Programme de colle - Semaine 19 (22 février)

La démonstration des énoncés marqués d'une étoile est exigible

1 Complexité et décidabilité

- Notion de problème de décision.
- Problèmes décidables. La notion de machine de Turing est hors programme, une machine est un algorithme ou un programme écrit en C ou en OCaml s'exécutant sur une machine à mémoire infinie.
- Classe **P**.
- Principe de réduction polynomiale d'un problème A à un problème B noté $A \leq_P B$.
- Si $B \in \mathbf{P}$ et $A \leq_P B$ alors $A \in \mathbf{P}$ (*)
- Classe **NP** : définie comme la classe des problèmes admettant des certificats qui peuvent être vérifiés en temps polynomial. Notion de certificat pour un problème de décision.
- Problèmes **NP**-complets : définis comme les problèmes de **NP** les plus difficiles.
- Si A est **NP**-complet, $B \in \mathbf{NP}$, et $A \leq_P B$ alors B est NP-complet (*)
- Théorème de Cook-Lévin.
- Exemples de problèmes NP-complets vus en cours ou en TD : CNF-SAT, 3-SAT, CLIQUE, INDEPENDANT, VERTEX-COVER
- Problèmes indécidables. Problème de l'arrêt HALT prenant en entrée le code d'une machine $< M >$ une entrée I pour M et décidant si la machine M termine sur l'entrée I. **HALT est indécidable (*)**.

2 Logique

- Logique propositionnelle (programme de 1ère année). En particulier les élèves doivent savoir :
  - Calculer et utiliser des tables de vérité (satisfiabilité, tautologies)
  - Utiliser l'algorithme de Quine
  - Utiliser les équivalences logiques usuelles.
  - Manipuler les formes normales.
- Logique des prédicats : seule la syntaxe est à connaître, variables libres/liées et portée d'un quantificateur. La sémantique doit être comprise intuitivement.
- Déduction naturelle : définition des séquents, dérivation d'un séquent (arbres de preuves), les règles de la déduction naturelle sont à connaître, on pourra demander la preuve de correction d'une règle au choix (*)

3 Composantes fortement connexes

- Rappels sur la connexité dans les graphes orientés ou non : relation d'accessibilité, composantes connexes, composantes fortement connexes.
- Ordre topologique sur un graphe orienté acyclique (DAG). Généralisation au cas des graphes orientés quelconques (on parle d'ordre pré-topologique) Si un sommet x est situé avant y dans un ordre pré-topologique et s'il existe un chemin de y à x alors il existe un chemin de x à y.
- Algorithme de Kosaraju (démonstration non exigible).
- Problème 2-SAT : graphe d'implications d'une instance de 2-SAT. Une insntance de 2-SAT est positive si et seulement si son graphe d'implications ne contient pas de composante fortement connexe contenant x et $\neg x$ à la fois. Algorithme pour 2-SAT. 2-SAT est dans **P**.

4 Grammaires et langages non-contextuels

- Définition d'une grammaire non-contextuelle (= hors-contexte = algébrique). Symboles non-terminaux (en majuscule) et terminaux (en minuscules). Règles de production de la forme $X \to u$.
- Dérivation immédiate $u \Rightarrow v$, dérivation $u \Rightarrow^* v$, dérivation gauche, dérivation droite. Si $u \Rightarrow^* v$ alors on peut obtenir la dérivation de u en v en utilisant uniquement des dérivations gauche (resp. 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).
- Arbre de dérivation (= arbre d'analyse). Définition à connaître précisément. $X \Rightarrow^* u$ si et seulement si il existe un arbre d'analyse de racine X la concaténation des feuilles donne u. Savoir passer d'un arbre de dérivation à une séquence de dérivations et réciproquement.
- Savoir sur des exemples simples et concrets : représenter un langage à l'aide d'une grammaire, proposer un algorithme pour construire l'arbre d'analyse d'un mot engendré par la grammaire (sur des cas simples seulement !).