

Agrégation

Xavier MONTILLET

8 mai 2016

Notations :

- ▶ todo (pour les oraux blancs)
- ✓ ok
- 🤔 bof (à changer si possible)
- ✗ pas ok

Première partie

Informatique

Table des matières

I	Informatique	3
1	Leçons	5
1.1	901 - ► Structures de données : exemples et applications. (RECOMPILE/2)	5
1.2	902 - ► Diviser pour régner : exemples et applications. (RECOMPILE/2)	6
1.3	903 - ► Exemples d'algorithmes de tri. Complexité. (RECOMPILE/2)	6
1.4	906 - ► Programmation dynamique : exemples et applications. (RECOMPILE/2)	8
1.5	907 - ► Algorithmique du texte : exemples et applications. (RECOMPILE/2)	8
1.6	909 - ► Langages rationnels. Exemples et applications. (RECOMPILE/2)	9
1.7	910 - ► Langages algébriques. Exemples et applications. (RECOMPILE/2)	9
1.8	912 - ► Fonctions récursives primitives et non primitives. Exemples. (RECOMPILE/2)	9
1.9	913 - ► Machines de Turing. Applications. (RECOMPILE/2)	9
1.10	914 - Décidabilité et indécidabilité. Exemples. (RECOMPILE/2)	10
1.11	915 - Classes de complexité : exemples. (RECOMPILE/2)	10
1.12	916 - ► Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications. (RECOMPILE/2)	10
1.13	917 - ► Logique du premier ordre : syntaxe et sémantique. (RECOMPILE/2)	11
1.14	918 - Systèmes formels de preuve en logique du premier ordre : exemples. (RECOMPILE/2)	12
1.15	919 - Unification : algorithmes et applications. (RECOMPILE/2)	12
1.16	920 - Réécriture et formes normales. Exemples. (RECOMPILE/2)	12
1.17	921 - ► Algorithmes de recherche et structures de données associées. (RECOMPILE/2)	13
1.18	922 - ► Ensembles récursifs, récursivement énumérables. Exemples. (RECOMPILE/2)	13
1.19	923 - ► Analyses lexicale et syntaxique : applications. (RECOMPILE/2)	14
1.20	924 - Théories et modèles en logique du premier ordre. Exemples. (RECOMPILE/2)	14
1.21	925 - ► Graphes : représentations et algorithmes. (RECOMPILE/2)	14
1.22	926 - Analyse des algorithmes : complexité. Exemples. (RECOMPILE/2)	14

1.23	927 - ➤ Exemples de preuve d'algorithme : correction, terminaison. (RECOMPILE/2)	15
1.24	928 - Problèmes NP-complets : exemples de réductions (RECOMPILE/2)	15
2	Développements	17
2.1	Théorème de Cook	17
2.2	Tri par Tas	17
2.3	Unification	17
2.4	Plus longue sous-séquence commune	17
2.5	Les points les plus proches	18
2.6	Bellman-Ford	18
2.7	Tri topologique	18
2.8	Arbres binaires de recherche optimaux	18
2.9	Décidabilité de l'arithmétique de Presburger	19
2.10	Calculable implique récursif	19
2.11	DPLL	19
2.12	FFT	19
3	Références	20
3.1	910 - ➤ Langages algébriques. Exemples et applications. (RECOMPILE/2)	21
3.2	Représentations	21
3.3	Propriétés	24
II	Mathématiques	25

Chapitre 1

Leçons

1.1 901 - ➤ Structures de données : exemples et applications. (3/2)




1.1.1 Rapport du jury

Le mot *algorithme* ne figure pas dans l'intitulé de cette leçon, même si l'utilisation des structures de données est évidemment fortement liée à des questions algorithmiques.

La leçon doit donc être orientée plutôt sur la question du choix d'une structure de données que d'un algorithme. Le jury attend du candidat qu'il présente différents types abstraits de structures de données en donnant quelques exemples de leur usage avant de s'intéresser au choix de la structure concrète. Le candidat ne peut se limiter à des structures linéaires simples comme des tableaux ou des listes, mais doit présenter également quelques structures plus complexes, reposant par exemple sur des implantations à l'aide d'arbres.

(*Rapport du jury.* 2015)

1.1.2 Développements

-  Tri par Tas
-  Arbres binaires de recherche optimaux
-  FFT

1.1.3 Références

Cormen, BBC, Froidevaux, Papadimitriou

1.1.4 Plan

- Défs BBC (type concret, type abstrait, structure de données)
- Structures de base (tableau, liste, arbres binaires)
- Ordonnancement (piles, files, files de priorité)
- Ensembles et dictionnaires

1.1.5 Remarques

Mentionner les implémentations naïves, et faire une remarques générale sur les implémentations implicites et paresseuses

1.2 902 - ➤ Diviser pour régner : exemples et applications. (2/2)

1.2.1 Rapport du jury

Cette leçon permet au candidat de proposer différents algorithmes utilisant le paradigme *diviser pour régner*. Le jury attend du candidat que ces exemples soient variés et touchent des domaines différents.

Un calcul de complexité ne peut se limiter au cas où la taille du problème est une puissance exacte de 2, ni à une application directe d'un théorème très général recopié approximativement d'un ouvrage de la bibliothèque de l'agrégation. *(Rapport du jury. 2015)*

1.2.2 Développements



Les points les plus proches



FFT

1.2.3 Références

Cormen, BBC, Papadimitriou

1.2.4 Plan

- Paradigme et premiers exemples (et Master Theorem)
- Tri et recherche
- Multiplication (Karatsuba, Strassen, FFT)
- Géométrie algorithmique
- Applications théoriques (Savitch, multiplication de matrice -> inversion)

1.2.5 Remarques

Graphe de dépendances acyclique (ou presque)

1.3 903 - ➤ Exemples d'algorithmes de tri. Complexité. (2/2)

1.3.1 Rapport du jury

Sur un thème aussi classique, le jury attend des candidats la plus grande précision et la plus grande rigueur.

Ainsi, sur l'exemple du tri rapide, il est attendu du candidat qu'il sache décrire avec soin l'algorithme de partition et en prouver la correction et que l'évaluation des complexités dans le cas le pire et en moyenne soit menée avec rigueur.

On attend également du candidat qu'il évoque la question du tri en place, des tris stables, ainsi que la représentation en machine des collections triées.

Le jury ne manquera pas de demander au candidat des applications non triviales du tri. (Rapport du jury. 2015)

1.3.2 Développements



Tri par Tas

Tri topologique

1.3.3 Références

Cormen, BBC, Papadimitriou

1.3.4 Plan

- Intro (tri, applications, stable, en place, [on se restreint aux entiers?], on compte le nombre de comparaisons, borne inférieure)
- Tris naïfs
- Tri par tas
- Diviser pour régner
- Autres tris (base, paquet, dénombrement, mémoire externe?)

1.3.5 Remarques

Quand on trie, on trie par rapport à un *préordre* \preceq . On a donc aussi une relation d'équivalence $x \approx y := (x \preceq y) \wedge (x \succeq y)$ qui n'a aucune raison d'être l'égalité.

Si l'on note \sqsubseteq le préordre associé au tableau donné en entrée d'un tri (donc $T[i] \sqsubseteq T[j] \iff i \leq j$), le fait qu'une procédure de tri soit *stable* veut dire que trier par rapport à \preceq ou par rapport au préordre lexicographique associé à (\preceq, \sqsubseteq) , que l'on note $\preceq_{\preceq, \sqsubseteq}^{lex}$, revient au même. En particulier, si on a une procédure de tri instable, on peut en construire une stable (mais le tri ne peut alors plus être *en place* car pour pouvoir savoir si $x \sqsubseteq y$, on a besoin de stocker de tableau d'entrée et on est donc forcé de travailler sur une copie).

La notion de tri *stable* n'a aucun intérêt si on se restreint aux entiers muni de leur ordre habituel (ou plus généralement à un ensemble muni d'un ordre [par opposition à un préordre]) car si l'on ne peut jamais distinguer $T[i]$ de $T[j]$, s'assurer qu'ils sont restés dans le même ordre n'est pas vraiment utile.

Pour trier selon $\preceq_{\preceq_1, \preceq_2}^{lex}$, on peut d'abord trier selon \preceq_2 et ensuite utiliser une procédure de tri stable pour trier selon \preceq_1 (ce qui peut être plus facile à implémenter que le tri selon $\preceq_{\preceq_1, \preceq_2}^{lex}$ et peut servir si l'on doit trier selon $\preceq_{\preceq_1, \preceq_2}^{lex}$ et $\preceq_{\preceq_1', \preceq_2}^{lex}$).

Dans le tri rapide, on peut choisir la médiane en $O(n)$ comme pivot et donc atteindre un pire cas en $O(n \log n)$ mais en pratique, c'est plus lent.




1.4 906 - ➤ Programmation dynamique : exemples et applications. (3/2)

1.4.1 Rapport du jury

Même s'il s'agit d'une leçon d'exemples et d'applications, le jury attend des candidats qu'ils présentent les idées générales de la programmation dynamique et en particulier qu'ils aient compris le caractère générique de la technique de mémorisation. Le jury appréciera que les exemples choisis par le candidat couvrent des domaines variés, et ne se limitent pas au calcul de la longueur de la plus grande sous-séquence commune à deux chaînes de caractères.

Le jury ne manquera pas d'interroger plus particulièrement le candidat sur la question de la correction des algorithmes proposés et sur la question de leur complexité en espace. (*Rapport du jury*. 2015)

1.4.2 Développements

-  Plus longue sous-séquence commune
-  Bellman-Ford
-  Arbres binaires de recherche optimaux

1.5 907 - ➤ Algorithmique du texte : exemples et applications. (0/2)

1.5.1 Rapport du jury

Cette leçon devrait permettre au candidat de présenter une grande variété d'algorithmes et de paradigmes de programmation, et ne devrait pas se limiter au seul problème de la recherche d'un motif dans un texte, surtout si le candidat ne sait présenter que la méthode naïve.

De même, des structures de données plus riches que les tableaux de caractères peuvent montrer leur utilité dans certains algorithmes, qu'il s'agisse d'automates ou d'arbres par exemple.

Cependant, cette leçon ne doit pas être confondue avec la 909 : *Langages rationnels. Exemples et applications* ni avec la 910 : *Langages algébriques. Exemples et applications*.

La compression de texte peut faire partie de cette leçon si les algorithmes présentés contiennent effectivement des opérations comme les comparaisons de chaînes : la compression LZW, par exemple, ressortit davantage à cette leçon que la compression de Huffman.

(*Rapport du jury*. 2015)

1.5.2 Développements

Aucun.

1.6 909 - ➤ Langages rationnels. Exemples et applications. (0/2)

1.6.1 Rapport du jury

Des applications dans le domaine de la compilation entrent naturellement dans le cadre de ces leçons. (*Rapport du jury*. 2015)

1.6.2 Développements

Aucun.

1.7 910 - ➤ Langages algébriques. Exemples et applications. (0/2)

1.7.1 Développements

Aucun.

1.8 912 - ➤ Fonctions récursives primitives et non primitives. Exemples. (2/2)

1.8.1 Développements



Décidabilité de l'arithmétique de Presburger



Calculable implique récursif

1.8.2 Références

Wolper, Carton

1.8.3 Plan

- Fonctions primitives récursives
- (Limites du modèle) (cardinal, Ackermann)
- Fonctions récursives (calculables, casto affairé)

1.8.4 Remarques

Aucunes.

1.9 913 - ➤ Machines de Turing. Applications. (2/2)

1.9.1 Développements



Théorème de Cook



Calculable implique récursif

1.9.2 Références

Wolper, Carton

1.9.3 Plan

- Défs (défs équivalentes, codage, machine universelle)
- Calculabilité et décidabilité (RE, R, Rice)
- Complexité

1.9.4 Remarques

Aucunes.

1.10 914 -Décidabilité et indécidabilité. Exemples. (0/2)

1.10.1 Développements

Aucun.

1.11 915 -Classes de complexité : exemples. (0/2)

1.11.1 Rapport du jury

Le jury attend que le candidat aborde à la fois la complexité en temps et en espace. Il faut naturellement exhiber des exemples de problèmes appartenant aux classes de complexité introduites, et montrer les relations d'inclusion existantes entre ces classes.

Le jury s'attend à ce que le caractère strict ou non de ces inclusions soit abordé, en particulier le candidat doit être capable de montrer la non-appartenance de certains problèmes à certaines classes.

Parler de décidabilité dans cette leçon serait hors sujet.

(Rapport du jury. 2015)

1.11.2 Développements

Aucun.

1.12 916 - ➤ Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications. (2/2)

1.12.1 Rapport du jury

Le jury attend des candidats qu'ils abordent les questions de la complexité de la satisfiabilité.

Pour autant, les applications ne sauraient se réduire à la réduction de problèmes NP-complets à SAT.

Une partie significative du plan doit être consacrée à la représentation des formules et à leurs formes normales. (*Rapport du jury*. 2015)

1.12.2 Développements



Théorème de Cook



DPLL

1.12.3 Références

- Huth, Ryan (tableau avec différentes représentations p. 361, ROBDD)
- Stern (Tseitin)
- Devisme, Lafourcade, Lévy (Tout le reste, en particulier DPLL)

1.12.4 Plan

- Syntaxe (lecture unique, notation parenthésée, notation polonaise, arbre, circuit)
- Sémantique (compacité, représentations [table, formes normales, OBDD])
- Problèmes associés (SAT, Valie, DPLL, Tseitin)

1.12.5 Remarques

Aucunes.

1.13 917 - ➤ Logique du premier ordre : syntaxe et sémantique. (2/2)

1.13.1 Rapport du jury

La question de la syntaxe dépasse celle de la définition des termes et des formules. Elle comprend aussi celle des règles de la démonstration.

Le jury attend donc du candidat qu'il présente au moins un système de preuve et les liens entre syntaxe et sémantique, en développant en particulier les questions de correction et complétude.

(*Rapport du jury*. 2015)

1.13.2 Développements



Unification

Décidabilité de l'arithmétique de Presburger

1.13.3 Références

- David Nour Raffali
- Cori Lascar
- CK ?

1.13.4 Plan

- qwe

1.13.5 Remarques

Aucunes.

1.14 918 -Systèmes formels de preuve en logique du premier ordre : exemples. (0/2)

1.14.1 Développements

Aucun.

1.15 919 -Unification : algorithmes et applications. (0/2)

1.15.1 Développements

Aucun.

1.16 920 -Réécriture et formes normales. Exemples. (0/2)

1.16.1 Rapport du jury

Au-delà des propriétés standards (terminaison, confluence) des systèmes de réécriture, le jury attend notamment du candidat qu'il présente des exemples sur lesquels l'étude des formes normales est pertinente dans des domaines variés : calcul formel, logique, etc.

Un candidat ne doit pas s'étonner que le jury lui demande de calculer des paires critiques sur un exemple concret.

Lorsqu'un résultat classique comme le lemme de Newman est évoqué, le jury attend du candidat qu'il sache le démontrer.

(Rapport du jury. 2015)

1.16.2 Développements

Aucun.

1.17 921 - ➤ Algorithmes de recherche et structures de données associées. (2/2)

1.17.1 Développements



Les points les plus proches



Arbres binaires de recherche optimaux

1.17.2 Références

— Cormen

1.17.3 Plan

- Liste et tableau (recherche, rang, min, max)
- ABR (AVL, ABR optimaux)
- Tables de Hachage

1.17.4 Remarques

Demander à Simon pourquoi ABR optimaux \rightarrow Huffman (l'ordre des clés factices est conservé dans ABR optimaux mais l'ordre des clés n'est pas conservé dans Huffman)

1.18 922 - ➤ Ensembles récursifs, récursivement énumérables. Exemples. (2/2)

1.18.1 Développements



Décidabilité de l'arithmétique de Presburger



Calculable implique récursif

1.18.2 Références

- Carton
- Wolper
- Sipser
- Lassaigne Rogemont (Logique et fondements de l'informatique)

1.18.3 Plan

- Fonction récursives et machines de Turing
- Ensembles récursifs et récursivement énumérables (équivalences, énumérateur, propriétés de clôture)
- Exemples et applications (langages, PCP, théories)

1.18.4 Remarques

Aucunes.

1.19 923 - ► Analyses lexicale et syntaxique : applications. (0/2)

1.19.1 Développements

Aucun.

1.20 924 -Théories et modèles en logique du premier ordre. Exemples. (0/2)

1.20.1 Développements

Aucun.

1.21 925 - ► Graphes : représentations et algorithmes. (2/2)

1.21.1 Développements



Bellman-Ford

Tri topologique

1.21.2 Références

— Cormen

1.21.3 Plan

- Défs
- Représentations
- Parcours
- Chemins
- Problèmes de réseaux

1.21.4 Remarques

Aucunes.

1.22 926 -Analyse des algorithmes : complexité. Exemples. (0/2)

1.22.1 Développements

Aucun.

1.23 927 - ➤ Exemples de preuve d'algorithme : correction, terminaison. (2/2)

1.23.1 Rapport du jury

Le jury attend du candidat qu'il traite des exemples d'algorithmes récursifs et des exemples d'algorithmes itératifs.

En particulier, le candidat doit présenter des exemples mettant en évidence l'intérêt de la notion d'invariant pour la correction partielle et celle de variant pour la terminaison des segments itératifs.

Une formalisation comme la logique de Hoare pourra utilement être introduite dans cette leçon, à condition toutefois que le candidat en maîtrise le langage. *(Rapport du jury. 2015)*

1.23.2 Développements



Tri par Tas



Unification

1.23.3 Références

- Cormen
- Winskel
- Luc Albert
- David Nour Raffali

1.23.4 Plan

```
let f (x : int ref) (y : int ref) =  
  x := !x lxor !y;  
  y := !x lxor !y;  
  x := !x lxor !y
```

- Terminaison (ensemble bien fondé, itératif et récursif)
- Correction (itératif et récursif)
- Sémantique axiomatique

1.23.5 Remarques

Aucunes.

1.24 928 -Problèmes NP-complets : exemples de réductions (0/2)

1.24.1 Rapport du jury

L'objectif ne doit pas être de dresser un catalogue le plus exhaustif possible ; en revanche, pour chaque exemple, il est attendu que le candidat puisse au moins expliquer clairement le problème considéré, et indiquer de quel autre problème une réduction permet de prouver sa NP-complétude.

Les exemples de réduction seront autant que possible choisis dans des domaines variés : graphes, arithmétique, logique, etc. Un exemple de problème NP-complet dans sa généralité qui devient P si on contraint davantage les hypothèses pourra être présenté, ou encore un algorithme P approximant un problème NP-complet.

Si les dessins sont les bienvenus lors du développement, le jury attend une définition claire et concise de la fonction associant, à toute instance du premier problème, une instance du second ainsi que la preuve rigoureuse que cette fonction permet la réduction choisie.

(Rapport du jury. 2015)

1.24.2 Développements

Aucun.

Chapitre 2

Développements

2.1 Théorème de Cook

Leçons :



913 - ➤ Machines de Turing. Applications. (2/2)



916 - ➤ Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications. (2/2)

Wolper (et Carton)

2.2 Tri par Tas

Leçons :



901 - ➤ Structures de données : exemples et applications. (3/2)



903 - ➤ Exemples d'algorithmes de tri. Complexité. (2/2)



927 - ➤ Exemples de preuve d'algorithme : correction, terminaison. (2/2)

Cormen

Remarques :

- n pour taille du tas et l pour longueur de tableau Faire un dessin.
- Dire que la série converge sans le justifier.

2.3 Unification

Leçons :



917 - ➤ Logique du premier ordre : syntaxe et sémantique. (2/2)



927 - ➤ Exemples de preuve d'algorithme : correction, terminaison. (2/2)

2.4 Plus longue sous-séquence commune

Leçons :



906 - ➤ Programmation dynamique : exemples et applications. (3/2)



Cormen

Remarques :

- Lemme, relation de récurrence et algorithme (et exemple si c'est trop court).
- Il faut juste changer les notations ($\tilde{u}u_{\bullet} = u$)



2.5 Les points les plus proches

Leçons :

-  902 - ➤ Diviser pour régner : exemples et applications. (2/2)
-  921 - ➤ Algorithmes de recherche et structures de données associées. (2/2)

2.6 Bellman-Ford

Leçons :

-  906 - ➤ Programmation dynamique : exemples et applications. (3/2)
-  925 - ➤ Graphes : représentations et algorithmes. (2/2)

Remarques :

- Bien différentier taille et poids d'un chemin
- *Le* poids minimal, *un* chemin de poids minimal

2.7 Tri topologique

Leçons :

-  903 - ➤ Exemples d'algorithmes de tri. Complexité. (2/2)
-  925 - ➤ Graphes : représentations et algorithmes. (2/2)




Cormen

Remarques :

- Parcours en profondeur avec date
- intervalles de dates disjoint ou inclusion (on suppose $u.d < v.d$ et après cas selon $u.f < v.d$ ou non)
- si (u, v) arête alors $u.f > v.f$ (cas selon la couleur de v quand on passe par (u, v))
- exemple ?




2.8 Arbres binaires de recherche optimaux

Leçons :

-  901 - ➤ Structures de données : exemples et applications. (3/2)
-  906 - ➤ Programmation dynamique : exemples et applications. (3/2)
-  921 - ➤ Algorithmes de recherche et structures de données associées. (2/2)




2.9 Décidabilité de l'arithmétique de Presburger

Leçons :

-  912 - ➤ Fonctions récursives primitives et non primitives. Exemples. (2/2)
-  917 - ➤ Logique du premier ordre : syntaxe et sémantique. (2/2)
-  922 - ➤ Ensembles récursifs, récursivement énumérables. Exemples. (2/2)

2.10 Calculable implique récursif

Leçons :


-  912 - ➤ Fonctions récursives primitives et non primitives. Exemples. (2/2)
-  913 - ➤ Machines de Turing. Applications. (2/2)
-  922 - ➤ Ensembles récursifs, récursivement énumérables. Exemples. (2/2)

Remarques :

- Carton pour les formules
- Wolper four nom des fonctions intermédiaires et vision globale
- Il faut prendre $f : \Sigma^* \rightarrow \Gamma^*$ et à la fin, rajouter une fonction pour transformer (u, q, v) et ε, q, v' pour pouvoir dire que le résultat est v' .



2.11 DPLL

Leçons :

-  916 - ➤ Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications. (2/2)

2.12 FFT

Leçons :

-  901 - ➤ Structures de données : exemples et applications. (3/2)
-  902 - ➤ Diviser pour régner : exemples et applications. (2/2)

Chapitre 3

Références

Livres a priori cools : KOZEN [1]

Bibliographie

- [1] Dexter C. KOZEN. *The Design and Analysis of Algorithms*. New York, NY, USA : Springer-Verlag New York, Inc., 1992. ISBN : 0-387-97687-6.
- [2] *Rapport du jury*. 2015. URL : http://cache.media.education.gouv.fr/file/agreg_ext/30/1/math_470301.pdf.

lassaigne rougeameont
froidevaux godel soria
cori lascar (1 ET 2)
dehornoy
informatique theorique (DEVISME, LAFOURCADE LEVY) : DPLL (et mieux pr calcul prop ?)
Hutt, Ryan (BDD)

3.0.1 Mémoire

3.1 910 - ➤ Langages algébriques. Exemples et applications. (0/2)

Les langages algébriques sont assez complexes pour pouvoir représenter la plupart des langages de programmations actuels tout en étant assez simples pour que certaines propriétés restent décidables. Ils ont de nombreuses applications en compilation.

Dans cette leçon, nous présentons d'abord les représentations usuelles des langages algébriques et quelques-unes de leurs propriétés. Nous constatons ensuite que
TODO

3.2 Représentations

3.2.1 Grammaires algébriques

Définition et premières propriétés

Définition 1 (Grammaire algébrique). Une grammaire algébrique est un quadruplet (V, Σ, R, S) où V et Σ sont des ensembles finis disjoints, $R \subseteq V \times (V \sqcup \Sigma)^*$ et $S \in V$. Les éléments de V sont appelés variables ou non terminaux et ceux de Σ sont appelés terminaux. Les éléments de R sont appelés règles. Le non terminal S est appelé axiome. On note $X \rightarrow u_1 + \dots + u_n$ pour $(X, u_1), \dots, (X, u_n) \in R$.

Exemple 2. $G_1 = (\Sigma_1, V_1, P_1)$, $A_1 = \{a, b\}$, $V_1 = \{S\}$, $R_1 = \{S \rightarrow aSb, S \rightarrow \varepsilon\}$

Remarque 3. On se contentera souvent de donner R en utilisant la convention que les lettres minuscules sont des symboles terminaux et que les lettres majuscules sont des symboles non terminaux.

Exemple 4. $G_1 = \{S \rightarrow aSb + \varepsilon\}$, $G_2 = \left\{ \begin{array}{l} P \rightarrow aI + \varepsilon \\ I \rightarrow aP \end{array} \right\}$

$$G_{3,n} = \left\{ \begin{array}{l} S \rightarrow ST + \varepsilon \\ T \rightarrow a_1 S b_1 S + \dots + a_n S b_n S + \varepsilon \end{array} \right\}$$

Définition 5 (Dérivation). On étend \rightarrow à $(V \sqcup \Sigma)^*$ par $\alpha X \beta \rightarrow \alpha u \beta$ si $X \rightarrow u$. On note \rightarrow^* la clôture réflexive transitive de \rightarrow . Si $u \rightarrow^* v$, on dit que u se dérive directement en v .

Exemple 6. $S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$ dans G_1

Définition 7 (Langage engendré). Pour tout $X \in V$, on pose $\widehat{\mathcal{L}}_G(u) := \{v \in (V \sqcup \Sigma)^* : u \rightarrow^* v\}$ et $\mathcal{L}_G(u) := \widehat{\mathcal{L}}_G(u) \cap \Sigma^*$. On appelle $\mathcal{L}_G(u)$ langage engendré par u dans G . On appelle langage engendré par la grammaire et note $\mathcal{L}(G)$ le langage $\mathcal{L}_G(S)$ engendré par l'axiom.

Exemple 8. $\mathcal{L}_{G_1}(S) = \{a^n b^n \mid n \in \mathbb{N}\}$, $\mathcal{L}_{G_2}(P) = \{a^{2n} \mid n \in \mathbb{N}\}$, $\mathcal{L}_{G_2}(I) = \{a^{2n+1} \mid n \in \mathbb{N}\}$, $D_n^* := \mathcal{L}_{G_{3,n}}(S)$ est appelé langage de Dyck. C'est le langage des mots bien parenthésés (si l'on considère a_i comme une parenthèse ouvrante et b_i comme la parenthèse fermante correspondante).

Définition 9 (Langage algébrique). Un langage algébrique est un langage engendré par une grammaire algébrique.

Lemme 10 (Fondamental). Soit $G = (V, \Sigma, R, S)$ une grammaire algébrique et u et v deux mots de $(V \sqcup \Sigma)^*$. On suppose que u se factorise $u = u_1 u_2$. Alors il existe une dérivation $u \rightarrow^k v$ de longueur k si et seulement si v se factorise $v = v_1 v_2$ et s'il existe deux dérivations $u_1 \rightarrow^{k_1} v_1$ et $u_2 \rightarrow^{k_2} v_2$ où $k = k_1 + k_2$.

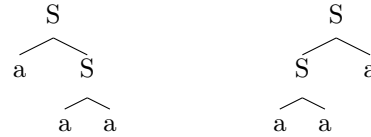
Arbres de dérivation

Définition 11 (Arbre de dérivation). Soit $G = (V, \Sigma, R, S)$ une grammaire algébrique. Un arbre de dérivation est un arbre fini dont les noeuds sont étiquetés par $V \sqcup \Sigma \sqcup \{\varepsilon\}$ vérifiant la propriété suivante. Si X est l'étiquette d'un noeud interne et $\alpha_1, \dots, \alpha_n$ sont les étiquettes de ses fils alors $S \rightarrow \alpha_1 \dots \alpha_n$.

Définition 12 (Frontière). La frontière d'un arbre de dérivation est le mot obtenu par concaténation des étiquettes de ses feuilles de gauche à droite.

Proposition 13. Pour tout non-terminal $X \in V$, le langage $\mathcal{L}_G(X)$ (resp. $\widehat{\mathcal{L}}_G(X)$) est l'ensemble des mots $u \in \Sigma^*$ (resp. $u \in (V \sqcup \Sigma)^*$) tels qu'il existe un arbre de dérivation de racine X et de frontière u .

Exemple 14. Les deux arbres suivants sont des arbres de dérivation (de frontière aaa) de la grammaire $S \rightarrow SS + a$.



Définition 15 (Grammaire ambiguë). Une grammaire G est dite ambiguë s'il existe un mot ayant deux arbres de dérivation distincts dont les racines ont la même étiquette.

Exemple 16. $S \rightarrow SS + a$ est une grammaire ambiguë.

Remarque 17. $S \rightarrow aS + a$ est une grammaire non ambiguë générant le même langage.

Simplifications des grammaires [Carton, p.79-82]

Définition 18. (Grammaire réduite)

Définition 19. (Grammaire propre)

Définition 20. (Forme normale quadratique / de Chomsky)

Définition 21. (Forme normale de Greibach) [Carton, p. 102]

Proposition 22. *Pour toute grammaire G , on a : [Hopcroft, p. 295] {Tout est dans le Carton sauf les complexités}*

- \mathcal{L}_G est engendré par une grammaire réduite de taille $O(|G|)$ calculable en temps $O(|G|)$.
- $\mathcal{L}_G \setminus \{\varepsilon\}$ est engendré par une grammaire propre de taille $O(|G|^2)$ calculable en temps $O(|G|^2)$.
- $\mathcal{L}_G \setminus \{\varepsilon\}$ est engendré par une grammaire en forme normale quadratique de taille $O(|G|^2)$ calculable en temps $O(|G|^2)$.
- $\mathcal{L}_G \setminus \{\varepsilon\}$ est engendré par une grammaire en forme normale de Greibach (quadratique).

3.2.2 Automates à pile [Carton, p. 104-109 ; 112]

Définition 23. (Automate à pile)

Définition 24. (Calcul d'un automate à pile)

Définition 25. (Modes d'acceptation d'un automate à pile)

- Pile vide
- État final

Proposition 26. (*Équivalence des modes d'acceptation*)

Théorème 27. (*Équivalence grammaires algébriques / automates à piles*)

Définition 28. (Automate à pile déterministe)

Remarque 29. Il n'y a plus équivalence des différents modes d'acceptation.

3.3 Propriétés

3.3.1 Lemme d'itération [Carton, p. 92-95]

Lemme 30. (*Ogden*)

Corollaire 31. (*Théorème de Bar-Hillel, Perles et Shamir*)

Affirmation 32. Le langage $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ n'est pas algébrique (par BHS).

Exemple 33. Ce langage pourrait représenter de la mise en forme en mode texte.

```
+-----+
|  titre  |
+-----+
```

Exemple 34. [Application] Le langage $\{a^m b^n c^m d^n \mid n, m \in \mathbb{N}\}$ n'est pas algébrique (par Ogden, BHS ne suffit pas).

Ce langage pourrait représenter le fait qu'on a déclaré deux procédures à m et n arguments et qu'on les a ensuite utilisées. [Dragon, p. 179]

3.3.2 Propriétés de clôture [TODO]

Définition 35. Morphisme

Définition 36. Substitution (algébrique)

Proposition 37. *L'ensemble des langages algébrique est clos par {voir figure ??}.*

3.3.3 Théorème de Chomsky et Schützenberger [Carton, p. 100-101]

Théorème 38. (*Chomsky et Schützenberger*)

Lemme 39. *Il existe un morphisme $\psi : A_n^* \rightarrow A_2^*$ tel que $D_n^* = \psi^{-1}(D_2^*)$.*

Corollaire 40. *Tout langage algébrique s'écrit $\varphi(\psi^{-1}(D_2^* \cap K))$ pour des morphismes φ et ψ et un langage rationnel K .*

Remarque 41. Une fonction $X \mapsto \varphi(\psi^{-1}(X \cap K))$ s'appelle une *transduction rationnelle*. Ce sont des transformations très naturelles qui peuvent être réalisées avec des automates à deux bandes (une pour l'entrée et une pour la sortie).

3.3.4 X - Théorème de Parikh [Carton, p. 86]

Définition 42. (Anagrammes)

Définition 43. (Image commutative)

Théorème 44. (*Parikh*)

3.3.5 Développements

Aucun.

Deuxième partie

Mathématiques