

Grammaires et Analyse Syntaxique - Cours 10

Grammaires et Automates à Pile

Ralf Treinen



treinen@irif.fr

28 mars 2024

Un mécanisme analyseur pour les langages algébriques ?

- ▶ Nous avons vu au cours 3 ce tableau :

Langages ...	Formalisme générateur	Formalisme analyseur
... rationnels	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 avons vu au cours 9 que la classe des langages algébriques n'est pas close sous complément.
- ▶ Souvenez vous comment on avait montré en L2 que les langages rationnels sont clos sous complément : on considère un automate *déterministe* et complet pour un langage rationnel, 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 :

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

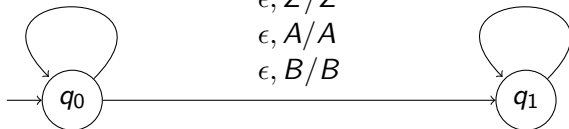
$$\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times \Gamma \times Q \times \Gamma^*$$

Représentation graphique

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

$$\begin{array}{ll} a, Z/ZA, & b, Z/ZB, \\ a, A/AA, & b, A/AB, \\ a, B/BA, & b, B/BB \end{array}$$

$$\begin{array}{l} a, A/\epsilon \\ b, B/\epsilon \\ \epsilon, Z/\epsilon \end{array}$$

$$\begin{array}{l} \epsilon, Z/Z \\ \epsilon, A/A \\ \epsilon, B/B \end{array}$$


Les configurations d'un automate à pile

- ▶ Une *configuration* d'un automate à pile est un triple

$$(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$$

- ▶ Une configuration décrit où on est actuellement dans l'exécution d'un automate à pile :

- ▶ l'état $q \in Q$;
- ▶ le mot qu'il reste à consommer $w \in \Sigma^*$.
- ▶ le contenu de la pile $\gamma \in \Gamma^*$ (le dernier symbole de γ correspond au sommet de la pile) ;

- ▶ Configuration initiale pour un mot d'entrée w :

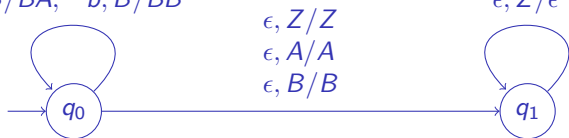
$$(q_0, w, Z_0)$$

- ▶ 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, et le mot entièrement consommé.

Exemple d'exécution d'un automate à pile

 $a, Z/ZA, \quad b, Z/ZB,$ $a, A/AA, \quad b, A/AB,$ $a, B/BA, \quad b, B/BB$ $a, A/\epsilon$ $b, B/\epsilon$ $\epsilon, Z/\epsilon$ ► Exécution sur $abba$:

- $(q_0, abba, Z)$ configuration initiale pour entrée $abba$
- $\vdash (q_0, bba, ZA)$
- $\vdash (q_0, ba, ZAB)$
- $\vdash (q_1, ba, ZAB)$
- $\vdash (q_1, a, ZA)$
- $\vdash (q_1, \epsilon, Z)$
- $\vdash (q_1, \epsilon, \epsilon)$ configuration acceptante

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 :

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

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 :

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

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 à pile

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

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

pour un $q \in Q$.

- ▶ Le langage reconnu par automate de l'exemple est le langage des palindromes de longueur paire $\{w\bar{w} \mid w \in \{a, b\}^*\}$.
- ▶ 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.

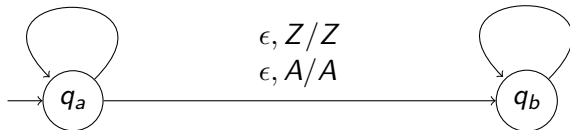
Exécution d'un automate à pile

- ▶ Pour l'exécution d'un automate à pile il y a 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

$a, Z/ZA,$ $a, A/AA,$
 $\epsilon, Z/ZA,$ $\epsilon, A/AA,$

$b, A/\epsilon$
 $\epsilon, Z/\epsilon$



- ▶ 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

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

Pourquoi ne peut-on pas les déterminer

- ▶ Parfois il y a un automate à pile déterministe pour un langage, mais pas toujours.
- ▶ Exemple où l'automate à pile est déterministe : l'automate pour $\{a^n b^n \mid n \geq 0\}$.
- ▶ Exemple où on ne peut trouver que des automates à pile non-déterministes : langage des palindromes.
- ▶ En fait, l'automate pour les palindromes doit "deviner" quand il est au milieu du mot.

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 la plus commode (voir les sections suivantes).

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

$$\begin{aligned} \delta' = \delta \quad & \cup \quad \{(q'_0, \epsilon, Z'_0, q_0, Z'_0 Z_0)\} \\ & \cup \quad \{(q, \epsilon, Z'_0, q_f, \epsilon) \mid q \in Q\} \end{aligned}$$

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 définit 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.
- ▶ Il faut aussi éviter que A_2 accepte car la pile est vide, dans des cas où A_1 est arrivé dans un état non-acceptant et une pile vide
- ▶ Solution : On utilise en A_2 un nouveau marqueur pour le bas de la pile.

Définition de A_2

- Soit donné

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

- Nous définissons :

$$A_2 = (\Sigma, Q \cup \{q'_0, q_v\}, q'_0, \Gamma \cup \{Z'_0\}, Z'_0, \delta')$$

- avec

$$\begin{aligned} \delta' = \delta \quad & \cup \quad \{(q'_0, \epsilon, Z'_0, q_0, Z'_0 Z_0)\} \\ & \cup \quad \{(q, \epsilon, X, q_v, X) \mid q \in F, X \in \Gamma \cup \{Z'_0\}\} \\ & \cup \quad \{(q_v, \epsilon, X, q_v, \epsilon) \mid X \in \Gamma \cup \{Z'_0\}\} \end{aligned}$$

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

$$\begin{aligned}\delta &= \{(q, a, a, q, \epsilon) \mid a \in \Sigma\} \\ &\cup \{(q, \epsilon, N, q, \bar{\alpha}) \mid (N \rightarrow \alpha) \in P\}\end{aligned}$$

- ▶ 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

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

- ▶ On construit l'automate

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

où δ contient les transitions suivantes :

$$(q, a, a, q, \epsilon)$$

$$(q, b, b, q, \epsilon)$$

$$(q, \epsilon, S, q, bSa)$$

$$(q, \epsilon, S, q, \epsilon)$$

Exemple d'exécution de cet automate

► Transitions

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

► Sur le mot d'entrée **aabb** :

$$\begin{array}{l} (q, aabb, S) \\ (q, aabb, bSa) \\ (qabb, bS,) \\ (q, abb, bbSa) \\ (q, bb, bbS) \\ (q, bb, bb) \\ (q, b, b) \\ (q, \epsilon, \epsilon) \end{array}$$

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

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

- ▶ c'est-à-dire le langage de tous les mots qui permettent l'automate d'aller de l'état q_0 en 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 :

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

- ▶ 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 en 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 $c \in \Sigma \cup \{\epsilon\}$:

$$\{[q, X, q_1] \rightarrow c\}$$

- ▶ car, en consommant le mot c , l'automate peut aller de l'état q en l'état q_1 en supprimant un X du sommet de la pile.

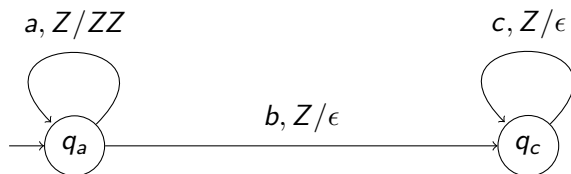
Traduction d'un automate à pile en grammaire

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

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

- ▶ car, en consommant un mot de la forme $cv_1 \dots v_n$, 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 :

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

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

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

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

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

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

$$\begin{aligned} [q_a, Z, q_a] &\rightarrow a[q_a, Z, q_a][q_a, Z, q_a] \mid a[q_a, Z, q_c][q_c, Z, q_a] \\ [q_a, Z, q_c] &\rightarrow a[q_a, Z, q_a][q_a, Z, q_c] \mid a[q_a, Z, q_c][q_c, Z, q_c] \end{aligned}$$

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 :

$$S \rightarrow [q_a, Z, q_c]$$

$$[q_a, Z, q_c] \rightarrow a[q_a, Z, q_c][q_c, Z, q_c] \mid b$$

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

- ▶ ou, après simplification :

$$S \rightarrow [q_a, Z, q_c] \quad [q_a, Z, q_c] \rightarrow a[q_a, Z, q_c]c \mid b$$

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 ...	Formalisme générateur	Formalisme analyseur
... réguliers	Expressions rationnelles	Automates finis
... algébriques	Grammaires algébriques	Automates à pile

Clôture par intersection avec des langages rationnels

- ▶ Les langages algébriques sont clos par intersection avec des langages rationnels.
- ▶ C'est-à-dire : Si L_1 est algébrique et L_2 est rationnel, 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 rationnels

- ▶ 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

- ▶ $(q_1, \epsilon, X, q'_1, \alpha) \in \delta_1$

Intersection de deux langages algébriques ?

- ▶ N'est-ce pas une contradiction avec le fait que la classe des langages algébriques n'est pas close sous intersection ?
- ▶ Si on essaye d'appliquer la construction pour l'intersection d'un langage algébrique et d'un rationnel 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.

Et si on permettait deux piles ?

- ▶ Si on permettait deux piles (ou plusieurs), alors l'automate à deux piles pourrait reconnaître $\{a^n b^n c^n \mid n \geq 0\}$.
- ▶ Idée :
 - ▶ Dans la première phase on lit des a et met chaque fois un A sur la première pile, at aussi la deuxième pile.
 - ▶ Dès qu'on trouve un b on va dans un deuxième état où on va dépiler un A de la première pile pour chaque b lu.
 - ▶ Dès qu'on trouve un c on va dans un troisième état où on va dépiler un A de la deuxième pile pour chaque c lu.
- ▶ Or, nous avons vu au cours 9 que ce langage n'est pas algébrique.

La hiérarchie de Chomsky (-Schützenberger)

- ▶ Type 3 : langages rationnelles
- ▶ Type 2 : langages algébriques
- ▶ Type 1 : langages *contextuels* - pas fait dans ce cours
- ▶ Type 0 : langages récursivement énumérables : tout langage L pour lequel il y a un algorithme, qui lit un mot w et dit “oui” quand $w \in L$, et qui dit “non” ou ne termine pas quand $w \notin L$.
- ▶ pour cette dernière classe, voir le cours de *calculabilité* en M1

Relation entre ces classes

