Grammaires et Analyse Syntaxique - Cours 09 Grammaires et Automates a Pile

Ralf Treinen

INFORMATIQUE
Sciences
Université Paris Cité

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

treinen@irif.fr

1er avril 2022

Un mécanisme analyseur pour les langages algébriques ?

- Nous avons vu au cours 3 ce tableau :

Langages ···	Formalisme générateur	Formalisme analyseur
··· réguliers	Expressions rationnelles	Automates finis
··· algébriques	Grammaires algébriques	? ? ?

- Nous avons jusqu'à maintenant vu des analyseurs pour des cas particuliers : les langages algébriques qui sont LL(1), LR(0), LR(1).
- Les algorithmes que nous avons étudiés sont très efficaces (temps linéaire dans la taille de l'arbre de dérivation) mais ne peuvent pas analyser tous les langages algébriques.
- Aujourd'hui nous allons remplir la case " ? ? ?"

Un mécanisme analyseur pour les langages algébriques ?

- Nous verons la semaine prochaine que les langages algébriques ne sont pas clos sous complément.
- Souvenez vous comment on avait montré en L2 que les langages réguliers sont clos sous complément : on considère un automate *déterministe* pour un langage régulier, et on inverse simplement le statut acceptant/non-acceptant des états.
- Le fait que les langages algébriques ne sont *pas* clos sous complément indique qu'un mécanisme analyseur pour les langages algébriques ne peut pas toujours être déterministe.
- Nous avons aussi vu que les algorithmes d'analyse utilisent toujours une pile, soit explicitement (LR(0) et LR(1)), ou implicitement car il y a des fonctions récursives (LL(1)).

Les automates à pile

- Un *automate à pile* est un sextuple $(\Sigma, Q, q_0, \Gamma, Z_0, \delta)$ tel que
  - Σ un ensemble fini de symboles terminaux ;
  - Q un ensemble fini d'états ;
  - $q_0 \in Q$ l'état initial ;
  - Γ un ensemble fini de symboles de pile ;
  - $Z_0 \in \Gamma$ le symbole de pile initiale ;
  - $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times \Gamma \times Q \times \Gamma^*$ est la relation de transition.
- En anglais : *pushdown automaton*.

Exemple

- Automate :

$$(\underbrace{\{a, b\}}_{\Sigma}, \underbrace{\{q_0, q_1\}}_{Q}, \underbrace{q_0}_{\text{initial}}, \underbrace{\{Z, A, B\}}_{\Gamma}, \underbrace{Z}_{\text{initial}}, \underbrace{\delta}_{\text{transitions}})$$

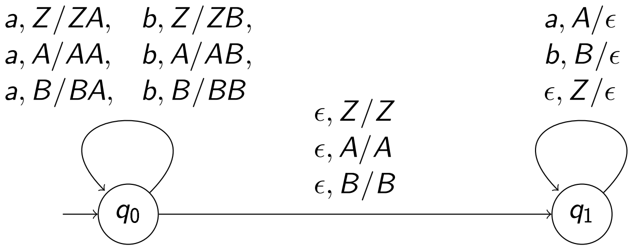
- avec δ comme suit :

$$\left\{ \begin{array}{lll} (q_0, a, Z, q_0, ZA), & (q_0, a, A, q_0, AA), & (q_0, a, B, q_0, BA), \\ (q_0, b, Z, q_0, ZB), & (q_0, b, A, q_0, AB), & (q_0, b, B, q_0, BB), \\ (q_0, \epsilon, Z, q_1, Z), & (q_0, \epsilon, A, q_1, A), & (q_0, \epsilon, B, q_1, B), \\ (q_1, a, A, q_1, \epsilon), & (q_1, b, B, q_1, \epsilon), & (q_1, \epsilon, Z, q_1, \epsilon) \end{array} \right\}$$

$$\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times \Gamma \times Q \times \Gamma^*$$

Représentation graphique

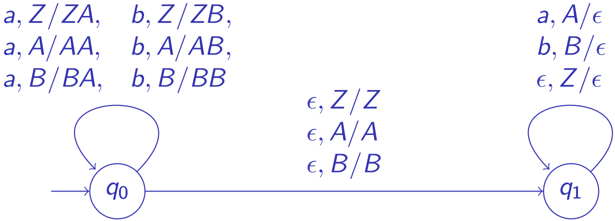
$$
\begin{array}{llll}
\{ & (q_0, a, Z, q_0, ZA), & (q_0, a, A, q_0, AA), & (q_0, a, B, q_0, BA), \\
 & (q_0, b, Z, q_0, ZB), & (q_0, b, A, q_0, AB), & (q_0, b, B, q_0, BB), \\
 & (q_0, \epsilon, Z, q_1, Z), & (q_0, \epsilon, A, q_1, A), & (q_0, \epsilon, B, q_1, B), \\
 & (q_1, a, A, q_1, \epsilon), & (q_1, b, B, q_1, \epsilon), & (q_1, \epsilon, Z, q_1, \epsilon) \quad \}
\end{array}
$$



Les configurations d'un automate à pile

- Une *configuration* d'un automate à pile est un triple
$$(q, \gamma, w) \in Q \times \Gamma^* \times \Sigma^*$$
- Une configuration décrit où on est actuellement dans l'exécution d'un automate à pile :
  - l'état $q \in Q$;
  - le contenu de la pile $\gamma \in \Gamma^*$ (le dernier symbole de γ correspond au sommet de la pile) ;
  - le mot qu'il reste à consommer $w \in \Sigma^*$.
- Configuration initiale pour un mot d'entrée w :
$$(q_0, Z_0, w)$$
- Une configuration est acceptante quand elle est de la forme
$$(q, \epsilon, \epsilon)$$
pour un $q \in Q$ quelconque. Le critère d'acceptation est donc que la pile est vide.

Exemple d'exécution d'un automate à pile



- Exécution sur *abba* :

$$
\begin{aligned}
& (q_0, Z, abba) && \text{configuration initiale pour entrée } abba \\
\vdash\ & (q_0, ZA, bba) \\
\vdash\ & (q_0, ZAB, ba) \\
\vdash\ & (q_1, ZAB, ba) \\
\vdash\ & (q_1, ZA, a) \\
\vdash\ & (q_1, Z, \epsilon) \\
\vdash\ & (q_1, \epsilon, \epsilon) && \text{configuration acceptante}
\end{aligned}
$$

Définition exacte de l'exécution d'un automate à pile

- On définit une relation $\vdash$ de passage d'une configuration à une autre.
- Pour le cas où on consomme un symbole de l'entrée :

  $$(q_1, \gamma X, aw) \vdash (q_2, \gamma\alpha, w) \text{ quand } (q_1, a, X, q_2, \alpha) \in \delta$$

  Quand l'état actuel est q_1 et X est au sommet de la pile, on consomme a, change l'état en q_2, et remplace au sommet de la pile X par α.
- Pour le cas où on ne consomme pas de symbole de l'entrée :

  $$(q_1, \gamma X, w) \vdash (q_2, \gamma\alpha, w) \text{ quand } (q_1, \epsilon, X, q_2, \alpha) \in \delta$$

  Quand l'état actuel est q_1 et X est au sommet de la pile, on change l'état en q_2, et remplace au sommet de la pile X par α.

Le langage reconnu par un automate a pile

- Pour un automate a pile $P = (\Sigma, \Sigma, q_0, \Gamma, Z_0, \delta)$, le *langage reconnu par P* est

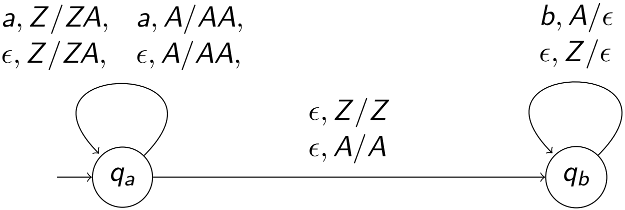
$$\mathcal{L}(P) = \{w \in \Sigma^* \mid (q_0, Z_0, w) \vdash^* (q, \epsilon, \epsilon)\}$$

  pour un $q \in Q$.
- Le langage reconnu par automate de l'exemple est le langage des palindromes de longueur paire $\{w\overline{w} \mid w \in \{a, b\}^*\}$.
- Cet automate utilise sa pile pour stocker la première partie du mot d'entrée quand il est dans l'état q_0, et vérifie la deuxième partie quand il est dans l'état q_1.
- Cet automate est non déterministe car il doit "deviner" quand il est au milieu du mot.

Exécution d'un automate à pile

- Pour l'exécution d'un automate à pile il y maintenant *quatre* possibilités :
  1. on atteint une configuration acceptante (entrée épuisée et pile vide) : le mot est accepté.
  2. on consomme toute l'entrée mais la pile n'est pas vide.
  3. on n'arrive même pas à consommer toute l'entrée car aucune transition est possible.
  4. l'exécution ne termine pas.
- Le quatrième cas n'a pas d'équivalent dans les automates finis du L2.

Exemple d'un automate avec exécution infinie



- Cet automate reconnaît $\{a^n b^m \mid n \leq m\}$.
- Dans l'état q_a il "devine" une valeur $m \geq n$.
- Il permet des exécutions infinies comme

$$(q_a, Z, bb) \vdash (q_a, ZA, bb) \vdash (q_a, ZAA, bb) \vdash (q_a, ZAAA, bb) \vdash \dots$$

Automates à pile avec états acceptants

- Il y a un autre modèle des automates à pile qui a en plus du sextuple un ensemble d'états acceptants.
- Dans ce modèle alternatif, une configuration est acceptante quand son état est acceptant, peu importe le contenu de la pile.
- Les deux modèles sont équivalents : on peut transformer un automate qui accepte par pile vide en un automate qui accepte par état acceptant et qui reconnaît le même langage, et inversement.
- Il est utile d'avoir les deux versions des automates à pile pour pouvoir choisir celle qui est le plus commode (voir les sections suivantes).

De l'acceptation par état à l'acceptation par pile vide

- Soit donné un automate

$$A_1 = (\Sigma, Q, q_0, \Gamma, Z_0, \delta, F)$$

  qui accepte par état.
- On cherche à construire un automate A_2 à acceptation par pile vide tel que $\mathcal{L}(A_1) = \mathcal{L}(A_2)$.
- Première idée : On définie A_2 comme A_1 plus des transitions qui lui permettent, une fois qu'il est dans un état de F, de vider sa pile. C-à-d on ajoute dans A_2 des transitions

$$\{(q, \epsilon, X, q, \epsilon) \mid q \in F, X \in \Gamma\}$$

- Comme ça on aura que, quand A_1 accepte, A_2 accepte aussi.
- Mais est-ce que l'inverse est aussi vrai ?

Problèmes avec la première idée

- Notre manipulation permet l'automate A_2 à supprimer des symboles de la pile *chaque fois* qu'il est dans un état $q \in F$, même si c'est au milieu d'une exécution.
- Solution : On utilise un nouvel état de "vidange", au lieu de permettre de vider la pile dans n'importe quel état acceptant.
- L'automate A_2 accepte quand la pile est vide, mais dans ce cas il se peut que A_1 n'accepte pas (quand l'état n'est pas acceptant).
- Solution : On utilise en A_2 un nouveau marqueur pour le bas de la pile, afin d'éviter un vidange inopportun de la pile.

Définition de A_2

- Soit donné

$$A_1 = (\Sigma, Q, q_0, \Gamma, Z_0, \delta, F)$$

- Nous définissons :

$$A_2 = (\Sigma, Q \cup \{q_0', q_v\}, q_0', \Gamma \cup \{Z_0'\}, Z_0', \delta')$$

- avec

$$\begin{aligned}
\delta' = \delta \quad & \cup \quad \{(q_0', \epsilon, Z_0', q_0, Z_0' Z_0)\} \\
& \cup \quad \{(q, \epsilon, X, q_v, X) \mid q \in F, X \in \Gamma \cup \{Z_0'\}\} \\
& \cup \quad \{(q_v, \epsilon, X, q_v, \epsilon) \mid X \in \Gamma \cup \{Z_0'\}\}
\end{aligned}$$

De l'acceptation par pile vide à l'acceptation par état

- Soit donné un automate

  $$A_1 = (\Sigma, Q, q_0, \Gamma, Z_0, \delta)$$

  qui accepte par pile vide.
- On cherche à construire un automate A_2 à acceptation par états acceptants tel que $\mathcal{L}(A_1) = \mathcal{L}(A_2)$.
- L'idée est que A_2 va dans un état acceptant quand la pile est vide.
- Le problème est qu'on ne peut plus faire de transition quand la pile est vide. Solution : ajouter un nouveau marqueur pour le bas de la pile.

Définition de A_2

- Soit donné
$$A_1 = (\Sigma, Q, q_0, \Gamma, Z_0, \delta)$$
- Nous définissons :
$$A_2 = (\Sigma, Q \cup \{q'_0, q_f\}, q'_0, \Gamma \cup \{Z'_0\}, Z'_0, \delta', \{q_f\})$$
- avec
$$\begin{aligned} \delta' = \delta \quad & \cup \quad \{(q'_0, \epsilon, Z'_0, q_0, Z'_0 Z_0)\} \\ & \cup \quad \{(q, \epsilon, Z'_0, q_f, \epsilon) \mid q \in Q\} \end{aligned}$$

Clôture par intersection avec des langages réguliers

- Les langages algébriques sont clos par intersection avec des langages réguliers.
- C'est-à-dire : Si L_1 est algébrique et L_2 est régulier, alors $L_1 \cap L_2$ est algébrique.
- Soit $A_1 = (\Sigma, Q_1, q_0^1, \Gamma_1, Z_1, \delta_1, F_1)$ un automate à pile qui accepte par états acceptants, et $A_2 = (\Sigma, Q_2, q_0^2, F_2, \delta_2)$ un automate fini déterministe.
- L'idée est la suivante : Nous exécutons les deux automates en parallèle, ce qui est possible car seulement A_1 agit sur la pile.
- Nous définissons un nouvel automate à pile

$$A_3 = (\Sigma, Q_1 \times Q_2, (q_0^1, q_0^2), \Gamma_1, Z_1, \delta_3, F_1 \times F_2)$$

Clôture par intersection avec des langages réguliers

- Les transitions suivantes exécutent A_1 et A_2 en parallèle pour le cas d'un symbole $a \in \Sigma$:

  $$((q_1, q_2), a, X, (q_1', q_2'), \alpha) \in \delta_3$$

  quand
  - $(q_1, a, X, q_1', \alpha) \in \delta_1$
  - $(q_2, a, q_2') \in \delta_2$
- Les transitions ϵ sont seulement exécutées par A_1 :

  $$((q_1, q_2), \epsilon, X, (q_1', q_2), \alpha) \in \delta_3$$

  quand
  - $(q_1, \epsilon, X, q_1', \alpha) \in \delta_1$

Clôture des langages algébriques sous intersection ?

- Est-ce que la classe des langages algébriques et close sous intersection ?
- Si on veux appliquer la construction pour l'intersection d'un langage algébrique et d'un régulier à l'intersection de deux algébriques il nous faut deux piles !
- On ne peut pas simuler deux piles par une seule car les deux peuvent croître ou décroître à des rythmes différents.
- On verra la semaine prochaine que la réponse est non : la classe des langages algébriques n'est pas close sous intersection.

Traduction Grammaire vers Automate à Pile

- Soit $G = (\Sigma, N, S, P)$ une grammaire algébrique.
- Nous définissons $A = (\Sigma, \{q\}, q, N \cup \Sigma, S, \delta)$ avec

$$\begin{array}{rcl} \delta & = & \{(q, a, a, q, \epsilon) \mid a \in \Sigma\} \\ & \cup & \{(q, \epsilon, N, q, \overline{\alpha}) \mid (N \to \alpha) \in P\} \end{array}$$

- On peut montrer que :

$$\mathcal{L}(G) = \mathcal{L}(A)$$

- En fait cet automate fait une analyse descendante, mais de façon violemment non-déterministe.

Exemple construction automate à pile pour une grammaire

- Donnée la grammaire

$$G = (\{a, b\}, \{S\}, S, \{S \to aSb \mid \epsilon\})$$

- On construit l'automate

$$A = (\{a, b\}, \{q\}, q, \{S, a, b\}, S, \delta)$$

où δ contient les transitions suivantes :

$$\begin{array}{l} (q, a, a, q, \epsilon) \\ (q, b, b, q, \epsilon) \\ (q, \epsilon, S, q, bSa) \\ (q, \epsilon, S, q, \epsilon) \end{array}$$

Exemple d'exécution de cet automate

- Transitions

$$\begin{array}{ll} (q, \mathtt{a}, \mathtt{a}, q, \epsilon) & (q, \mathtt{b}, \mathtt{b}, q, \epsilon) \\ (q, \epsilon, \mathsf{S}, q, \mathtt{b}\mathsf{S}\mathtt{a}) & (q, \epsilon, \mathsf{S}, q, \epsilon) \end{array}$$

- Sur le mot d'entrée `aabb` :

$$\begin{array}{l} (q, \mathsf{S}, \mathtt{aabb}) \\ (q, \mathtt{b}\mathsf{S}\mathtt{a}, \mathtt{aabb}) \\ (q, \mathtt{b}\mathsf{S}, \mathtt{abb}) \\ (q, \mathtt{bb}\mathsf{S}\mathtt{a}, \mathtt{abb}) \\ (q, \mathtt{bb}\mathsf{S}, \mathtt{bb}) \\ (q, \mathtt{bb}, \mathtt{bb}) \\ (q, \mathtt{b}, \mathtt{b}) \\ (q, \epsilon, \epsilon) \end{array}$$

Traduction d'un automate à pile en grammaire

- Soit $A = (\Sigma, Q, q_0, \Gamma, Z_0, \delta)$ un automate à pile qui accepte par pile vide.
- L'idée derrière la grammaire est d'avoir des non-terminaux de la forme $[q_0, X, q_1]$ avec $q_0, q_1 \in Q$ et $X \in \Gamma$. Le langage des mots qu'on peut dériver à partir de $[q_0, X, q_1]$ sera

$$\{w \in \Sigma^* \mid (q_0, X, w) \vdash^* (q_1, \epsilon, \epsilon)\}$$

- c'est-à-dire le langage de tous les mots qui permettent l'automate d'aller de l'état q_0 à q_1 en supprimant X du sommet de sa pile.
- On définit la grammaire $G = (\Sigma, (Q \times \Gamma \times Q) \cup \{S\}, S, P)$, (voir les transparents suivant pour P)

Traduction d'un automate à pile en grammaire

- Productions pour l'axiome :

$$\{S \to [q_0, Z_0, q] \mid q \in Q\}$$

- car on veut avoir dans le langage engendré par la grammaire tous les mots acceptés par l'automate, c'est-à-dire tous les mots w qui permettent l'automate d'aller de son état initial q_0 à un état q quel conque en supprimant le seul symbole Z_0 se trouvant initialement sur la pile.

Traduction d'un automate à pile en grammaire

- Productions pour une transition (q, c, X, q_1, ϵ) avec $c \in \Sigma \cup \{\epsilon\}$:
$$\{[q, X, q_1] \to c\}$$
- car, en consommant le mot c, l'automate peut aller de l'état q à l'état q_1 en supprimant un X du sommet de la pile.

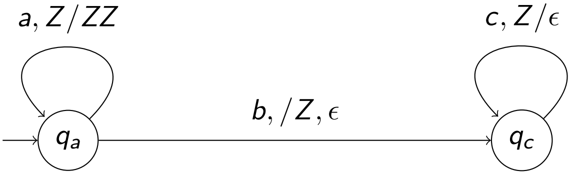
Traduction d'un automate à pile en grammaire

- Productions pour une transition $(q, c, X, q_1, X_n \dots X_1)$, $c \in \Sigma \cup \{\epsilon\}$, quand $n \geq 1$:

$$\{[q, X, p] \to c[q_1, X_1, q_2][q_2, X_2, q_3] \dots [q_n, X_n, p] \mid p, q_2, \dots, q_n \in Q\}$$

- car, en consommant un mot de la forme $cv_1 \cdots v_n$, l'automate peut aller de l'état q à l'état p en supprimant un X du sommet de la pile quand on a, pour chaque i : en consommant le mot v_i l'automate peut aller de q_i à q_{i+1} $(p = q_{n+1})$ et supprimer X_i de la pile.
- On peut montrer que $\mathcal{L}(G) = \mathcal{L}(A)$.

Exemple

- 

- Cet automate reconnaît le langage $\{a^n bc^n \mid n \geq 0\}$.
- Transitions :

$$(q_a, a, Z, q_a, ZZ)$$
$$(q_a, b, Z, q_c, \epsilon)$$
$$(q_c, c, Z, q_c, \epsilon)$$

Exemple

- Productions pour l'axiome :

$$S \to [q_a, Z, q_a] \mid [q_a, Z, q_c]$$

- Pour la transition $(q_c, c, Z, q_c, \epsilon)$:

$$[q_c, Z, q_c] \to c$$

- Pour la transition $(q_a, b, Z, q_c, \epsilon)$:

$$[q_a, Z, q_c] \to b$$

- Pour la transition (q_a, a, Z, q_a, ZZ) :

$$\begin{array}{l} [q_a, Z, q_a] \to a[q_a, Z, q_a][q_a, Z, q_a] \mid a[q_a, Z, q_c][q_c, Z, q_a] \\ [q_a, Z, q_c] \to a[q_a, Z, q_a][q_a, Z, q_c] \mid a[q_a, Z, q_c][q_c, Z, q_c] \end{array}$$

Exemple

- Les non terminaux productifs sont :

$$[q_c, Z, q_c], [q_a, Z, q_c]$$

- Les non terminaux non productifs sont :

$$[q_a, Z, q_a], [q_c, Z, q_a]$$

- On supprime toutes les règles qui contiennent des non terminaux qui ne sont pas productifs :

$$\begin{array}{l} S \to [q_a, Z, q_c] \\ [q_a, Z, q_c] \to a[q_a, Z, q_c][q_c, Z, q_c] \mid b \\ [q_c, Z, q_c] \to c \end{array}$$

- ou, après simplification :

$$S \to aSc \mid b$$

Équivalence grammaires et automates à pile

- Nous avons montré comment construire pour une grammaire algébrique G un automate à pile A tel que $\mathcal{L}(G) = \mathcal{L}(A)$, et vice versa.
- Donc : Un langage est algébrique si et seulement s'il est reconnaissable par un automate à pile.
- Ce qui nous permet de compléter le tableau :

Langages $\cdots$	Formalisme générateur	Formalisme analyseur
$\cdots$ réguliers	Expressions rationnelles	Automates finis
$\cdots$ algébriques	Grammaires algébriques	Automates à pile