cup \{S\}$ où $S \notin V_1 \cup V_2$ (renommage éventuel)
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $R = R_1 \cup R_2 \cup \{S \rightarrow S_1 \mid S_2\}$

(b) Concaténation
Soit $G = (V, \Sigma, R, S)$ avec :
- $V = V_1 \cup V_2 \cup \{S\}$ où $S \notin V_1 \cup V_2$ (renommage éventuel)
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $R = R_1 \cup R_2 \cup \{S \rightarrow S_1S_2\}$

(c) Opération étoile
Soit $G = (V, \Sigma, R, S)$ avec :
- $V = V_1 \cup \{S\}$ où $S \notin V_1$ (renommage éventuel)
- $\Sigma = \Sigma_1$
- $R = R_1 \cup \{S \rightarrow S_1S \mid \varepsilon\}$

Propriétés des langages algébriques

Propriétés de stabilité

- Contrairement à la classe des langages rationnels, la classe des langages algébriques n'est pas stable par intersection et complémentation

- Théorème

  L'intersection d'un langage rationnel et d'un langage algébrique est algébrique

Propriétés des langages algébriques

Lemme de l'étoile pour les langages algébriques

- Définition

  Une grammaire algébrique $G = (V, \Sigma, R, S)$ est sous forme normale de Chomsky si chaque règle est de la forme :

  $A \rightarrow BC$ avec $B, C \in V - \{S\}$

  ou $A \rightarrow \sigma$ avec $\sigma \in \Sigma$

  ou $A \rightarrow e$

- Théorème

  Pour toute grammaire algébrique, il existe une grammaire sous forme normale de Chomsky équivalente

Propriétés des langages algébriques

Lemme de l'étoile pour les langages algébriques

- Théorème (lemme de la double étoile)

Soit L un langage algébrique

Il existe un nombre k, dépendant de L, tel que tout mot $z \in L$, $|z| \geq k$, peut être décomposé en $z = uvwxy$ avec :

(*i*) $|vwx| \leq k$

(*ii*) $|v| + |x| > 0$ (ie. $v \neq \varepsilon$ ou $x \neq \varepsilon$)

(*iii*) $uv^iwx^iy \in L$, $\forall\ i \geq 0$

(d'où l'appellation de double étoile : v^i et x^i = v^* et x^*)

Propriétés des langages algébriques

Lemme de l'étoile pour les langages algébriques

- Lemme

  Soit $G = (V, \Sigma, R, S)$ une grammaire algébrique sous forme normale de Chomsky

  Soit $S \Rightarrow_G^* w$ une dérivation de $w \in \Sigma^*$ dont l'arbre de dérivation est noté T

  Si la hauteur de T est n alors $|w| \leq 2^{n-1}$

- Corollaire

  Soit $G = (V, \Sigma, R, S)$ une grammaire algébrique sous forme normale de Chomsky

  Soit $S \Rightarrow_G^* w$ une dérivation de $w \in L(G)$

  Si $|w| \geq 2^n$ alors l'arbre de dérivation est de hauteur $\geq n+1$

Propriétés des langages algébriques

Lemme de l'étoile pour les langages algébriques

- Exemple

Montrons que $L = \{ a^n b^n c^n \mid n \geq 0\}$ est non algébrique

Supposons que L est algébrique

D'après le lemme de la double étoile, il existe une constante k, dépendant de L, telle que :

$\forall z \in L, |z| \geq k$, z peut être décomposé en z = uvwxy avec :

(*i*) $|vwx| \leq k$

(*ii*) $|v| + |x| > 0$ (au moins un des deux n'est pas le mot vide)

(*iii*) $uv^iwx^iy \in L, \forall i \geq 0$

Propriétés des langages algébriques
Lemme de l'étoile pour les langages algébriques

- Exemple

  Considérons la chaîne particulière $z_0 = a^k b^k c^k$.
  On a bien $z_0 \in L$ et $|z_0| = 3k \geq k$.
  Les décompositions de $z_0 = uvwxy$ satisfaisant $|vwx| \leq k$ et $|v| + |x| > 0$ sont telles que :

  – Soit l'une des sous-chaînes v ou x contient plus d'un type de symbole, de la forme a^+b^+ ou b^+c^+.
  uv^iwx^iy avec $i > 1$ contient un a après un b ou un b après un c.
  (par exemple uv^2wx^2y = u aabb aabb w x x y, si v = aabb)
  donc la chaîne uv^iwx^iy n'est plus de la forme $a^pb^pc^p$ avec $p \geq 0$,
  donc $uv^iwx^iy \notin L$ pour $i > 1$.

  – Soit v et x sont des sous-chaînes de a^k ou de b^k ou de c^k.
  Comme au plus une des chaînes v ou x est vide, toute chaîne de la forme uv^iwx^iy avec $i > 1$ est caractérisée par une augmentation de un ($v = \varepsilon$ ou $x = \varepsilon$) ou deux ($v \neq \varepsilon$ et $x \neq \varepsilon$) des trois types de terminaux.
  donc pour $i > 1$, la chaîne uv^iwx^iy est de la forme $a^pb^qc^r$ mais avec $p \neq q$ ou $q \neq r$.
  donc $uv^iwx^iy \notin L$ pour $i > 1$.

  – Pas d'autres possibilités pour v et x, les autres sous-chaines u, w et y n'influencent pas.

  Pour toutes les décompositions possibles de la chaîne z_0 il y a une contradiction.

  Donc l'hypothèse est fausse $\Rightarrow$ L non algébrique.

Propriétés des langages algébriques

Preuve de non algébricité

- Pour montrer qu'un langage est non algébrique, on peut utiliser :
  - Le lemme de la double étoile
  - Les propriétés de stabilité de la classe des langages algébriques
  - Le théorème qui dit que l'intersection d'un langage algébrique et d'un langage rationnel est algébrique

Notion de décidabilité

- Une question est décidable s'il existe un algorithme (c'est-à-dire un processus déterministe) qui s'arrête avec une réponse (oui ou non) pour chaque entrée

- Une question est indécidable si un tel algorithme n'existe pas

Problèmes indécidables pour les langages algébriques

- Théorème

  Les questions suivantes sont décidables :
  - Étant donnés une grammaire algébrique G et un mot w
    est-ce que $w \in L(G)$?
  - Étant donnée une grammaire algébrique G, est-ce que $L(G) = \emptyset$?

  Les questions suivantes sont indécidables :
  - Soit G une grammaire algébrique. Est-ce que $L(G) = \Sigma^*$?
  - Soient G_1 et G_2 deux grammaires algébriques. Est-ce que $L(G_1) = L(G_2)$?
  - Soient M_1 et M_2 deux automates à pile. Est-ce que $L(M_1) = L(M_2)$?
  - Soit M un automate à pile. Trouver un automate à pile équivalent minimal en nombre d'états.