Programme de colle 8 (6 novembre)

La démonstration des énoncés marqués d'une étoile est exigible

1 Langages réguliers

- Mots : alphabet, mot sur un alphabet, concaténation, puissance d'un mot, structure de monoïde.
- Langages : opérations ensemblistes sur les langages (union, intersection, différence, complémentaire), concaténation de deux langages, étoile de Kleene d'un langage
- Langages réguliers (aussi appelés langages rationnels): définition inductive de la classe des langages réguliers, **tout langage fini est régulier (*)**
- Expressions régulières : définition inductive, langage dénoté par une expression régulière (notation $\mu(e)$), **un langage est régulier si et seulement si il est dénoté par une expression régulière (* interroger sur un des deux sens seulement)**, expressions régulières équivalentes
- Expressions régulières étendues : aucune connaissance spécifique n'est exigible mais les étudiants doivent savoir trouver une expression régulière équivalente pour chacune des opérations introduites.

2 Automates finis

Les étudiants doivent savoir appliquer les constructions et algorithmes sur les automates.

- Automate fini déterministe (afd) : définition et calcul d'un afd, mots reconnus, langages acceptés
- Accessibilité et co-accessibilité; afd émondé
- Afd complet; complétion d'un afd
- Automate complémentaire
- Automate produit
- Classe des langages reconnaissables : stabilité par complémentaire, intersection et union finie (concaténation et étoile pas encore vu)
- Automate fini non déterministe (afnd) : définition et calcul d'un afnd, mots reconnus, langage accepté
- Automate fini non déterministe avec transitions instantanées (ε-afnd) : définition, ε-clôture, calcul d'un ε-afnd, mots reconnus, langage accepté
- Déterminisation d'un afnd ou d'un ε-afnd (automate des parties) : les classes de reconaissabilité sont les mêmes pour tous les types d'automates finis.
- Langages non reconnaissables : **lemme de l'étoile (*)**, les exemples $\{a^n b^n \mid n \in \mathbb{N}\}$ et $\{u \mid |u|_a = |u|_b\}$ ont été vus en cours.

3 Union-Find et algorithme de Kruskal

- Structure Union-Find : implémentation naïve favorisant l'opération `FIND` avec un tableau dans lequel chaque case i contient le représentant de l'élément numéro i. Complexités des opérations.
- Structure Union-Find : implémentation à l'aide d'une forêt favorisant l'opération `UNION`.

  1. Optimisation 1 : en plaçant systématiquement l'arbre de plus petite hauteur comme fils lors de l'opération union. **Dans ce cas, la hauteur de chaque arbre ne dépasse pas $\log_2(m)$ avec m la taille de l'arbre considéré (*)**
  2. Optimisation 2 : en compressant les chemins lors de l'opération `FIND`. La complexité amortie obtenue est hors programme.
- Notion d'arbre couvrant d'un graphe non orienté. Algorithme générique de construction d'un arbre couvrant : utilisation de Union-Find dans ce cadre.
- Arbre couvrant de poids minimal d'un graphe non orienté pondéré. Existence. **Algorithme de Kruskal** : savoir décrire l'algorithme en pseudo-code avec la structure Union-Find, savoir l'appliquer sur un exemple, la preuve n'est pas au programme de colle.