

## Programme de colle - Semaine 19 (semaine du 9 février au 13 février)

*La démonstration des énoncés marqués d'une étoile est exigible*

### 1 Décidabilité et complexité

Dans ce cours on s'intéresse à la hiérarchisation des **problèmes de décision**. On ne parle pas de machine de Turing, le modèle de calcul a été nommé machine et peut désigner au choix un algorithme décrit en pseudo-code ou une fonction écrite en C ou en OCaml.

- Notion de problème de décision.
- Problèmes décidables : ce sont les problèmes de décision pour lesquels on a un algorithme qui répond à la question.
- Classe **P** : ce sont les problèmes de décision pour lesquels on a un algorithme de complexité *polynomiale* qui répond à la question.
- Principe de réduction d'un problème  $A$  à un problème  $B$  noté  $A \leq B$  : il existe un algorithme qui transforme les instances de  $A$  en instances de  $B$  en préservant la réponse oui/non. Ceci signifie que le problème  $B$  est plus difficile que le problème  $A$ .
- Principe de réduction polynomiale d'un problème  $A$  à un problème  $B$  noté  $A \leq_P B$ .
- Si  $A \leq_P B$  et  $B \leq_P C$  alors  $A \leq_P C$  (\*) (par composition des deux réductions)
- Si  $B \in P$  et  $A \leq_P B$  alors  $A \in P$  (\*)
- Classe **NP** : définie comme la classe des problèmes dont les instances positives admettent des certificats qui peuvent être vérifiés par un algorithme en temps polynomial. Notion de certificat pour un problème de décision.
- $P \subset NP$  (\*)
- Classe **NP-complet** : définie comme la classe des problèmes qui sont dans **NP** et qui sont **NP-durs** (c'est-à-dire que tout problème de **NP** se réduit à lui). Exemples faits en classe : 3-SAT (réduction difficile avec la transformation de Tseitin), CNF-SAT (réduction triviale depuis 3-SAT), CLIQUE (réduction depuis 3-SAT).
- Preuve de NP-complétude par réduction avec le théorème : **Si A est NP-complet, si  $B \in NP$  et si  $A \leq_P B$  alors B est NP-complet** (\*)
- Problèmes indécidables : problème de l'arrêt HALT, **HALT est indécidable** (\*) (preuve par argument diagonal) NB: les grammaires algébriques n'ont pas été vues, le problème de correspondance de Post n'est pas au programme.

#### À savoir faire :

- Identifier un problème de décision.
- Montrer qu'un problème est décidable.
- Montrer qu'un problème est dans  $P$
- Établir une réduction entre deux problèmes de décision. En autonomie si la réduction est simple, avec un exercice guidé dans les cas plus difficiles. Exemples vus en TD : réductions

entre CLIQUE et INDEPENDENT-SET, réduction de 3-COLOR vers 4-COLOR, réduction de 3-COLOR vers  $k$ -COLOR avec  $k \geq 4$ .

- Montrer qu'un problème est dans  $NP$  (SAT, CLIQUE,  $k$ -COLOR vus en cours ou en TD) : l'élève doit savoir identifier les certificats pour un problème de décision et donner un vérificateur polynomial.
- Montrer qu'un problème est  $NP$ -complet (3-SAT, CNF-SAT et CLIQUE faits en cours) : encore une fois soit la réduction est facile (voire très facile dans le cas de l'identité), soit la réduction est difficile et dans ce cas l'exercice guide l'élève. L'exercice peut admettre qu'un problème est  $NP$ -complet pour montrer la  $NP$ -complétude d'un autre.
- Montrer qu'un problème est indécidable avec une preuve par l'absurde : le programme officiel est très peu développé sur ce point, seul **HALT** est au programme, le théorème de Rice a été abordé sous forme d'exercice. On ne peut faire que très peu de choses...

## 2 Grammaires et langages non-contextuels

- Définition d'une grammaire non-contextuelle (= hors-contexte = algébrique). Symboles non-terminaux (notés en majuscule) et terminaux (notés en minuscules). Règles de production de la forme  $X \rightarrow u$ .
- Dérivation immédiate  $u \Rightarrow v$ , dérivation  $u \Rightarrow^* v$ , dérivation gauche, dérivation droite.
- Langage engendré par une grammaire. Langages non-contextuels (= langages algébriques). Les langages réguliers sont non-contextuels (mais l'inverse n'est pas toujours vrai.)
- Exemples traités en cours :
  - $S \rightarrow aSb \mid \epsilon$  qui engendre le langage  $\{a^n b^n, n \in \mathbb{N}\}$  (\*) (démonstration par double inclusion : par récurrence sur la longueur de la dérivation dans un sens, par récurrence sur la longueur du mot dans l'autre sens)
  - $S \rightarrow aSbS \mid \epsilon$  qui engendre le langage des mots bien parenthésés (langage de Dyck).
- À savoir faire :
  - Reconnaître le langage engendré par une grammaire.
  - Écrire une grammaire pour engendrer un langage.
  - Démontrer que le langage engendré par une grammaire  $G$  est le langage  $L$  (preuve par double inclusion).

À venir (au prochain programme) : fin du chapitre : arbres de dérivation, ambiguïté .