THL - Janvier 2019 - PROMO 2021

Ce qui est en rose est pas sûr, à vérifier

1. Automates finis

**Expression à automate de Thompson**: Soit une expression de n symboles, alors l’automate de Thompson la représentant aura **2n** états.

**Equivalence**: Deux automates finis sont équivalents si et seulement s’ils reconnaissent le même langage

**NFA**: Non Deterministic Finite Automaton

* Plusieurs transitions peuvent lier deux états
* Une transition ne consomme pas nécessairement de symbol (transition spontanée)

**DFA**: Deterministic Finite Automaton

* une transition est définie de manière unique par son symbole en entrée
* une transition consomme nécessairement un symbol (pas de tr spontannée)

**Déterminisation**: Pour tout NFA à n états on peut construire un DFA équivalent avec au plus 2^n états.

Algorithme d’existence

Soit A un automate fini, il existe un algorithme permettant de décider si :

* L(A) est vide
* L(A) est fini / infini

Soient A1 et A2 deux automates finis. Il existe une procédure permettant de décider si A1 et A2 sont équivalents.

2. Langage rationnel

Langage rationnel = langage régulier = langage reconnaissable

* Un langage est reconnaissable s’il existe un automate fini qui le reconnaît
* Un langage est régulier s’il existe une expression rationnelle dénotant le langage
* Définit sur une ou plusieurs combinaison d’un alphabet, de l’ensemble vide et des opérations rationnelles

Opérations rationnelles

* Union
* Concaténation
* Etoile de Kleen

Clôture rationnelle (L1 et L2 deux langages rationnels)

* Complémentarité: Le langage complémentaire d’un L rat est rat.
* Union: L1 U L2 rationnel
* Intersection: L1 ∩ L2 rationnel
* Mirroire
* Concaténation: L1 . L2 rationnel
* Etoile de Kleen: L1\* rationnel

3. Hiérarchie de Chomsky

Type 0 **Grammaire Syntagmatique**

* Production non contraire
* Décrit un langage reconnaissable par une Machine de Turing
* Engendre un **langage récursivement énumérable**

Remarque:

* Langage récursif =/= langage récursivement énumérable
* Tous les langages récursifs sont récursivement énumérable, mais la réciproque est fausse.

Type 1 **Grammaire Monotone**

* La partie droite d’une production est nécessairement plus longue (ou =) que la partie gauche
* On ne peut pas engendrer le mot vide

**Grammaire Contextuelle**

* Chaque production ne réécrit qu’un symbol non-terminal à la fois, les autres (= contexte) restent inchangés.

Propriétés:

* Si G est une grammaire monotone, il existe une grammaire monotone équivalente dans laquelle toutes les parties droites ont une longueur inférieure ou égale à 2.
* Il existe une grammaire contextuelle équivalente à toute grammaire monotone.
* Les **langages contextuel** sont engendrés par les grammaires contextuelles et les grammaires monotones.
* Tout langages contextuels est récursif.

Type 2 **Grammaire Hors-contexte**

* Chaque production ne réécrit qu’un symbol non-terminal à la fois, et le contexte doit être vide.

Type 3 **Grammaire Linéaire**

* Toute partie droite d’une production contient au plus un symbole non-terminal
  + Grammaire linéaires gauche: le non-terminal se trouve au début de la partie droite de la production.
  + Grammaire linéaires droite: le non-terminal se trouve à la fin de la partie droite de la production.
  + Grammaire linéaire ni à droite, ni à gauche: le non-terminal est au mileu d’un ou plusieurs terminaux.

**Grammaire Rationnelle**

* Grammaire linéaire à droite ou à gauche

Type 4 **Grammaire à Choix Finis**

* Tout non-terminal ne réécrit que des terminaux

4. Analyseurs

a. Analyseurs descendants et ascendant

Un analyseur syntaxique doit retracer le cheminement d'application des règles de syntaxe qui ont mené d’une production au texte analysé.

**Analyseur descendant**

* Retrace la dérivation en partant de la production et en essayant d'appliquer les règles pour retrouver le texte.
* Cette analyse procède en morcelant la phrase en éléments de plus en plus réduits jusqu'à atteindre les unités lexicales (production terminale).
* **Analyseur LL**
* Impossible d’avoir une récursion à gauche

**Analyseur ascendant**

* Retrouve la dérivation en partant du texte, en essayant de déduire les productions du niveau du haut de la grammaire en les construisant à partir des feuilles (production terminale).
* **Analyseur LR**

b. Analyseurs déterministes

Un analyseur syntaxique est déterministe si une seule règle de réécriture est applicable dans chaque configuration de l'analyseur.

=> une seule séquence de règles permettant d'analyser le texte dans sa totalité

=> ne peut être syntaxiquement ambigu.

analyse descendante + déterministe = prédictive.

c. Analyseur LL(k)

* Analyse descendante
* Analyse de gauche à droite (**Left to right**)
* k de regard avant : on peut construire une table où il n’y a qu’une dérivation possible en regardant k caractères
* Dérivation gauche: dérivation du non terminal à gauche (**Leftmost derivation**)
* Applicable sur un sous ensemble de grammaire hors contexte : **Grammaire LL**

**Il existe des grammaires non-ambiguës, pour lesquelles aucun regard avant de taille borné ne permettra une analyse descendante déterministe.**

Repose sur une table de prédictions: FIRST, FOLLOW, NULL

* **FIRST:** l’ensembles des symboles terminaux pouvant apparaître en tête d’une dérivation gauche On ne prend pas en compte le mot vide
* **FOLLOW**: l’ensemble des symboles terminaux pouvant suivre un non terminal dans la dérivation gauche d’une production.
* **NULL** Les manières de produir le mot vide

Grammaire LL

* SLL(1) Grammaire **simple** LL : tout non-terminal réécrit comme premier symbole un symbole terminal.

Remarque: Toute grammaire LL(1) est non ambigüe

Sources

<https://fr.wikipedia.org/wiki/Analyseur_LR>

<https://epiportal.com/ING/Module%20IF%20-%20Sciences%20de%20base%2C%20Informatique%20Fondamentale/THL%20-%20Theorie%20des%20langages/cours/theorie-des-langages-notes-de-cours.pdf>

<http://www.linguist.univ-paris-diderot.fr/~amsili/Ens16/pdf/poly-me03li-2.pdf>

<https://fr.wikipedia.org/wiki/Langage_r%C3%A9cursif>