Leçons d'informatique

Manon Ruffini

13 juin 2016

Leçon 901 : Structures de données : exemples et applications.

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.
Motivations : choisir une structure de données adaptée à un problème algorithmique permet de faciliter la conception et/ou d'améliorer la complexité

1 Types de données

1.1 Définitions

Définition 1.1.1. *Type [BBC] p 37 + intérêt*

Exemple 1.1.1.
- *Types de base (entiers, booléens, caractères...) [FG] p 61*
- *Tableau [FG] p 61*
- *Listes chainées [FG] p 70*
- *Listes doublement chainées [FG] p 72*

Définition 1.1.2. *Type de donnée abstrait [BBC] p 38*

Définition 1.1.3. *Structure de données [BBC] p 38*

1.2 Description des types abstraits

Description : signature (sortes + opérations) ; préconditions ; axiomes [FG] p 54-60

Exemple 1.2.1. *Dictionnaire [BBC] p 52-53*
précondition = supprimer ssi chercher = vrai
axiome = estvide(dicovide) = vrai

1.3 Implémentation

Démarche descendante [FG] p 53-54

2 Structures adaptées à la collection d'objets

But : insérer et supprimer des éléments dans un ensemble

2.1 Structures séquentielles

Liste

Définition 2.1.1. *liste [FG] p 63*

Signature [FG] p 64
Implémentation + avantages et inconvénients
- Tableau [FG] p 68
- Liste chainée [FG] p 70

Pile
Définition (liste lifo) + implémentations [FG] p 74
Application : parcours de graphe en profondeur

File
Définition + implémentations [FG] p 77-78
Application : parcours de graphe en largeur

2.2 Structures arborescentes

2.2.1 Graphes

Définition 2.2.1. *Graphe orienté/non-orienté [FG] p 137*

Exemple 2.2.1. *Réseau aérien [FG] p 136*

Signature [FG] p 143
Implémentation : matrice d'adjacence [FG] p 148 ; liste d'adjacence [FG] p 150
Avantages - inconvénients (graphe dense - peu dense) [Cor] p 545-546

Application 2.2.1. *Kosaraju, Dijkstra, Prim, Kruskal [Cor] (table des matières)*

2.2.2 Arbres

Définition 2.2.2. *Arbre [FG] p 140*

Exemple 2.2.2. *Généalogie*

Définition 2.2.3. *Arbre binaire [FG] p 99*

Tas

Définition 2.2.4. *Tas binaire [Cor] p 140*

Signature + implémentation [Cor] p 141-140

Application 2.2.2. *Tri par tas [Cor] p 147*

3 Structures adaptées à la recherche

En plus de l'insertion et de la suppression, on souhaite pouvoir rechercher un élément efficacement.

3.1 Table de hachage

Dans un ensemble d'objets sans ordre

Table à adressage direct[Cor] p 236
déf U,K + dessin ; les opérations se font en $O(1)$;

Hachage[Cor] p 238 pb quand |K| est petit devant |U|

Définition 3.1.1. *fonction, table de hachage ; collision*

Résolution par chaînage Les éléments dans la même alvéole sont placées dans une liste chaînée [Cor] p 239 + dessin
Complexité recherche [Cor] p 241

Application 3.1.1. (*Hachage parfait*) *[Cor] p 258*

3.2 Arbres binaires de recherche

(Dans un ensemble totalement ordonné)

Définition 3.2.1. *Arbre binaire de recherche [Cor] p 268*

Proposition 3.2.1. *Complexité des opérations rechercher, min, max... [Cor] p 273*

Pb quand la hauteur est grande [Cor] p 287

3.3 AVL

Définition 3.3.1. *AVL [FG] p 224-227*

Proposition 3.3.1. *Encadrement de la hauteur [FG] p 225*

Proposition 3.3.2. *Complexité de adjonction/suppression dans les AVL [FG] p 227*

Conclusion : il faut choisir une structure adaptée à ce que l'on veux faire

3.4 ABRO

Explication du problème

Définition 3.4.1. *ABR optimal [Cor]*

(Algo programmation dynamique) [Cor]

Développements

- ABR optimaux
- Hachage parfait

Références

- Beauquier
- Froidevaux
- Cormen

Leçon 902 : Diviser pour régner : exemples et applications.

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.

1 Diviser pour régner : principe

1.1 Description du paradigme

La méthode diviser pour régner résout un problème en suivant les trois étapes suivantes :[Cor] p 59

- Diviser On divise le problème en sous-problèmes qui sont des instances strictement plus petites du même problème
- Régner On résout les sous-problèmes de manière récursive (ou directement si ils sont de taille suffisamment petite)
- Combiner On combine les solutions des sous-problèmes pour produire la solution du problème original

Exemple 1.1.1. *Exponentiation rapide Référence ?*

1.2 Calcul de complexité

Théorème 1.2.1. *Théorème général [Cor] p 86*

En pratique, on utilisera :

Théorème 1.2.2. *Cas particulier : Théorème fondamental [Pap] p 52*

Application 1.2.1. *Calcul de la complexité : relation de récurrence + utilisation du théorème [Pap] p 52*

Exemple 1.2.1. *Cas de l'exponentiation rapide Référence ?*

2 Tri et recherche

2.1 Recherche de la médiane dans un tableau de nombres non trié

En fait, on s'intéressera à un problème plus général : trouver l'élément de rang k dans un tableau non trié. [Cor] p 197 Entrée ; Sortie ; Diviser ; Régner ; Combiner ; Complexité en pire cas et en moyenne (espérance)

2.2 Tri fusion

Entrée ; Sortie (autre tableau alloué ! ! !) ; Diviser ; Régner ; Combiner ; Complexité (formule de récu + valeur) [Cor] p 26

2.3 Tri rapide

(méthode probabiliste) [Cor] p 157

Analyse du tri rapide [Cor]

3 Multiplications

3.1 De nombres : Karatsuba

[Pap] p 49

3.2 De matrices : Strassen

[Cor] p 72

3.3 De polynômes : Transformée de Fourier rapide

[Cor] p 827

4 Géométrie

4.1 Points les plus proches

Algorithme correction et complexité [Cor] p 256

4.2 Enveloppe convexe

[PS] p 112

4.3 Voronoï

[PS] p 195

Développements

- Analyse du tri rapide
- Points les plus proches

Références

- Cormen
- Papadimitriou
- Preparata Shamos
- Froidevaux

Leçon 903 : Exemples d'algorithmes de tri. Complexités.

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.

1 Introduction

1.1 Énoncé du problème

Entrée ; Sortie [FG] p 305

Dans la suite, on se restreint à des tableaux de nombres.

1.2 Définitions

Définition 1.2.1. *Tri stable [FG] p 307*

Définition 1.2.2. *Tri en place [FG] p 139*

On s'intéresse à la complexité en temps ; pour cela, on compte le nombre de comparaisons.

1.3 Applications

1.3.1 Trier une base de données

Exemple 1.3.1. *Librairie Java*

1.3.2 Enveloppe convexe

Algorithme déterministe [BY] p 185

1.3.3 Algorithme du peintre

[dB] p 6

2 Tris naïfs

2.1 Tri par sélection

On cherche le min et on le met au début

Correction : invariant de boucle

Complexité

Tri en place et stable

[FG] p 310

2.2 Tri par insertion

Algorithme

Terminaison

Correction : invariant de boucle

Complexité

Trie en place et stable

[Cor] p 15

2.3 Tri bulles

Algorithme

Complexité

Tri en place et stable [Cor] p 35

3 Utilisation de structures de données : tri par tas

[Cor] p 140

Définition 3.0.1. *Tas ; fils ; père*

Opérations : entasser-max, construire + complexités

Algorithme

Complexité

Tri en place mais pas stable

4 Diviser pour régner

4.1 Tri fusion

[Cor] p 26 Algorithme

Correction (invariant de boucle)

Complexité

Tri pas en place, stable

4.2 Tri rapide

[Cor] p 157

Ø Diviser, régner, combiner

Algorithme

Analyse du tri rapide Remarque sur le choix du pivot

4.3 Optimalité

Théorème 4.3.1. *Minoration de la complexité d'un tri par comparaison [Cor] p 179*

5 Autres tris

5.1 Tri par dénombrement

[Cor] p 180
Algorithme
Complexité
Tri pas en place, stable
Rq : on améliore le minorant mais ce n'est PAS un tri par comparaisons

5.2 Tri par paquets

[Cor] p 85
Hypothèse + principe
Algorithme
Complexité
Tri pas en place, pas stable

5.3 Tri externe : tri et fusion

[FG] p 379 Utilisation quand la liste à trier ne tient pas en mémoire centrale
Création d'une monotonie
Fusion des monotonies

Développements

- Tri par tas
- Analyse du tri rapide
- Optimalité des tris par comparaison

Références

- Cormen
- Froidevaux
- Boissonat

Leçon 906 : Programmation dynamique : Exemples et applications.

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 mémoïsation. 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.

Introduction

1 Présentation du paradigme

1.1 Exemple : découpe de barre

Définition 1.1.1. *Problème de découpe de barres [Cor] p 334*

Proposition 1.1.1. *Algo naïf exponentiel (couper-barre) + explication + complexité [Cor] p 336, 338*

Rq : le problème de cet algo, c'est qu'on recalcule souvent les mêmes valeurs... On va les garder en mémoire [Cor] p 337

1.2 Mémoïsation et programmation dynamique

Définition 1.2.1. *Mémoïsation (approche descendante) [Cor] p 338*

Exemple 1.2.1. *Découpe de barre [Cor] p 338*

Définition 1.2.2. *Programmation dynamique (approche ascendante) [Cor] p 339*

Exemple 1.2.2. *Découpe de barre [Cor] p 340*

1.3 Comparaison et limites

L'approche programmation dynamique a l'avantage d'être itérative, et parfois des optimisations mémoires sont possibles (cf exemples plus loin). Mais parfois, la mémoïsation est plus efficace : elle ne calcule pas de sous-problèmes dont on aura pas besoin :

Exemple 1.3.1. *Sac à dos (sans répétition) : Entrée, sortie [Pap] p 164*
Sous-problèmes et expression en fonction de plus petits sous-problèmes [Pap] p 168
Mémoïsation plus efficace dans ce cas [Pap] p 169

Remarquer que le problème du sac à dos est NP complet : on ne pourra pas le résoudre en temps polynômial. Il est PSEUDO POLYNOMIAL ! En fait, il faut faire attention à ce qu'on entend par taille de l'entrée : ici, la taille de l'entrée $w \in \mathbb{N}$ n'est pas la valeur w mais $\log w$. Donc, l'algorithme est polynômial en w, mais exponentiel en $\log w$

Exemple 1.3.2. *C-ex : pas de programmation dynamique : plus long chemin simple non pondéré [Cor] p 354*

2 Application à l'algorithmique du texte

2.1 Distance d'édition

Définition 2.1.1. *Opérations d'éditions [Cro] p 224-225*

Définition 2.1.2. *Distance d'éditon [Cro] p 225*

Exemple 2.1.1. *Distance de Hamming : définition + retrouvée avec Lev [Cro] p 224-225*

Définition 2.1.3. *Graphe d'édition + ex en annexe ; Rq : à chaque chemin correspond un alignement, on veut un alignement optimal [Cro] p 228*

2.2 Calcul de la distance d'édition

Entrée, sortie, sous-problèmes, initialisation, récursion, solution, complexité [Cro] p 231-...
Rq : on peut calculer un alignement optimal [Cro]

2.3 Plus longue sous suite commune

Définition 2.3.1. *Pb de la plus longue sous suite commune [Cro] p 242*

Proposition 2.3.1. *Relation de la longueur avec la distance d'édition [Cro] p 242*

Rq : On peut aussi calculer la PLSSC avec un algo de programmation dynamique [Cro] p 243

3 Application à l'algorithmique des graphes

3.1 Bellman-Ford

Définition du problème, sous-problèmes et récursion + complexité [Cor] p 603

3.2 Floyd Warshall

[Cor] p 640

3.3 Voyageur de commerce

Définition 3.3.1. *Problème du voyageur de commerce [Pap] p 173*

Définition des sous-problèmes et récursion [Pap] p 174
Algo ? Complexité [Pap] p 175

4 Application aux grammaires algébriques

Algorithme de Cocke-Younger-Kasami :
Entrée : Un grammaire algébrique G et un mot w
Sortie : Oui lorsque w est engendré par G
Principe de l'algorithme + Complexité en $O(n^3|G|))$ [Car] p 199

Développements

- Bellman-Ford
- Distance de Levenstein
- ABR optimaux ?

Références

- Cormen
- Papadimitriou
- Crochemore
- Carton

Leçon 907 : Algorithmique du texte : exemples et applications

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.

1 Recherche de motif dans un texte

1.1 Algorithme naïf

Principe, algo, exemple [Cor]

1.2 Algorithme de Knuth-Morris-Pratt

Algorithme de Simon et KMP + complexités [Cor]

1.3 Arbre des suffixes

[Cro]

1.4 Boyer-Moore

[BBC]

2 Comparaison de mots

2.1 Distance de Levenstein

définitions, algo(prog dyn) [Cro]

2.2 Plus longue sous-suite commune

[Cro]

3 Compression

Lempel-Ziv-Welch [?]

Développements

- KMP
- Levenshtein

Références

- Cormen
- Crochemore
- Beauquier
- Marsault

Leçon 909 : Langages rationnels. Exemples et applications

Rapport du jury

`Des applications dans le domaine de la compilation entrent naturellement dans le cadre de ces leçons.`

Les langages rationnels sont au premier niveau de la hiérarchie de Chomsky. Ils servent notamment dans les éditeurs de texte, avec la recherche de motif.

1 Langages rationnels et langages reconnaissables

1.1 Langages rationnels

Soit A un alphabet.

Définition 1.1.1. *Langage [Car] p 13*

Définition 1.1.2. *Opérations sur les langages : union, produit et étoile [Car] p 15*

Exemple 1.1.1. *Soit $L = \{u \in A^* || u| \text{ est paire}\}$; soit $L' = \{u \in A^* || u| \text{ est impaire}\}$. Alors : $L + L' = A^*$; $LL' = L'L = L'$ [Car] p 15*

Exemple 1.1.2. *Ensemble des mots qui se terminent par un b.*

Définition 1.1.3. *Expressions rationnelles [Car] p 34*

Cette définition (si on parenthèse) inductive est non-ambigüe. [Sak] p 126

Définition 1.1.4. *Sémantique [Sak] p 126*

Définition 1.1.5. *Equivalence des expressions rationnelles [Sak] p 126*

Exemple 1.1.3. *$L = A^*aA^*$ est l'ensemble des mots contenant au moins un a. [Car] p 34*

Exemple 1.1.4. $(0 + a(1 + b1))^* = (ab + a)^*$ *[BBC] p 309*

Définition 1.1.6. *Langage rationnel [Sak] p 126*

Proposition 1.1.1. *La classe des langages rationnels est la plus petite classe .... [Car] p 33*

1.2 Langages reconnaissables

Définition 1.2.1. *Automate [Car] p 33*

Représentation graphique

Exemple 1.2.1. *[Car] p 35*

Définition 1.2.2. *Mot reconnu [Car] p 35*

Définition 1.2.3. *Langage reconnu [Car] p 35*

Exemple 1.2.2. *L'automate de l'exemple précédent reconnaît les mots ayant un nombre pair de a. [Car] p 35*

1.3 Théorème de Kleene

Proposition 1.3.1. *A toute expression rationnelle on peut associer un automate fini : construction de Thompson [Sak] p 94 ; [Car] p 37 ; [Sak] p 98 (dessin en annexe)*

Exemple 1.3.1. $(a + b)^*aba$ *[BBC] p 296*

Exemple 1.3.2. $(b + ab^*a)^*$ *[Car] p 31*

Lemme 1.3.1. *Arden [Car] p 40*

Exemple 1.3.3. $(b^*a)^*$ *[Sak] p 107*

Exemple 1.3.4. $(a + b(ab^*a)^*b)^*$ *[Sak] p 109*

Théorème 1.3.1. *Kleene [Car] p 36*

2 Propriétés des langages rationnels

2.1 Quotients d'un langage

Proposition 2.1.1. *Tout automate fini est équivalent à un automate fini déterministe complet [Car]*

Définition 2.1.1. *Quotient à gauche d'un langage [Car] p 45*

Proposition 2.1.2. *Propriétés sur les quotients [Car] p 45*

Exemple 2.1.1. $a^{-1}(b + ab^*a)^* = b^*a(ab^*a + b)^*$ *[Car] p 45*

Définition 2.1.2. *Automate minimal [Car] p 46*

Exemple 2.1.2. $(ab)^*$ *[Car] p 46*

Proposition 2.1.3. *Rationnel ssi nombre fini de quotients à gauche [Car] p 45*

Proposition 2.1.4. *L'automate minimal est celui avec le moins d'états [BBC] p 317*

L'automate minimal peut être construit à partir d'algorithmes qui s'appuient sur :

Définition 2.1.3. *Congruence de Nérode [Car] p 47*

Proposition 2.1.5. *Unicité de l'automate minimal à isomorphisme près [Car] p 48*

Exemple 2.1.3. *[Sak] p 125*

2.2 Lemme de l'étoile

Théorème 2.2.1. *Lemme de l'étoile (3 formulations) [Sak] p 78*

Ce théorème sert à montrer que des langages ne sont PAS rationnels [Sak] p 79

Exemple 2.2.1. $L = \{a^n b^n | n \in \mathbb{N}\}$ *n'est pas rationnel. [Sak] p 79*

Exemple 2.2.2. $L = \{a^n b^m | n, m \in \mathbb{N}\}$ *satisfait* 1 *mais pas* 2 *donc n'est pas rationnel [Sak] p 80*

La réciproque est fausse!!!!!!!

Exemple 2.2.3. $L = \left\{a^{k_1} b a^{k_2} ... a^{k_n} b | \exists i, i \neq k_i\right\}$ *satisfait tous les lemmes de l'étoile mais n'est pas rationnel [Sak] p 80*

Théorème 2.2.2. *Une réciproque : il faut que le langage et son complémentaire vérifient la 3è version (par blocs)[Sak] p 127*

3 Applications

Les langages rationnels ont de nombreuses applications, notamment liées à l'analyse lexicale des langages de programmation et à la recherche de motif. [Sak]

3.1 Recherche de motif

[BBC] On considère un texte $t = t_1...t_n$; on cherche une occurrence du motif $x = x_1...x_n$ dans t.

3.1.1 Algorithme naïf

On compare le motif à chaque facteur de t.
Complexité : en nk.

3.1.2 Algorithme de Knuth-Morris et Pratt et automate de Simon

KMP et SIMON [Cor] On optimise le décalage dans la comparaison des motifs en considérant le plus grand suffixe lu qui est aussi préfixe du motif. On peut aussi construire l'automate des occurrences et l'utiliser dans la recherche du motif. Complexité en $n + k$

3.2 Décidabilité de l'arithmétique de Presburger

Théorème 3.2.1. *Décidabilité Presburger* *[Car]*

Développements

- Knuth Morris Pratt
- automate minimal
- Presburger

Références

- Carton
- Sakarovitch
- Beauquier
-

Leçon 910 : Langages algébriques. Exemples et applications.

Rapport du jury

1 Représentations des langages algébriques

1.1 Grammaires algébriques

1.1.1 Définitions

Définition 1.1.1. *Grammaire algèbre [Car] p 83*

Rq : conventions : Majuscules et minuscules

Exemple 1.1.1. *Ex 2.2 [Car] p 84*

Définition 1.1.2. *Dérivation [Car] p 84*

Exemple 1.1.2. *2.4 [Car] p 84*

Définition 1.1.3. *Langage engendré [Car] p 85*

Définition 1.1.4. *Langage algébrique [Car] p 85*

Exemple 1.1.3. *Langage de Dyck [Car] p 85*

Lemme 1.1.1. *Fondamental [Car] p 87*

1.1.2 Simplification des grammaires

Définition 1.1.5. *Grammaire réduite [Car] p 88*

Proposition 1.1.1. *réduction [Car] p 88*

Définition 1.1.6. *Grammaire propre [Car] p 89*

Proposition 1.1.2. ∃ *grammaire propre équivalente [Car] p 89*

Rq : Grammaire propre : pas ϵ

Exemple 1.1.4. *2.19 [Car] p 89*

Proposition 1.1.3. *Forme normale quadratique/Chomsky [Car] p 90*

Proposition 1.1.4. ∃ *FNC équivalente [Car] p 90*

Greibach ? ? ?

1.2 Arbres de dérivation

Définition 1.2.1. *Arbre de dérivation [Car] p 99*

Exemple 1.2.1. *2.41 [Car] p 100*

Proposition 1.2.1. *langage engendré et dérivation [Car] p 99*

Définition 1.2.2. *Grammaire ambigüe [Car] p 99*

Exemple 1.2.2. *2.41 [Car] p 100*

Langage ambigü ? Ou dans sous classes ?

1.3 Automates à piles

Définition 1.3.1. *Automate à pile [Car] p 115*

Définition 1.3.2. *Calcul [Car] p 115*

Définition 1.3.3. *Mot accepté par pile vide, par état final [Car] p 116*

Exemple 1.3.1. *2.66 [Car] p 116*

Proposition 1.3.1. *Equivalence des modes d'acceptation [Car] p 117*

Proposition 1.3.2. *Equivalence grammaire/automate à pile [Car] p 119*

Exemple 1.3.2. *2.72 et 2.73 [Car] p 121*

Définition 1.3.4. *Automate à pile déterministe [Car] p 123*

Exemple 1.3.3. *2.76 [Car] p 124*

Rq : Si l'automate est déterministe, les modes d'acceptation ne sont plus équivalents [Car] p 124

2 Propriétés des langages algébriques

2.1 Lemme d'itération

Lemme 2.1.1. *Ogden [Car] 100*

Corollaire 2.1.1. *Théorème de Bar-Hillel, Perles et Shamir [Car] p 103*

Proposition 2.1.1. $a^n b^n c^n$ *pas algébrique [Car] p 103*

Corollaire 2.1.2. *LA pas clos par intersection ou complémentation [Car] p 103*

2.2 Langages ambigus

Définition 2.2.1. *Langage ambigu, inhéremment ambigu [Car] p 100*

Exemple 2.2.1. *2.41 [Car] p 100*

Rq : Le lemme d'Ogden sert à montrer qu'un langage n'est PAS algébrique, MAIS AUSSI qu'il est inhéremment ambigu

Proposition 2.2.1. *Langages non ambigus inclus strictt 2.48[Car] p 104*

2.3 Langages rationnels

Définition 2.3.1. *Opérations rationnelles [Car] p 106*

Corollaire 2.3.1. *Rationnels aussi algébriques [Car] p 106*

Proposition 2.3.1. *Intersection avec un rationnel [Car] p 107*

2.4 Langages algébriques déterministes

Définition 2.4.1. *Langage déterministe [Car] p 124*

Proposition 2.4.1. *Complémentaire d'un déterministe l'est aussi [Car] p 124*

Proposition 2.4.2. *Déterministe ⇒ non ambigu [Car] p 126*

Proposition 2.4.3. *déterministe inter rationnel déterministe [Aut] p 127*

Proposition 2.4.4. *Inclusion stricte [Aut] p 128*

3 Problèmes de décision

3.1 Problèmes indécidables

Proposition 3.1.1. *Problèmes indécidables* *[Car] p 167*

3.2 Problèmes décidables

Proposition 3.2.1. *On peut décider si un mot appartient à une grammaire : algo CYK [Car] p 199*

Proposition 3.2.2. *4 Problèmes décidables [Aut] p 80-84*

4 Application des langages algébriques : analyse syntaxique

cf. Leçon analyse syntaxique

Théorème 4.0.1. *CYK*

Développements

- Problèmes indécidables sur les grammaires
- CYK

Références

- Carton
- Autebert
- Hopcroft Ullman

Leçon 912 : Fonctions récursives primitives et non-primitives. Exemples.

Rapport du jury

RIEN

On considère des fonctions définies sur un sous-ensemble de $\mathbb{N}^p$ à valeurs dans $\mathbb{N}$.

1 Fonctions récursives primitives

1.1 Construction

Définition 1.1.1. *Fonctions de base [LdR] p 138*

Définition 1.1.2. *Fonction composée [LdR] p 138*

Définition 1.1.3. *Schéma de récurrence [LdR] p 138*

Exemple 1.1.1. *Somme ; prédécesseur ; produit (formule par récu) [Car] p 182*

Définition 1.1.4. *Fonction récursive primitive [LdR] p 139*

Exemple 1.1.2. *Somme, prédécesseur, produit, égalité, division, reste, puissance, racine, logarithme sont toutes primitives récursives [Car] p 182-183*

Exemple 1.1.3. *Différence et signe [Wol] p 124-125*

1.2 Prédicats récursifs primitifs

Définition 1.2.1. *prédicat [Wol] p 125*

Exemple 1.2.1. $n < m$ *[Wol] p 126*

Définition 1.2.2. *Prédicat récursif primitif [Wol] p 126*

Exemple 1.2.2. *prédicat zéro ; prédicat plus petit [Wol] p 126-127*

1.3 Autres méthodes de construction

1.3.1 Définition par cas (IF THEN ELSE)

Proposition 1.3.1. *Autre méthode de construction : par cas [LdR] p 139*
Énoncer avec prédicat

1.3.2 Somme et produit bornés

Proposition 1.3.2. *Somme bornée et produit borné sont primitifs récursifs [LdR] p 140*

1.3.3 Quantification bornée

Proposition 1.3.3. *Quantification bornée [LdR] p 140*

1.3.4 Minimisation bornée

Définition 1.3.1. *minimisation bornée [LdR] p 141*

Exemple 1.3.1. $\pi : n \mapsto n+1$*-è nombre premier [CLb] p 15*

2 Les limites du modèle

2.1 Cardinalité

Proposition 2.1.1. *Les fonctions récursives primitives sont en nombre dénombrable [Wol] p 129*

Rq : il y a un nombres indénombrable de fonctions calculables

Théorème 2.1.1. *Il existe des fonctions calculables non récursives primitives [Wol] p 129*

Exemple 2.1.1. *Construction par argument diagonal [Wol] p 130*

2.2 La fonction d'Ackermann

Définition 2.2.1. *Fonction d'Ackermann [Deh] p 189*

Proposition 2.2.1. *Ackermann n'est pas récursive primitive* *[Deh] p 191*

Rq : on verra plus loin qu'elle est récursive

3 Fonctions récursives

Définition 3.0.2. *Minimisation non-bornée [Wol] p 131*

3.1 Fonction récursives totales

Définition 3.1.1. *Prédicat sûr [Wol] p 131*

Définition 3.1.2. *fonction récursive (totale) [Wol] p 131*

Proposition 3.1.1. *Les fonctions récursives primitives sont récursives*

3.2 Lien avec les machines de Turing

Définition 3.2.1. *Fonction calculable pas MT [Car] p 160*

Théorème 3.2.1. *récursive ⇒ calculable [Wol] p 134*

Théorème 3.2.2. *calculable ⇒ récursive* *[Wol] p 135 [Car]*

Exemple 3.2.1. *Ackermann est récursive [Deh]*

Exemple 3.2.2. *C-ex : le Castor affairé n'est pas calculable [Deh]*
Donc, il existe des fonctions totales qui ne sont pas récursives

3.3 Fonctions récursives partielles, ensembles RE

Exemple 3.3.1. *fonction partielle [Wol] p 136*

Définition 3.3.1. *Fonction récursive partielle [Wol] p 137*

Définition 3.3.2. *fonction partielle calculable par MT [semi-calc] [Wol] p 136*

Théorème 3.3.1. *partielle : récursive ssi semi-calculable [Wol] p 137*

RE ?

Développements

- calculable ssi récursif
- Ackerman

Références

- Carton
- Lassaigne
- CL2
- Wolper
- Dehornoy

Leçon 913 : Machine de Turing. Applications.

Rapport du jury

RIEN

1 Définitions

1.1 Machine de Turing

Définition 1.1.1. *Machine de Turing [Car] p 141*

Définition 1.1.2. *Machine de Turing déterministe [Car] p 141*

Exemple 1.1.1. *En annexe [Car] p 142*

Définition 1.1.3. *Principe rapide de fonctionnement + configuration [Car] p 142*

Exemple 1.1.2. *Configuration [Car] p 143 (en annexe)*

Définition 1.1.4. *Etape de calcul [Car] p 143*

Définition 1.1.5. *Calcul [Car] p 143*

Rq : calculs infinis ?

1.2 Mode accepteur

Définition 1.2.1. *Langage accepté [Car] p 145*

Exemple 1.2.1. *Langage accepté dans l'ex de MT précédent [Car] p 141*

1.3 Variantes

1.3.1 Normalisation

Proposition 1.3.1. *Normalisation [Car] p 146*

1.3.2 Bandes bi-infines

Définition 1.3.1. *Bande bi-infinie [Car] p 149*

Proposition 1.3.2. *Equivalence [Car] p 149*

1.3.3 Machines à plusieurs bandes

Définition 1.3.2. *Machine à plusieurs bandes [Car] p 150*

Proposition 1.3.3. *Equivalence [Car] p 151*

1.3.4 Machines déterministes

Définition 1.3.3. *Machine déterministe [Car] p 152*

Proposition 1.3.4. *Equivalence [Car] p 152*

1.4 Mode calculateur

Définition 1.4.1. *Fonction calculable [Car] p 160*

Théorème 1.4.1. *fonction calculable ssi récursive (⇒)* *[Car] p 185*

Exemple 1.4.1. *C-ex : Castor affairé [Deh] p 186-187*

2 Décidabilité

2.1 Langages récursivement énumérables

Définition 2.1.1. *RE [Car] p 155*

Définition 2.1.2. *Enumérateur [Car] p 155*

Proposition 2.1.1. *RE ssi énuméré [Car] p 155*

Exemple 2.1.1. *L_E est RE [Car] p 159*

Proposition 2.1.2. *Cardinalité : il existe des langages pas récursivement énumérables [Car] p 156*

Exemple 2.1.2. *Langage diagonal [Car] p 156*

Proposition 2.1.3. *Stabilité par union et intersection [Car] p 157*

2.2 Langages récursifs

Définition 2.2.1. *Langage récursif [Car] p 158*

Exemple 2.2.1. *L_E n'est pas récursif*

Proposition 2.2.1. *Stable par union intersection complémentaire [Car] p 158*

Proposition 2.2.2. *Si L est L^c sont RE, L est R [Car] p 159*

Corollaire 2.2.1. *L_E^c n'est pas RE [Car] p 160*

2.3 Réduction

Définition 2.3.1. *Réduction [Car] p 160*

Proposition 2.3.1. *Réduction - décidables [Car] p 161*

Corollaire 2.3.1. *Réduction - indécidable [Car] p 161*

Application 2.3.1. *langage non vide et non égalité de langages indécidables [Car] p 161*

2.4 Exemples

2.4.1 Arrêt et Rice

Proposition 2.4.1. *Le problème de l'arrêt est indécidable [Wol] p 146*

Théorème 2.4.1. *Rice [Car] p 162*

2.4.2 Machines linéairement bornées

Définition 2.4.1. *MT linéairement bornée [Car] p 171*

Théorème 2.4.2. *langage vide ou mot $\in$ indécidables [Car] p 174*

2.4.3 POST

Définition 2.4.2. *PCP [Car] p 163*

Théorème 2.4.3. *PCP est indécidable [Car] p 165*

3 Complexité

3.1 Complexité temporelle

Définition 3.1.1. *Complexité en temps [Car] p 194*

Définition 3.1.2. *Classes de complexité temporelle [Car] p 196-197*

Théorème 3.1.1. *Inclusions [Car] p 197*

Exemple 3.1.1. *SAT est NP [Car] p 203*

Définition 3.1.3. *réduction polynômiale [Car] p 202*

Proposition 3.1.1. *réduction avec P [Car] p 202*

3.2 NP-complétude

Définition 3.2.1. *NP dur et NP complet [Car] p 202*

Proposition 3.2.1. *réduction NP dur [Car] p 203*

Théorème 3.2.1. *Cook [Car] p 203*

Application 3.2.1. *chemin hamiltonien est NP complet [Car] p 206*

3.3 Complexité spatiale

Définition 3.3.1. *Complexité spatiale [Car] p 194*

Théorème 3.3.1. *Savitch (admis ?) [Car] p 219*

Définition 3.3.2. *Classes de complexité en espace [Car] p 221*

Proposition 3.3.1. *Inclusions [Car] p 221*

Annexe : inclusions [Car] p 222

Développements

- calculable ssi récursif
- Cook
- Rice

Références

- Carton
- Dehornoy
- Sipser

Leçon 914 : Décidabilité et indécidabilité. Exemples.

Rapport du jury

RIEN

On considère des problèmes qui sont des questions auxquelles on doit répondre par oui par non. On essaie de trouver des algorithmes pour répondre à ces questions (automatisation) : s'il y a un algo, c'est décidable ! Parfois, c'est pas décidable et c'est inutile de chercher un algo.

1 Définitions et outils

1.1 Problèmes et codages

Définition 1.1.1. *Problème de décision [Car] p 139*

Exemple 1.1.1. *Nombres premiers [Car] p 139*

Définition 1.1.2. *Langage associé à un problème ; codage [Car] p 139*

Exemple 1.1.2. *Codage des entiers ; des graphes [Car] p 139*

1.2 Rappels sur les machines de Turing

Définition 1.2.1. *Langage accepté ; langage décidé par MT [Car] p 145,158*

Définition 1.2.2. *Langage récursif ; langage récursivement énumérable ; notation R et RE [Car] p 155,158*

Exemple 1.2.1. *Langage d'acceptation est RE mais pas R [Car] p 159*

Proposition 1.2.1. $R \subset RE$

Proposition 1.2.2. *Stabilité des R [Car] p 158*

Proposition 1.2.3. *Complémentation et décidabilité [Car] p 159*

Corollaire 1.2.1. $\overline{L_\epsilon}$ *n'est pas RE [Car] p 159*

1.3 Décidabilité, indécidabilité des problèmes

Définition 1.3.1. *Problème décidable ; semi décidable [?] p 92*

A l'oral : insister sur le fait que la décidabilité ne dépend pas du codage ! ! ! !

Proposition 1.3.1. *Décidable* $\Rightarrow$ *semi décidable*

Définition 1.3.2. *Fonction calculable [Car] p 160*

Définition 1.3.3. *Réduction [Car] p 160*

Proposition 1.3.2. *Réduction et décidabilité [Car] p 161*

2 Problèmes indécidables sur les machines de Turing

2.1 Problème de l'acceptation

Définition 2.1.1. *Problème de l'acceptation [Car] p 159 (problème associé à* L_ϵ*)*

Théorème 2.1.1. *Il est semi décidable [Car] p 159*

2.2 Problème de l'arrêt

Définition 2.2.1. *Problème de l'arrêt [Wol] p 146*

Proposition 2.2.1. *indécidable (mais RE) [Wol] p 146*

Lemme 2.2.1. *Déf : arrêt sur mot vide ; indécidable [Wol] p 147*

Définition 2.2.2. *Arrêt existentiel [Wol] p 147*

Proposition 2.2.2. *Indécidable [Wol] p 147*

Définition 2.2.3. *Arrêt universel [Wol] p 148*

Proposition 2.2.3. *Indécidable [Wol] p 148*

2.3 Théorème de Rice

Définition 2.3.1. *Propriété non triviale sur RE [Car] p 161*

Théorème 2.3.1. *Rice (rédiger sous la forme d'un problème indécidable) [Car] p 162*

On ne peut pas construire de procédure effective qui permet de prouver qu'un algo est correct ! !

3 Problèmes en logique

3.1 Théories et problèmes associés

Définition 3.1.1. *Théorie [DNR] p 79*

Définition 3.1.2. *Théorie consistante (admet un modèle), ie non contradictoire [DNR] p 80*

Définition 3.1.3. *Théorie complète [DNR] p 80*

Définition 3.1.4. *Théorie récursive ; théorie décidable [DNR] p 125*

Attention ! Ce sont des problèmes bien DIFFERENTS ! ! ! ! ! !

Théorème 3.1.1. *Une théorie complète et récursive sur L dénombrable* $\Rightarrow$ *décidable [DNR] p 126*

3.2 Arithmétique de Péano

Définition 3.2.1. *Arithmétique de Péano [DNR] p 111*

Théorème 3.2.1. *T consistant contenant PA alors indécidable (admis) [DNR] p 126*

3.3 Arithmétique de Presburger

Définition 3.3.1. *Presburger [DNR] p 136*

Théorème 3.3.1. *décidable* *[DNR] p 136 [Car]*

4 Problèmes sur les langages algébriques

4.1 Problème de correspondance de Post

Définition 4.1.1. *PCP [Car] p 163*

Théorème 4.1.1. *PCP est indécidable [Car] p 165*

4.2 Application à des problèmes indécidables

Proposition 4.2.1. *Problèmes indécidables sur les grammaires* *[Car] p 167*

4.3 Problèmes décidables

Proposition 4.3.1. *4 Problèmes décidables sur les grammaires [Aut] p 80-84*

5 Problèmes associés à la réécriture

5.1 Définitions

Définition 5.1.1. *Règle et système de réécriture [Baa] p 61*

Définition 5.1.2. *Système confluent [Baa] p 9*

Définition 5.1.3. *Système localement confluent [Baa] p 28*

Définition 5.1.4. *Terminant et convergeant [Baa] p 9*

Théorème 5.1.1. *Lemme de Newman [Baa] p 28*

5.2 Terminaison

Théorème 5.2.1. *Indécidabilité de la terminaison* *[Baa] p 99*

Théorème 5.2.2. *Décidabilité dans le cas d'un système clos à droite [Baa] p 99,100*

5.3 Confluence

Définition 5.3.1. *Problème de la confluence [Baa] p 134*

(indécidable)

Définition 5.3.2. *Paire critique [Baa] p 139*

Exemple 5.3.1. *[Baa] p 139*

Lemme 5.3.1. *Des paires critiques [Baa] p 140*

Théorème 5.3.1. *Paires critiques [Baa] p 140*

(décidable dans ce cas)

Développements

- Problèmes indécidables sur les grammaires
- Indécidabilité de la terminaison d'un système de réécriture
- Décidabilité de l'arithmétique de Presburger

Références

- Carton
- Sipser
- Wolper
- DNR
- Autebert, décidabilité
- Baader

Leçon 915 : Classes de complexité. Exemples.

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.

Rq : on ne considère que des problèmes décidables, et des machines qui s'arrêtent tout le temps [Car] p 193

1 Machine de Turing et complexité

1.1 Définition de la notion de complexité

Définition 1.1.1. *Complexité en temps et en espace [Car] p 194*

Proposition 1.1.1. *Inégalités fondamentales [Car] p 194*

1.2 Changement de modèle

1.2.1 Machine à bande bi-infinie

Proposition 1.2.1. *Complexité en temps et en espace inchangée [Car] p 195,218*

1.2.2 Machine à k bandes

Proposition 1.2.2. *Complexité en temps quadratique, inchangée en espace [Car] p 195,218*

1.2.3 Machine non-déterministe

Proposition 1.2.3. *Complexité en temps changée [Car] p 195*

Théorème 1.2.1. *Savitch [Car] p 219*

2 Complexité en temps

2.1 Classes de complexité temporelles

Définition 2.1.1. *Classes de complexité temporelles [Car] p 196*

Rq : machines à plusieurs bandes mais c'est seulement quadratique quand on passe à une bande (DANS LE CAS DÉTERMINISTE)

Définition 2.1.2. *classes duales, autoduales [Car] p 197*

Proposition 2.1.1. *Cas des machines déterministes = autodual [Car] p 197*

Exemple 2.1.1. *Path, langages algébriques sont dans P [Car] p 198*

Définition 2.1.3. *Vérificateur [Car] p 200*

Proposition 2.1.2. *Équivalence vérificateur et NP [Car] p 200*

Exemple 2.1.2. *Sat, Ham sont dans NP [Car] p 199*

Rq : idem pour NEXP

2.2 Inclusions

Proposition 2.2.1. *inclusions [Car] p 197*

Dessin des inclusions [Car] p 197

Définition 2.2.1. *Fonction constructible en temps [Car] p 233*

Théorème 2.2.1. *Hiérarchie (temps) [Car] p 233*

Corollaire 2.2.1. *P strict inclus dans EXPTIME [Car] p 234*

Rq : P=NP ouvert [Car] [Sip]

2.3 Réduction polynômiale

Définition 2.3.1. *Réduction polynômiale [Car] p 202*

Application 2.3.1. *réduc et pb de P [Car] p 202*

2.4 Complétude

Définition 2.4.1. *NP-difficile et NP-complet [Car] p 202*

Théorème 2.4.1. *Cook [Car] p 203 [?]*

Proposition 2.4.1. *Réduction polynômiale et NP-complétude [Car] p 203*

Exemple 2.4.1. *Chemin hamiltonien*, *clique [Car]*

3 Complexité en espace

3.1 Classes de complexité en espace

Définition 3.1.1. *Classes de complexité en espace [Car] p 221*

Rq : d'après Savitch, pas besoin de distinguer les machines déterministes et non-déterministes [Car] p 221

Exemple 3.1.1. *Universalité d'un automate fini [Car] p 222,223*

Définition 3.1.2. *QSAT [Car] p 223*

Proposition 3.1.1. *QSAT est dans PSPACE [Car] p 223*

3.2 Inclusions

Proposition 3.2.1. *Inclusions + dessin en annexe [Car] p 221*

Définition 3.2.1. *Fonction constructible en espace [Car] p 233*

Théorème 3.2.1. *Hiérarchie (espace) [Car] p 233*

Corollaire 3.2.1. *PSCPACE strict inclus dans EXPSPACE [Car] p 234*

3.3 PSPACE-complétude

Définition 3.3.1. *PSPACE-complétude [Car] p 223*

Théorème 3.3.1. *QSAT est PSPACE-complet [Car] p 224*

Lemme 3.3.1. *Réduction et complétude [Car] p 225*

Développements

- Cook
- HAM est NP-complet

Références

- Carton
- Sipser
- Perifel

Leçon 916 : Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications.

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.

1 Syntaxe et sémantique

1.1 Syntaxe

Ensemble de symboles : $\neg, lor, land, \Rightarrow, \Leftrightarrow$;
Ensemble (dénombrable) de variables propositionnelles

Définition 1.1.1. *Ensemble des formules propositionnelles par induction [CLa] p 18*

Proposition 1.1.1. $\mathcal{F} = \cup \mathcal{F}_n$ *[CLa] p 19*

Définition 1.1.2. *Hauteur d'une formule [CLa] p 20*

Exemple 1.1.1. *Représentation sous forme d'arbre [CLa] p 23*

Lemmes ?

Théorème 1.1.1. *Lecture unique [CLa] p 27*

Rq : i.e. définition inductive non ambigüe

Définition 1.1.3. *Substitution : $F[G/p]$ par induction [CLa] p 30*

Exemple 1.1.2. *[CLa] p 30*

1.2 Sémantique

Définition 1.2.1. *Valuation [CLa] p 32*

Théorème 1.2.1. *Extension : valeur de vérité [CLa] p 35*

Proposition 1.2.1. *La valeur de vérité ne dépend que des valuations des vp de la formule [CLa] p 37*

Proposition 1.2.2. *Valeur de vérité si substitution [CLa] p 40*

Corollaire 1.2.1. *Tautologies, formules équivalentes [CLa] p 40,41*

Exemple 1.2.1. *Représentation : table de vérité [CLa] p 36 + annexe [CLa] p 37,38*

1.3 Tautologies, formules équivalentes et satisfiabilité

Définition 1.3.1. *Formule satisfaite [CLa] p 36*

Définition 1.3.2. *Tautologie [CLa] p 38*

Définition 1.3.3. *Formules équivalentes [CLa] p 39*

Exemple 1.3.1. *Tautologie et formules équivalentes [CLa] p 42,44*

Définition 1.3.4. *Ensemble de formules satisfiable , finiment satisfiable[CLa] p 59*

Définition 1.3.5. *Conséquence [CLa] p 59*

Proposition 1.3.1. *F csquce de Γ ssi $\Gamma \cup \{\neg F\}$ satisfiable [CLa] p 60*

Théorème 1.3.1. *Compacité [CLa] p 62*

App : k-coloriage ? ? ?

2 Formes équivalentes

2.1 Formes normales conjonctive et disjonctive

Définition 2.1.1. *FND, FNC [CLa] p 50*

Théorème 2.1.1. *Il y a une FND, FNC équivalente à toute formule [CLa] p 51*

Exemple 2.1.1. *[CLa] p 52*

2.2 Systèmes complets de connecteurs

Définition 2.2.1. *Système complet, système complet minimal [CLa] p 54*

Proposition 2.2.1. $\neg, \wedge, \vee$; $\neg, \vee$ *[CLa] p 54*

2.3 Représentation par les ROBDD

Définition 2.3.1. *Arbre binaire de décision [HR] p 320*

Définition 2.3.2. *Diagramme binaire de décision [HR] p 323*

Exemple 2.3.1. *[HR] p 319*

2.4 Le problème SAT

Définition 2.4.1. *SAT [Car] p 199*

Théorème 2.4.1. *Cook* *[Car] p 203*

Application 2.4.1. *Le problème du chemin hamiltonien est NP-complet [Car] p 207*

Proposition 2.4.1. *Algo qui résout SAT : DPLL [DaML] p 57*

Application 2.4.2. *Pour résoudre un problème, on l'encode par des formules et on applique un algo pour SAT*

3 Systèmes de déduction

3.1 Déduction par coupure

Les hypothèses sont des clauses (FNC) On met les formules sous forme normale conjonctive : c'est un produit de clauses. On ne garde que les clauses.

Définition 3.1.1. *Règle de coupure [LdR] p 31*

Définition 3.1.2. *Preuve par coupure, réfutation par coupure, notation $\vdash$ [LdR] p 32*

Proposition 3.1.1. *Correction [LdR] p 32-33*

Proposition 3.1.2. *Complétude [LdR] p 36*

Exemple 3.1.1. *[LdR] p 39*

3.2 Déduction naturelle

Définition 3.2.1. *Règles de déduction, logique classique LK [LdR] p 41*

Définition 3.2.2. *Logique intuitionniste [LdR] p 43*

Exemple 3.2.1. *[LdR] p 43*

Développements

- Théorème de compacité
- Théorème de Cook
- DPLL

Références

- Cori Lascar I
- Huth Ryan
- Carton
- Devismes Lafourcade
- Lassaigne Rougemont

Leçon 917 : Logique du premier ordre : Syntaxe et sémantique.

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.

1 Syntaxe

1.1 Les formules

Définition 1.1.1. *Langage [DNR] p 11*

Exemple 1.1.1. *Langage sur la théorie des groupes [DNR] p 12*

On note $\mathcal{V}$ un ensemble infini de variables

Définition 1.1.2. *Terme (formelle) [DNR] p 13*

Définition 1.1.3. *Terme clos [DNR] p 13*

Définition 1.1.4. *Hauteur [DNR] p 13*

Exemple 1.1.2. *Terme (th des groupes) + représentation sous forme d'arbre [DNR] p 14*

Définition 1.1.5. *Formule atomique [CLa] p 149*

Définition 1.1.6. *Formule [CLa] p 151*

Rq : on définit la hauteur comme dans le cas d'un terme.

Exemple 1.1.3. *Formule th des groupes + représentation sous forme d'arbre[DNR] p 16*

Théorème 1.1.1. *Lecture unique [CLa] p 151*

1.2 Variables libres, variables liées et substitutions

Définition 1.2.1. *Variables libres, liées(induction) [DNR] p 19*

Exemple 1.2.1. *[DNR]*

Définition 1.2.2. *Formules α-équivalentes [DNR] p 20*

Définition 1.2.3. *Formule close [DNR] p 21*

Définition 1.2.4. *Cloture [DNR] p 21*

Exemple 1.2.2. *th des groupes [DNR] p 16*

1.3 Unification

Définition 1.3.1. *Substitution [DNR] p 22*

Définition 1.3.2. *Termes unifiables [DNR] p 248*

Théorème 1.3.1. *Unificateur principal [DNR] p 248*

Théorème 1.3.2. *algorithme d'unification ; terminaison, correction [DNR] p 249*

1.4 Théories

Définition 1.4.1. *Théorie [DNR] p 79*

Exemple 1.4.1. *Théorie de l'égalite [DNR] p 104*

Exemple 1.4.2. *Arithmétique de Presburger [DNR] p 136*

Exemple 1.4.3. *Arithmétique de Peano [DNR] p 111*

2 Sémantique

2.1 Interprétation

Définition 2.1.1. *structure sur un langage [DNR] p 68*

Définition 2.1.2. *Valuation (environnement) [DNR] p 69*

Définition 2.1.3. *Interprétation d'un terme, d'une formule [DNR] p 69-70*

Exemple 2.1.1. *th des groupes [DNR] p 69*

Définition 2.1.4. *Modèle d'une formule + notation [DNR] p 71*

Lemme 2.1.1. *L'interprétation ne dépend que des variables libres [DNR] p 71*

Proposition 2.1.1. *si F close [DNR] p 71*

Définition 2.1.5. *Formule valide [CLa] p 177*

2.2 Modèle d'une théorie

Définition 2.2.1. *Modèle d'une théorie [CLa] p 178*

Définition 2.2.2. *Théorie consistante, contradictoire, conséquence ; équivalence [CLa] p 178*

Exemple 2.2.1. *PA consistante ($\mathbb{N}$) [DNR] p 112*

Proposition 2.2.1. *$T \models^* G$ ssi $T\{\neg G\}$ contradictoire [CLa] p 181*

Théorème 2.2.1. *Lowenheim Skolem [DNR]*

3 Déduction

4 Déduction : méthode de résolution

4.1 Mise sous forme de clauses

Définition 4.1.1. *Littéral [DNR] p 264*

Définition 4.1.2. *Clause [DNR] p 264*

Mise d'un ensemble de formules sous forme de clause [LdR] p 84 Forme de Skolem : thm sur l'existence de modèles !

4.2 Méthode de résolution/correction

Définition 4.2.1. *Termes unifiables, unificateur [DNR] p 248*

Définition 4.2.2. *Filtre [DNR] p 248*

Théorème 4.2.1. *Unificateur le plus général [DNR] p 249*

But : on prend un ensemble de clauses E et on veut dériver la clause vide.

Définition 4.2.3. *Méthode de résolution [DNR] p 265*

Rq : on définit les unificateurs sur les littéraux comme sur les termes

Exemple 4.2.1. *(preuve en annexe) [DNR] p 265*

Définition 4.2.4. *INCOHERENT lorsqu'on peut dériver la clause vide [DNR] p 267*

Proposition 4.2.1. *Correction : incohérent $\Rightarrow$ contradictoire*

4.3 Modèles de Herbrand et complétude

Définition 4.3.1. *Domaine et structure de Herbrand [LdR] p 98*

Définition 4.3.2. *Modèle de Herbrand [LdR] p 99*

Proposition 4.3.1. *modèle ssi modèle de Herbrand [LdR] p 99*

Lemme 4.3.1. *dérivation d'un littéral qui filtre [DNR] p 266 (utilisé mais pas montré dans le dvpt)*

Théorème 4.3.1. *Complétude de la méthode de résolution [DNR] p 267*

Développements

- Unification
- Lovenheim Skolem
- Résolution

Références

- DNR
- Cori Lascar ?

Leçon 918 : Systèmes formels de preuve en logique du premier ordre : exemples.

Rapport du jury

RIEN

Définition 0.3.3. *Système formel et système de déduction cf cours Cachera*

1 Déduction naturelle

1.1 Séquents et règles de démonstration

Définition 1.1.1. *Séquent [DNR] p 24*

Définition 1.1.2. *Séquent prouvable, toutes les règles [DNR] p 24-29*

Notations $\vdash, \nvdash$

Définition 1.1.3. *Formule prouvable [DNR] p 24*

Exemple 1.1.1. *Un séquent prouvable [DNR] p 31 + arbre en annexe ? ?*

1.2 Validité et complétude

Théorème 1.2.1. *Validité [LdR] p 74*

Définition 1.2.1. *Théorie complète (ou PAS exclusif), théorie de Henkin [LdR] p 74*

Théorème 1.2.2. *Henkin et complète ET cohérente $\Rightarrow$ admet un modèle [LdR] p 75*

Définition 1.2.2. *Extension [LdR] p 75*

Théorème 1.2.3. *extension qui est de Henkin [LdR] p 75*

Corollaire 1.2.1. *cohérente (ie...) $\Rightarrow$ admet un modèle [LdR] p 76*
i.e. si F conséquence alors F prouvable [LdR] p 76

2 Calcul des séquents

Rq : [DNR] p 185

2.1 Description du système (LK)

Définition 2.1.1. *Séquent (!!!! on change la définition d'avant) [DNR] p 186*

Définition 2.1.2. *prouvable dans LK + règles [DNR] p 187*

Exemple 2.1.1. *[DNR] p 189*

2.2 Équivalence avec la déduction naturelle

Proposition 2.2.1. *preuve dans NK $\Rightarrow$ dans LK [DNR] p 195*

Proposition 2.2.2. *(thm + cor) LK $\Rightarrow$ NK [DNR] p 200*

Rq : on a donc validé et complétude de ce système de déduction.

2.3 Élimination des coupures

Définition 2.3.1. *Dérivation normale [DNR] p 200*

Définition 2.3.2. *Règle mix [DNR] p 201*

Lemme 2.3.1. *Lemmes sur mix et coupures [DNR] p 201,202*

Théorème 2.3.1. *élimination des coupures [DNR] p 200*

3 Résolution

Rq : on cherche à montrer qu'un ensemble de clauses est contradictoire.

3.1 Mise sous forme de clauses

Définition 3.1.1. *Clause [LdR] p 84*

Méthode : mise sous forme de clauses (étapes) [LdR] p 84

Exemple 3.1.1. *[LdR] p 84*

Proposition 3.1.1. *Equivalence [LdR] p 84*

3.2 Unification

Définition 3.2.1. *Termes unnifiables, unificateur principal [DNR] p 248*

Algo d'unification [DNR] p 249

3.3 Méthode de résolution

Définition 3.3.1. *règles res et contr [DNR] p 265*

Exemple 3.3.1. *[DNR] p 265*

Définition 3.3.2. *Ensemble de clauses vrai, contradictoire, [inconsistant] inCohérent [DNR] p 267*

Proposition 3.3.1. *Correction de la résolution [DNR] p 267*

3.4 Modèles de Herbrand, correction et complétude

Définition 3.4.1. *Modèle et structure de Herbrand [LdR] p 98*

Lemme 3.4.1. *modele ssi modele de Herbrand [LdR] p 99*

Définition 3.4.2. *instance [LdR] p 99*

Proposition 3.4.1. *pas de modèle de Herbrand alors ... [LdR] p 99*

Lemme 3.4.2. *Filtres [DNR] p 266*

Proposition 3.4.2. *Complétude de la résolution [DNR] p 267*

Développements

- Complétude de la déduction naturelle
- Complétude de la méthode de résolution

Références

- DNR
- Lassaigne-Rougemont

Leçon 919 : Unification : algorithmes et applications.

Rapport du jury

RIEN

1 Unification

1.1 Termes et substitutions

Définition 1.1.1. *Langage du premier ordre [DNR] p 11*

Définition 1.1.2. *Terme sur un langage [DNR] p 12*

Définition 1.1.3. *Terme clos [DNR] p 13*

Définition 1.1.4. *Hauteur d'un terme [DNR] p 13*

Définition 1.1.5. *Substitution [LdR] p 85*

Définition 1.1.6. *Domaine d'une substitution [LdR] p 85*

Définition 1.1.7. *Substitution appliquée à un terme (inductivement) [LdR] p 86*

Attention à la notation ! prendre [DNR]

Exemple 1.1.1. *??*

1.2 Unification

Définition 1.2.1. *Deux termes unifiables + unificateur [DNR] p 248*

Définition 1.2.2. *Filtre [DNR] p 248*

Rq : Notation + $u[\sigma][\sigma'] = u[\sigma' \circ \sigma]$ [DNR] p 248

Exemple 1.2.1. *[DNR] p 248*

Définition 1.2.3. *unificateur d'un ensemble de couples de termes (= équations) [LdR] p 87*

Définition 1.2.4. *unificateur principal [DNR] p 249 (pour un ensemble d'équations ?)*

1.3 Algorithmes d'unification

Définition 1.3.1. *Problème : Entrée : Un ensemble fini d'équations Sortie : un unificateur principal [DNR] p 249*

Théorème 1.3.1. *Existence d'un principal quand unificateur [DNR] p 249*

Définition 1.3.2. *Algorithme d'unification [DNR] p 249*

Théorème 1.3.2. *Il est correct et termine [DNR] p 250*

Exemple 1.3.1. *[DNR] p 250*

Autre algo ?

2 Application à la résolution

2.1 Méthode de résolution

Rq : mise sous forme de clauses [DNR]

Définition 2.1.1. *règles res et contr [DNR] p 265*

Exemple 2.1.1. *[DNR] p 265*

Définition 2.1.2. *Ensemble de clauses vrai, contradictoire, [inconsistant] inCohérent [DNR] p 267*

Proposition 2.1.1. *Correction de la résolution [DNR] p 267*

2.2 Modèles de Herbrand et complétude

Définition 2.2.1. *Modèle et structure de Herbrand [LdR] p 98*

Lemme 2.2.1. *modele ssi modele de Herbrand [LdR] p 99*

Définition 2.2.2. *instance [LdR] p 99*

Proposition 2.2.1. *pas de modèle de Herbrand alors ... [LdR] p 99*

Lemme 2.2.2. *filtre [DNR]*

Proposition 2.2.2. *Complétude de la résolution [DNR] p 267*

3 Application à la réécriture

Définition 3.0.3. *Règle et système de réécriture [Baa] p 61*

Définition 3.0.4. *Système confluent [Baa] p 9*

Définition 3.0.5. *Système localement confluent [Baa] p 28*

Définition 3.0.6. *Terminant et convergeant [Baa] p 9*

Théorème 3.0.1. *Lemme de Newman [Baa] p 28*

Définition 3.0.7. *Paire critique [Baa] p 139*

Exemple 3.0.1. *[Baa] p 139*

Lemme 3.0.3. *Des paires critiques [Baa] p 140*

Théorème 3.0.2. *Paires critiques [Baa] p 140*

Développements

- Algo d'unification
- Complétude de la résolution

Références

- DNR
- Lassaigne-Rougemont

Leçon 920 : Réécriture et formes normales. Exemples.

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.

Est-ce qu'on commence par des exemples ?

1 Définition d'un système de réécriture

1.1 Termes et substitutions

Définition 1.1.1. *signature [Baa] p 34*

Exemple 1.1.1. *Signature de la théorie des groupes [Baa] p 34*

Définition 1.1.2. *Terme [Baa] p 35*

Rq : on peut représenter les termes par des arbres [Baa] p 35

Exemple 1.1.2. *[Baa] p 35,36 (avec arbre)*

Définition 1.1.3. *Position, taille d'un terme, sous-terme, remplacer un sous terme, variable d'un terme + notations [Baa] p 37*

Exemple 1.1.3. *[Baa] p 37*

Définition 1.1.4. *Substitution et son domaine + extension aux termes par induction[Baa] p 38*

Définition 1.1.5. *Instance d'un terme [Baa] p 39*

1.2 Système de réécriture

Définition 1.2.1. *Théorie équationnelle [Baa] p 50*

Exemple 1.2.1. *[Baa] p 40*

Exemple 1.2.2. *[Baa] p 53*

Définition 1.2.2. *Règle et système de réécriture [Baa] p 61*

Rq : un système de réécriture peut être obtenu à partir d'une théorie équationnelle en orientant les égalité
Rq : on peut appliquer les règles de réécriture n'importe ou dans le terme et pas seulement à la racine

Exemple 1.2.3. *[Baa] p 40*

Définition 1.2.3. $\leftrightarrow^*$ *[Baa] p 8*

Théorème 1.2.1. $\leftrightarrow^*$ *plus petite relation d'équivalence telle que .. [Baa] p 41*

2 Propriété des systèmes de réécriture (SR) et formes normales

2.1 Formes normales

Définition 2.1.1. *terme réductible et forme normale [Baa] p 9*

Exemple 2.1.1. *[Baa] p 9*

Définition 2.1.2. *Forme normale d'un terme +notation [Baa] p 9*

Exemple 2.1.2. *24 est une forme normale de* $(3+5)(1+2)$

2.2 Confluence

Définition 2.2.1. *Termes joignables [Baa] p 9*

Exemple 2.2.1. *[Baa] p 9*

Définition 2.2.2. *Système confluent [Baa] p 9*

Définition 2.2.3. *Système localement confluent [Baa] p 28*

Définition 2.2.4. *SR church-rosser [Baa] p 9*

Théorème 2.2.1. *confluent ssi Church Rosser [Baa] p 11*

Proposition 2.2.1. *Si confluent, alors au plus UNE forme normale [Baa] p 12*

Définition 2.2.5. *Système normalisant [Baa] p 9*

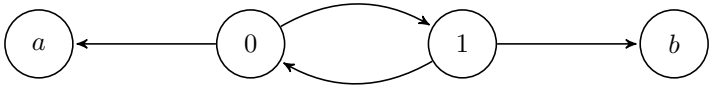
Théorème 2.2.2. *Si normalisant et confluent, alors* $\exists!$ *forme normale pour chaque terme [Baa] p 12*

Théorème 2.2.3. *si confluent et normalisant :* $x \leftrightarrow^* y$ *ssi ils ont la même forme normale [Baa] p 12*

2.3 Terminaison

Définition 2.3.1. *Terminant et convergeant [Baa] p 9*

Théorème 2.3.1. *Lemme de Newman [Baa] p 28*



Exemple 2.3.1.

Théorème 2.3.2. *Birkhoff [Baa] p 55*

3 Prouver la terminaison et la confluence

3.1 Ordre de réduction et terminaison

Théorème 3.1.1. *Indécidabilité de la terminaison* *[Baa] p 99*

Théorème 3.1.2. *Décidabilité dans le cas d'un système clos à droite [Baa] p 99,100*

Définition 3.1.1. *Ordre de réduction [Baa] p 102*

Exemple 3.1.1. *[Baa] p 102*

Théorème 3.1.3. *Terminaison ssi ∃ ordre bien fondé [Baa] p 103*

3.2 Confluence et paires critiques

Définition 3.2.1. *Problème de la confluence [Baa] p 134*

Définition 3.2.2. *Paire critique [Baa] p 139*

Exemple 3.2.1. *[Baa] p 139*

Lemme 3.2.1. *et thm Des paires critiques* *[Baa] p 140*

Rq : pour savoir si un système est confluent, il suffit de considérer les paires critiques et de regarder si elles sont joignables. Si elles ne le sont pas, on ajoute une règle pour les rendre.
On en déduit la procédure de Knuth Bendix qui prend en entrée un système équationnel et un ordre de réduction et qui renvoie un système de réécriture fini, équivalent et convergent KNUTH BENDIX

Proposition 3.2.1. *Complétion de Knuth Bendix (Terese ?)*

Développements

- Indécidabilité de la terminaison d'un système de réécriture
- Théorème des paires critiques

Références

- Baader

Leçon 921 : Algorithmes de recherche et structures de données associées.

Rapport du jury

RIEN

Définition 0.2.3. *Dictionnaire - ensemble de clés/valeurs [Cor] p 211 [FG] p 172*

On s'intéresse aux opérations : rechercher, insérer, supprimer
On suppose les clés distinctes

1 Recherche dans une liste ou un tableau

1.1 Recherche séquentielle dans une liste non triée

On compare l'élément cherché à tous les éléments de la liste.
Complexités.
Rq : dans le cas d'une liste triée, la complexité est la même.

1.2 Recherche dichotomique (dans un tableau trié)

Algo (occurrence quelconque) [Cor] p 178
Complexité [Cor] p 188

Proposition 1.2.1. *Optimalité [FG] p 190*

1.3 Recherche par rangs (dans un tableau non-trié)

- Entrée : un tableau A et un entier i
- Sortie : l'élément de A qui est plus grand que $i-1$ autres éléments de A

[FG] p 197
Recherche du min ou du max Algo + complexité [Cor] p 198
Diviser pour régner Regarder partition [Cor] p 158 + Sélection randomisée [Cor] p 200
Complexités en moyenne et pire cas [Cor] p 200,202
Rq : Les opérations insérer et supprimer sont en : $O(n)$ [FG] p 69. On veux améliorer ces complexités

2 Arbres binaires de recherche (ABR)

2.1 Définition et recherche

Définition 2.1.1. *Arbre binaire de recherche [FG] p 198*

- Rechercher : algo + complexité [FG] p 199
- Insérer : algo + complexité [FG] p 199
- Supprimer : algo + complexité [FG] p 205

(Complexité : O(hauteur))

Lemme 2.1.1. *Hauteur entre* $\log n$ *et* n *[FG] p 208*

Proposition 2.1.1. *Complexités en moyenne et pire cas (arbre dégénéré) [FG] p 213-214*

2.2 Arbres équilibrés : les AVL

Définition 2.2.1. *Arbre équilibré [FG] p 221*

Conséquence sur la complexité [FG] p 222
RotationsPrincipe + dessins en annexe [FG] p 222

Proposition 2.2.1. *Les rotations conservent la propriété d'ABR [FG] p 224*

Temps constant ???
AVL

Définition 2.2.2. *AVL ou abre H-équilibré [FG] p 224*

Proposition 2.2.2. *Hauteur d'un AVL [FG] p 225*

Ajout et suppression : comme dans les ABR [FG] p 227
Si déséquilibre : rééquilibrage -> Principe [FG] p 230

Proposition 2.2.3. *Complexités en $O(n)$, rééquilibrages compris [FG] p 227*

Autres types d'arbres ?

2.3 Application : codage de Huffman

Principe de l'algo (compression de texte) + complexité [Cor] p 396 Autre appli : intersection de segments ?

2.4 ABR optimaux

ABRO [Cor]

3 Méthodes de hachage

3.1 Principe général et fonction de hachage

Principe [FG] p 261

Exemple 3.1.1. *Méthode de la division [Cor] p 244*

Proposition 3.1.1. *Gestion des collisions par chaînage [FG] p 262*

Proposition 3.1.2. *Gestion des collisions par calcul [FG] p 262*

Principe de recherche [FG] p 262
Rq : Le choix de la fonction de hachage est fondamental, pour qu'il y ait le moins de collisions [FG] p 261

Définition 3.1.1. *Hachage uniforme [FG] p 261*

3.2 Hachage universel

Rq : si la fonction de hachage est connue : utilisateurs malveillants [Cor] p 246
Principe et complexité [Cor] p 246, 248

3.3 Hachage parfait

Hachage parfait [Cor]

4 Conclusion

Tableau de comparaison des complexités [FG] p 298

Développements

- Hachage parfait
- ABR optimal

Références

- Cormen
- Froidevaux

Leçon 922 : Ensembles récursifs, récursivement énumérables. Exemples.

Rapport du jury

RIEN

On considère les fonctions définies sur un sous-ensemble de $\mathbb{N}^p$, à valeurs dans $\mathbb{N}$

1 Fonctions récursives et machines de Turing

1.1 Fonctions récursives primitives

Définition 1.1.1. *Fonctions de base, fonction composée et schéma de récurrence primitive [LdR] p 138*

Exemple 1.1.1. *Addition [LdR] p 138*

Définition 1.1.2. *Fonction récursive primitive [LdR] p 139*

Exemple 1.1.2. *Somme et produit finis [LdR] p 140*

Exemple 1.1.3. *Ackerman n'est PAS primitive récursive [LdR] p 143*

1.2 Fonctions récursives non-primitives

Définition 1.2.1. *Fonction partielle [LdR] p 145*

Définition 1.2.2. *Schéma de minimisation [LdR] p 145*

Définition 1.2.3. *Fonction récursive partielle [LdR] p 146*

Exemple 1.2.1. *Ackermann est récursive [CLb] p 18*

1.3 Machines de Turing

Définition 1.3.1. *Machine de Turing [Car] p 141*

Définition 1.3.2. *Etape de calcul et calcul [Car] p 143*

Mode accepteur :

Définition 1.3.3. *Langage accepté [Car] p 143*

Mode calculateur :

Définition 1.3.4. *Fonction partielle associée à une MT [LdR] p 123*

1.4 Équivalence ?

Définition 1.4.1. *Thèse de Church [LdR] p 146 [Car] p 186*

Théorème 1.4.1. *Fonction récursive ⇒ calculable [Car] p 185*

Théorème 1.4.2. *Calculable ⇒ récursive* *[Car]*

2 Ensembles récursifs et récursivement énumérables

2.1 Ensembles récursifs

Définition 2.1.1. *Ensemble récursif primitif [LdR] p 139*

Définition 2.1.2. *Ensemble récursif [LdR] p 146*

Définition 2.1.3. *Ensemble décidable [Car] p 158*

Proposition 2.1.1. *Les ensembles récursifs correspondent aux langages décidés par MT [Car] p 158*

2.2 Ensembles récursivement énumérables

Définition 2.2.1. *Récursivement énumérable [LdR] p 161*

Proposition 2.2.1. *langage récursivement énumérable lorsque accepté par MT [Car] p 155*

Proposition 2.2.2. *Tout R est dans RE*

Définition 2.2.2. *Enumérateur [Car] p 155*

Proposition 2.2.3. *rec énum ssi énuméré par un énumérateur [Car] p 155*

Application 2.2.1. *Il existe des langages non RE [Car] p 156*

2.3 Propriétés de clôture

Proposition 2.3.1. *Récursifs stables par union intersection et complémentaire [CLb] p 42*

Proposition 2.3.2. *rec énum stable par intersection et union [CLb] p 42*

Proposition 2.3.3. *si L et L^c sont RE alors L et L^c R [CLb] p 42*

Proposition 2.3.4. *La projection d'un RE est RE [CLb] p 43*

Proposition 2.3.5. *ss ens récursivement énumérable est proj d'un ss rec prim [CLb] p43*

Application 2.3.1.
- *Graphe d'une fonction partielle réc est RE [CLb] p 43 - La réciproque est vraie*
- *L'image d'une FPR est RE [CLb] p 44*
- *ss ens RE non vide est l'image d'une FPR [CLb] p 44*

3 Exemples (et applications ?)

Définition 3.0.1. *Réduction [Car] p 160*

Proposition 3.0.6. *réduction et décidabilité [Car] p 161*

Corollaire 3.0.1. *Réduction et indécidabilité [Car] p 161*

3.1 Problèmes sur les langages

Définition 3.1.1. *L_ϵLangage d'acceptation [Car] p 159*

Proposition 3.1.1. *L_ϵ RE [Car] p 159*

Proposition 3.1.2. *L_ϵ pas R ie indécidable [Car] p 159*

Corollaire 3.1.1. *$\overline{L_\epsilon}$ pas RE [Car] p 160*

Application 3.1.1. *cide et différents indécidables [Car] p 161*

Théorème 3.1.1. *Pb de l'arrêt indécidable [Wol] p 146*

Théorème 3.1.2. *arrêt existentiel [Wol] p 147*

Théorème 3.1.3. *arrêt universel [Wol] p 148*

Théorème 3.1.4. *Rice [Car] p 162*

Corollaire 3.1.2. *L_{rat} indécidable [Sip] p 218*

Définition 3.1.2. *MT linéairement bornée [Car] p 171*

Proposition 3.1.3. *arrêt sur LB décidable [Car] p 174*

Proposition 3.1.4. *vide sur LB indécidable [Car] p 176*

3.2 Problème de correspondance de Post

Définition 3.2.1. *PCP [Car] p 163*

Proposition 3.2.1. *PCP pas R, mais RE [Car] p 165*

Corollaire 3.2.1. *pbs indécidables sur les grammaires [Car] p 167*

3.3 Décidabilité et théories logiques

Définition 3.3.1. *Théorie récursivement axiomatisable [LdR] p 172*

Définition 3.3.2. *Théorie décidable [LdR] p 172*

Proposition 3.3.1. *ensemble des fomules, preuves R si T RA [LdR] p 173*

Corollaire 3.3.1. *RA et complète $\Rightarrow$ décidable [LdR] p 173*

Théorème 3.3.1. *Décidabilité de l'arithmétique de Presburger [Car] p 178*

3.4 Réécriture

Théorème 3.4.1. *Indécidabilité de la terminaison d'un système de réécriture [Baa] p 99*

Développements

- calc ssi rec
- Presburger
- Indécidabilité de la terminaison d'un syst de rééc

Références

- Lassaigne Rougemont
- Cori Lascar
- Carton
- Sipser
- Wolper
- Baader

Leçon 923 : Analyse lexicale et syntaxique : applications.

Rapport du jury

1 Compilation

La compilation traduit un programme source compréhensible par l'homme en un programme source compréhensible par l'ordinateur. Exs : attention différence compilation interprétation
Elle se compose d'une phase d'analyse qui génère un arbre syntaxique et d'une phase de génération qui la transforme en code machine. On s'intéresse ici à la phase d'analyse qui comprend une analyse lexicale et une analyse syntaxique. [Sch] p 134

2 Analyse lexicale

2.1 Définitions préliminaires

Définition 2.1.1. *Unité lexicale [ALSU] p 103*

Définition 2.1.2. *Motif [ALSU] p 103*

Les motifs sont représentés par une expression rationnelle

Définition 2.1.3. *léxèmes [ALSU] p 103*

Rq : le nom d'une unité lexicale correspond à une classe de lexèmes, tous décrits par un même motif.

Définition 2.1.4. *Expression rationnelle [Car] p 39*

Exemple 2.1.1. *[Sch]*

2.2 Reconnaissance des lexèmes

Définition 2.2.1. *Automate fini [Car] p 39*

Théorème 2.2.1. *Kleene [Car] p 40*

Proposition 2.2.1. *Automate fini déterministe minimal équivalent [Sch] p 23*

Exemple 2.2.1. *Nombres réels [Sch] p 23*

2.3 Construction de l'automate pour l'analyse lexicale

[?] p 173 [?] p 236
Automate minimal pour chaque motif, automate union. Ordre de priorité sur les motifs, état final.

2.4 Algorithme glouton

[?] p 237 [Sch] p 23

3 Analyse syntaxique

Elle se fait sur les noms des unités lexicales

3.1 Grammaire et arbre de décision

Définition 3.1.1. *Grammaire algébrique [Car] p 83*

Exemple 3.1.1. *[Sch] p 39*

Définition 3.1.2. *Automate à pile [Car] p 115*

Théorème 3.1.1. *algébrique ssi reconnu par un automate à pile [Car] p 119*

Définition 3.1.3. *Dérivation [Sch] p 37*

Exemple 3.1.2. *[Sch] p 39*

But : construire l'arbre de dérivation. Une méthode générique serait par essai/erreur

3.2 CYK : un algorithme générique

Algorithme et analyse [Car] p 199, [FB] p 318
Décide si un mot appartient au langage engendré par une grammaire propre G. Il est polynômial en la taille de la grammaire et cubique en la taille du mot.
Autres méthodes pas génériques, et parfois plus couteuses en la taille de la grammaire MAIS linéaires en la taille du texte :

3.3 Analyse descendante dans le cas LL(1)

On construit l'arbre à partir de la racine. Soit G une grammaire algébrique.

Définition 3.3.1. *Premier, Suivant [?] p 226-227*

Définition 3.3.2. *Grammaire LL(1) [Sch] p 66,74*

Proposition 3.3.1. *Calcul pratique de premier et suivant [ALSU] p 203,204*

Exemple 3.3.1. *[Sch] p 75*

Rq : on regarde un caractère en avant ; construction d'une table prédictive

Proposition 3.3.2. *Calcul d'une table prédictive [?] p 248*

Exemple 3.3.2. *[?] p 249*

3.4 Analyse ascendante

On construit l'arbre à partir des feuilles

3.4.1 Analyse LR(0)

Définition 3.4.1. *item [Sch] p 91*

Définition 3.4.2. *Règle de réduction [Sch] p 92*

Définition 3.4.3. *Cloture [Sch] p 92*

Définition 3.4.4. *Automate LR(0) [Sch] p 94*

Définition 3.4.5. *Grammaire LR(0) [Sch] p 101*

Exemple 3.4.1. *LR(0) ou pas [Sch] p 101, 116*

3.4.2 Analyse LR(1)

Définition 3.4.6. *item LR(1) [Sch] p 121*

Définition 3.4.7. *Premier LR(1) [Sch] p 122*

Définition 3.4.8. *Cloture LR(1) [Sch] p 123*

Définition 3.4.9. *Automate LR(1) [Sch] p 124*

Définition 3.4.10. *Grammaire LR(1) [Sch] p 126*

Exemple 3.4.2. *Grammaire LR(1) [Sch] p 127*

3.4.3 Analyse SLR(1)

Définition 3.4.11. *Automate SLR(1) [Sch] p 144*

Définition 3.4.12. *Grammaire SLR(1) [Sch] p 144*

Rq : Analyse de type LR(0)

Exemple 3.4.3. *Construction des automates LR(0) et SLR(1) [Sch] p 144*

3.5 Inclusions

[Sch] p 263

Développements

- Construction d'un automate LR(0) sur un exemple
- CYK

Références

- Dragon
- Carton
- Schwarzentruber
- Whilhelm Maurer
-

Leçon 924 : Théories et modèles en logique du premier ordre. Exemples.

Rapport du jury

?

A travers des ensembles de formules, on cherche à décrire des structures (entiers, groupes...). Un fois qu'on a un ensemble de formules (qui peut être infini), on veut savoir ce qu'on a défini : est-ce que ça correspond à des objets mathématiques qui existent, est-ce qu'il y en a qui sont remarquables, etc.

1 Syntaxe et sémantique en logique du premier ordre

On commence par regarder : quelles formules j'ai le droit d'écrire (syntaxe) ; et quelle est leur signification (sémantique)

1.1 Syntaxe

Définition 1.1.1. *Langage au premier ordre [DNR] p 11*

Rq : on ne considère que des langages égalitaires qui ont au moins un symbole de constante

Exemple 1.1.1. *Langage de la théorie des groupes [DNR] p 12*

Soit V un ensemble infini dénombrable de variables [DNR] p 12

Définition 1.1.2. *Terme ; terme clos [DNR] p 12*

Exemple 1.1.2. *Un terme et un terme clos (th des groupes) [DNR] p 13*

Définition 1.1.3. *Formules atomiques [DNR] p 15*

Définition 1.1.4. *formules [DNR] p 15*

Exemple 1.1.3. *Formule en th des groupes [DNR] p 16*

1.2 Sémantique et modèles

Définition 1.2.1. *Structure [DNR] p 68*

Définition 1.2.2. *Valuation [DNR] p 69*

Définition 1.2.3. *Valeur d'un terme [DNR] p 69 (notations [LdR])*

Définition 1.2.4. *Valeur d'une formule [DNR] p 70*

Proposition 1.2.1. *La valeur d'une formule ne dépend que de la valuation des variables libres [LdR] p 59*

Notation : $\mathcal{M}, \rho \models F$ [DNR] p 71

Exemple 1.2.1. *[DNR] p 71*

Définition 1.2.5. *Formule valide [DNR] p 72 [LdR] p 60*

Définition 1.2.6. *Formules équivalentes [DNR] p 75*

1.3 Théories

Définition 1.3.1. *Théorie [DNR] p 79*

Exemple 1.3.1. *Arithmétique de Péano [DNR] p 111*

Définition 1.3.2. *Modèle d'une théorie [CLa] p 178*

Définition 1.3.3. *Théorie consistante, contradictoire, conséquence ; équivalence [CLa] p 178*

Exemple 1.3.2. *PA consistante ($\mathbb{N}$) [DNR] p 112*

Proposition 1.3.1. *$T \models^* G$ ssi $T\{\neg G\}$ contradictoire [CLa] p 181*

Théorème 1.3.1. *Lowenheim Skolem [DNR]*

2 Déduction : méthode de résolution

2.1 Mise sous forme de clauses

Définition 2.1.1. *Littéral [DNR] p 264*

Définition 2.1.2. *Clause [DNR] p 264*

Mise d'un ensemble de formules sous forme de clause [LdR] p 84 Forme de Skolem : thm sur l'existence de modèles !

2.2 Méthode de résolution/correction

Définition 2.2.1. *Termes unifiables, unificateur [DNR] p 248*

Définition 2.2.2. *Filtre [DNR] p 248*

Théorème 2.2.1. *Unificateur le plus général [DNR] p 249*

But : on prend un ensemble de clauses E et on veut dériver la clause vide.

Définition 2.2.3. *Méthode de résolution [DNR] p 265*

Rq : on définit les unificateurs sur les littéraux comme sur les termes

Exemple 2.2.1. *(preuve en annexe) [DNR] p 265*

Définition 2.2.4. *INCOHERENT lorsqu'on peut dériver la clause vide [DNR] p 267*

Proposition 2.2.1. *Correction : incohérent ⇒ contradictoire*

2.3 Modèles de Herbrand et complétude

Définition 2.3.1. *Domaine et structure de Herbrand [LdR] p 98*

Définition 2.3.2. *Modèle de Herbrand [LdR] p 99*

Proposition 2.3.1. *modèle ssi modèle de Herbrand [LdR] p 99*

Lemme 2.3.1. *dérivation d'un littéral qui filtre [DNR] p 266 (utilisé mais pas montré dans le dvpt)*

Théorème 2.3.1. *Complétude de la méthode de résolution [DNR] p 267*

2.4 Applications

Théorème 2.4.1. *Compacité [DNR] p 86*

Application 2.4.1. *L-S ascendant [DNR] p 87*

Application 2.4.2. *Il existe des modèles non isomorphes de PA [DNR] p 112*

3 Décidabilité

Définition 3.0.1. *Théorie décidable [DNR] p 125*

Proposition 3.0.1. *PA indécidable [DNR] p 126*

Proposition 3.0.2. *Presburger décidable [DNR] p 126*

Développements

- Lowenheim Skolem descendant
- Complétude de la résolution
- Presburger

Références

- David Nour Raffali
- Lassaigne-Rougemont
- Cori-Lascar

Leçon 925 : Graphes : représentations et algorithmes

Rapport du jury

...

1 Définitions

1.1 Définitions de graphes

Définition 1.1.1. *Graphe orienté [FG] p 137*

Définition 1.1.2. *Graphe non-orienté [FG] p 137*

Définition 1.1.3. *Graphe pondéré (valué) [FG] p 138*

Exemple 1.1.1. *3.1 [Car] p 136*

1.2 Terminologie

Définition 1.2.1. *successeur/prédécesseur [FG] p 138*

Définition 1.2.2. *Sommets adjacents [FG] p 139*

Définition 1.2.3. *Degré entrant/sortant [FG] p 139*

Exemple 1.2.1. *1.1.1*

Définition 1.2.4. *Chemin/chaine [FG] p 140*
+coût ! (dans le cas pondéré)

Définition 1.2.5. *Circuit/cyle [FG] p 140*

Définition 1.2.6. *graphe/composante connexe/fortement connexe [FG] p 140*

1.3 Arbres

Définition 1.3.1. *Arbre [FG] p 140*

Exemple 1.3.1. *Dessiner*

Proposition 1.3.1. *Caractérisation d'un arbre [FG] p 141*

1.4 Représentations

Définition 1.4.1. *Matrice d'adjacence (graphes denses) [Cor] p 547*

Définition 1.4.2. *Liste d'adjacence (graphes peu denses) [Cor] p 546*

Proposition 1.4.1. *Espace mémoire + complexités (comparaisons) [Cor] p 546-547*

Exemple 1.4.1. *22.1 en annexe [Cor] p 546*

2 Parcours

2.1 Parcours en largeur

Présentation
Algo + complexité [Cor] p 549

Application 2.1.1. *Dijkstra, Prim, plus court chemin [Cor] p 549 (cf plus loin)*

2.2 Parcours en profondeur

Présentation + algo + complexité [Cor] p 557-560

Application 2.2.1. *Tri topologique [Cor] p 566*

Application 2.2.2. *Composantes fortement connexes [Cor] p 568*

3 Arbres couvrants minimaux

3.1 Définition et algorithme générique

Définition 3.1.1. *Arbre couvrant [Cor] p 577*

Définition 3.1.2. *Pb de l'arbre couvrant minimal (entrée - sortie) [Cor] p 577*

Algorithme générique [Cor] p 579

3.2 Algorithme de Kruskal

Algorithme + complexité [Cor] p 584

3.3 Algorithme de Prim

Algorithme + complexité [Cor] p 586

4 Plus courts chemins dans un graphe

Entrée - Sortie [Cor] p 595

Définition 4.0.1. *Poids de plus court chemin [Cor] p 595*

4.1 Origine unique

Bellman Ford : Algo + complexité [Cor] p 602
Dijkstra : Algo + complexité [Cor] p 609

4.2 Entre toutes paires

Floyd Warshall + complexité [Cor] p 641

5 Problèmes NP-complets

5.1 Cycle hamiltonien

Définition 5.1.1. *Chemin hamiltonien [Car] p 206*

Définition 5.1.2. *Problème du chemin hamiltonien [Car] p 279*

Proposition 5.1.1. *NP-complet* *[Car] p 207*

Rq : le problème du voyageur de commerce est de trouver un CYCLE hamiltonien.

5.2 Cliques

Définition 5.2.1. *Clique [Car] p 209*

Proposition 5.2.1. *pb clique de taille k NP complet [Car] p 209*

Développements

- HAM est NP complet
- BF

Références

- Froidevaux
- Cormen
- Carton

Leçon 926 : Analyse des algorithmes : complexité. Exemples.

Rapport du jury

...

1 Algorithmes et complexités

1.1 Complexité

Définition 1.1.1. *Algorithme [FG] p 5*

Objectif : analyser la complexité ; comparer les algos [FG] p 13

Définition 1.1.2. *Complexité en temps, en espace [FG] p 13 ; 14,15 (temps)*

Exemple 1.1.1. *Analyse du tri par insertion [Cor] p 22*

Exemple 1.1.2. *matrice ; Recherche séquentielle [FG] p 21,22*

Définition 1.1.3. *Complexité en pire cas, cas moyen, meilleur cas [FG] p 17 ! p_n*

Proposition 1.1.1. $\min \leqslant moy \leqslant \max$ *[FG] p 21*

Exemple 1.1.3. *Matrice ; recherche séquentielle [FG] p 21,22*

1.2 Comparaison des algorithmes, échelles de grandeurs

Rq : en général, une approximation suffit pour savoir si un algo est utilisable ou pour comparer deux algorithmes [FG] p 23

Définition 1.2.1. *Notation Θ, O, Ω [Cor] p 40-44*

Exemple 1.2.1. *Multiplication matrice ; recherche séquentielle [FG] p 21,22*

1.3 Optimalité

Définition 1.3.1. *complexité optimale [FG] p 27*

Exemple 1.3.1. *Optimalité du tri [Cor] [FG]*

2 Méthodes de calcul de complexité

2.1 Calcul direct

Exemple 2.1.1. *Un algorithme itératif : CYK [Car] p 199*

Exemple 2.1.2. *Tri par tas [Cor]*

Exemple 2.1.3. *Analyse du tri rapide* *[Cor] [FG]*

2.2 Diviser pour régner

Théorème 2.2.1. *général [Cor] p 86*

Corollaire 2.2.1. *théorème fondamental [Pap] p 49*

Exemple 2.2.1. *Tri fusion ; points les plus proches ; TFR [Cor]*

3 Analyse amortie

Définition 3.0.1. *Analyse amortie + blabla [Cor] p 417*

Exemple 3.0.2. *Table dynamique [Cor] p 428*

3.1 Méthode de l'agrégat

Définition 3.1.1. *Méthode de l'agrégat [Cor] p 418*

Exemple 3.1.1. *Opérations de pile [Cor] p 418*

3.2 Méthode comptable

Définition 3.2.1. *Méthode comptable [Cor] p 422*

Union find ?

Exemple 3.2.1. *Incrémentation d'un compteur binaire [Cor] p 424*

3.3 Méthode du potentiel

Définition 3.3.1. *Méthode du potentiel [Cor] p 424*

Exemple 3.3.1. *Table dynamique [Cor] p 434*

4 Améliorations ?

4.1 Compromis temps/espace

Parfois, c'est intéressant d'augmenter la complexité spatiale pour diminuer la complexité temporelle et inversement

Exemple 4.1.1. *Calcul de Fibo naïf et par programmation dynamique [Cor] p 902*

4.2 Choix de structure de donnée

Selon la structure de donnée choisie, les opérations fondamentales n'ont pas le même coût.

Exemple 4.2.1. *graphes représentés par listes ou matrice d'adjacence...*

Exemple 4.2.2. *ABR optimaux* *[Cor]*

Développements

- Analyse du tri rapide
- ABR optimaux

Références

- Froidevaux
- Cormen

Leçon 927 : Exemples de preuves d'algorithmes : correction, terminaison.

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.

1 Terminaison

1.1 Ensembles bien fondés

Définition 1.1.1. *Ensemble bien fondé [Alb] p 83*

Exemple 1.1.1. $\mathbb{N}$ *+ ordre lexicographie [Alb] p 83*

1.2 Algorithmes itératifs

Définition 1.2.1. *Variant de boucle Sans ref!!*

Proposition 1.2.1. *Si une boucle admet un variant, alors elle termine Sans ref*

Exemple 1.2.1. *Boucle "for" sans ref*

Exemple 1.2.2. *Kruskal, Prim, Bellman Ford ? ? [Cor]*

Exemple 1.2.3. *Unification [DNR] p 248*

1.3 Algorithmes récursifs

Théorème 1.3.1. *Terminaison [Alb] p 85*

Exemple 1.3.1. *Binôme, ackerman [Alb] p 86*

Exemple 1.3.2. *Diviser pour régner, ex : tri fusion ; points les plus proches[Cor] p 26*

1.4 Indécidabilité

Exemple 1.4.1. *Pas toujours de variant : Collatz [Alb] p 87*

Théorème 1.4.1. *Arrêt [Wol] p 146*

2 Correction

Définition 2.0.1. *Correction partielle / correction totale (=correction + terminaison) Sans ref*

2.1 Algorithmes itératifs

Définition 2.1.1. *Invariant de boucle (init + conservation) [Cor] p 16*

Rq : terminaison : c'est vrai à la fin de la boucle [Cor] p 16
Rq : Aide pour la terminaison ?

Exemple 2.1.1. *Reprendre les exs de la partie I + tri insertion ? [Cor]*

Exemple 2.1.2. *L'algo d'unification est totalement correct [DNR]*

Est-ce qu'on a un exemple de correction partielle ? ? ?

2.2 Algorithmes récursifs

Proposition 2.2.1. *Induction bien fondée [Alb] p 89*

Théorème 2.2.1. *Correction [Alb] p 88*

Exemple 2.2.1. *Tri fusion ; factorielle [Cor] p 53*

Exemple 2.2.2. *Pas toujours induction bien fondée*
Points les plus proches totalement correct [Cormen]

3 Logique de Hoare

On veut formaliser la preuve d'algorithme

3.1 Triplet de Hoare

On travaille à partir d'un langage de programmation constitué de : ... [Win] p 12

Définition 3.1.1. *Triplet de Hoare (Précondition programme postcondition)* $\{A\}c\{B\}$ *[Win] p 78*

Exemple 3.1.1. *Des fois ça ne termine pas : ex + notation* $[]$ *[Win] p 79*

Définition 3.1.2. *Notation :* $\models$ *[Win] p 79*

Rq : on ne peut pas montrer $\models$, on va essayer de le prouver

3.2 Notion de preuve

Définition 3.2.1. *Règles de la logique de Hoare [Win] p 89*

Exemple 3.2.1. *Arbre de preuve [Win] p 93*

Développements

- Algorithme d'unification
- Points les plus proches

Références

- Albert
- Cormen
- Wolper
- Winskel

Leçon 928 : Problèmes NP-complets : exemples et réduction.

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.

Intro : On a un problème à résoudre, et on voudrait le faire efficacement ; c'est pas toujours possible !

1 NP-complétude

1.1 Problème de décision et classes de complexité

Définition 1.1.1. *Problème de décision [Car] p 139*

Rq : on ramène le problème de décision à un langage constitué des codages de ses instances positives, pour un codage raisonnable. !

Exemple 1.1.1. *2-SAT [Car] p 966 + [Car] p 139*

Rq : on ne s'intéresse qu'à des problèmes décidables donc les machines s'arrêtent sur toute entrée [Car] p 193

Définition 1.1.2. *Complexité en temps [Car] p 194*

Définition 1.1.3. *P et NP [Car] p 196*

Proposition 1.1.1. *$P \subset NP$ [Car] p 197*

Proposition 1.1.2. *2-SAT $\in P$; 3 SAT $\in NP$ [Cor] p 966*

1.2 Réduction polynômiale

Définition 1.2.1. *Réduction polynômiale $\leqslant_P$[Car] p 202*

Proposition 1.2.1. *si $B \in P$ ou NP alors $A \in P$ ou NP [Car] p 202 [?] p 64*

1.3 Classe NP

Définition 1.3.1. *Vérificateur en temps polynômial [Car] p 200*

Proposition 1.3.1. *Carac des NP [Car] p 200*

Définition 1.3.2. *SAT [Car] p 203*

Exemple 1.3.1. *SAT est dans NP [Car] p 203*

Définition 1.3.3. *NP-difficulté ; NP-complétude [Car] p 202*

Théorème 1.3.1. *Cook [Car] p 203*

Proposition 1.3.2. *Réduction et NP difficulté [Car] p 203*

Exemple 1.3.2. *3-SAT est NP dur, et même NP complet [Car] p 203*

2 Problèmes NP complets sur les graphes

2.1 Problème de la clique

Définition 2.1.1. *Problème de la clique optimisation + décision [Cor] p 1000*

La première fois, on écrit les deux, et ensuite on dit à l'oral le pb d'optimisation et on écrit seulement le problème de décision associé

Théorème 2.1.1. *Le problème de la clique est NP-complet, par réduction de 3-SAT [Cor] p 1000*

Conséquences

Définition 2.1.2. *Couverture de sommet ; optimisation + décision [Cor] p 1003*

Théorème 2.1.2. *NP-complet par réduction de Clique [Cor] p 1003*

2.2 Problème du chemin hamiltonien

[FB] Est-ce qu'on peut faire direct cycle ham ?

Définition 2.2.1. *Problème Chemin HAM [Car] p 206*

Théorème 2.2.1. *NP complet par réduction de SAT [Car] p 207*

Conséguence

Définition 2.2.2. *Problème de CYCLE hamiltonien [Cor] p 1004*

Théorème 2.2.2. *NP complet par réduction de chemin ham OU de couverture de sommets [Cor] p 1004*

Définition 2.2.3. *Voyageur de commerce : (optimisation +) décision [Cor] p 1008*

Théorème 2.2.3. *NP complet par réduction de cycle hamiltonien [Cor] p 1008*

3 Autres problèmes NP-complets

3.1 Somme de sous-ensemble

Définition 3.1.1. *Somme-sous-ensemble : (optimisation +) décision [Cor]*

Théorème 3.1.1. *NP complet par réduction de 3 SAT [Cor] p 1011*

Définition 3.1.2. *Problème du sac à dos [FB] p 554*

Corollaire 3.1.1. *Sac à dos est NP complet (par réduction de SSE) [FB] p 554*

3.2 Problème de séparation par automate

Définition 3.2.1. *Problème de séparation par automate [FB] p 556*

Théorème 3.2.1. *NP complet par réduction de SAT [FB] p 556*

Conclusion : Contourner la NP-difficulté

On a vu : parfois si on restreint les instances, le problème devient P.
On peut essayer d'améliorer la complexité des algorithmes en général en utilisant des méthodes comme : glouton, backtracking, ...
Ou, quand on est face à un problème d'optimisation, on peut essayer d'APPROXIMER les solutions optimales : algorithmes d'approximation

Développements

- HAM est NP complet
- Cook

Références

- Carton
- Cormen
- Floyd Begel

Références

[Alb] Luc Albert. *Cours et exercices d'informatique.*

[ALSU] Alfred Aho, Monica Lam, Ravi Sethi, and Jeffrey Ullman. *Compilateurs : principes, techniques et outils.*

[Auta] J-M Autebert. *Calculabilité et complexité.*

[Autb] J-M Autebert. *Théorie des langages et des automates.*

[Baa] Franz Baader. *Term rewriting and all that.*

[BBC] Beauquier, Berstelle, and Chrétienne. *Eléments d'algorithmique.*

[BY] Jeand-Daniel Boissonnat and Mariette Yvinec. *Géométrie algébrique.*

[Car] Olivier Carton. *Langages formels.*

[CLa] René Cori and Dianel Lascar. *Logique mathématique, Tome 1.*

[CLb] René Cori and Dianel Lascar. *Logique mathématique, Tome 2.*

[Cor] Cormen. *Algorithmique.*

[Cro] Crochemore. *Algorithmique du texte.*

[DaML] Stéphane Devismes and Pascal Lafourcade ans Michel Levy. *Logique et démonstration automatique.*

[dB] Mark de Berg. *Ray shooting, Depths Orders and Hidden Surface Removal.*

[Deh] Patrick Dehornoy. *Mathématiques de l'informatique.*

[DNR] René David, Karim Nour, and Christophe Raffalli. *Introduction à la logique.*

[FB] Floyd and Beigel. *Language of machines.*

[FG] Christine Froidevaux and Marie-Claude Gaudel. *Types de données et algorithmes.*

[HR] Huth and Ryan. *Logic in Computer Science.*

[LdR] Richard Lassaigne and Michel de Rougemont. *Logique et fondements de l'informatique.*

[Mar] Xavier Marsault. *Compression et cryptage des données multimédia.*

[Pap] Papadimitriou. *Algorithms.*

[Per] Sylvain Perifel. *Complexité algorithmique.*

[PS] Preparata-Shamos. *Computational geometry.*

[Sak] Sakarovitch. *Eléments de théorie des automates.*

[Sch] François Schwarzentruber. *Compilation : analyse lexicale et syntaxique.*

[Sip] Michael Sipser.

[Win] Glynn Winskel. *The formal semantics of programming languages.*

[WM] R. Wilhelm and D. Maurer. *Les compilateurs.*

[Wol] Pierre Wolper. *Introduction à la calculabilité.*