Programme de colle - Semaine 12 (du 23 novembre)

La colle est constituée d'une démonstration ou exercice de cours (parmi les énoncés ci-dessous marqués par une (*)) (20min max) suivi d'un ou plusieurs exercices portant sur le programme ci-dessous.

1 Langages formels. Langages réguliers

- Mots : alphabet, mot sur un alphabet, concaténation, puissance d'un mot, structure de monoïde $(\Sigma^*, .)$.
- Préfixes, suffixes, facteurs, sous-mots.
- 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 existe une expression régulière qui le dénote (*)** [NB : interroger sur un seul sens uniquement, chaque sens se démontre par induction]
- Expressions régulières équivalentes
- Expressions régulières étendues : par exemple $e?$, $e+$, $e\{3, 5\}$, etc. Aucune syntaxe n'est exigible mais il faut savoir comprendre une expression régulière étendue et la traduire en expression régulière du cours équivalente (ex : $e? \equiv (e|\varepsilon)$).

2 Automates finis

- **Automates finis déterministes (afd)**
  - Définis comme un quadruplet (Q, q_0, F, δ) avec Q ensemble des états, q_0 état initial, F états finaux, $\delta : Q \times \Sigma \to Q$ fonction (partielle) de transition.
  - Définition, notion de calcul dans un afd, notion de blocage.
  - Langage reconnu (ou accepté) par automate : calcul réussi, mots acceptés. Classe des langages reconnaissables notée $\mathrm{REC}(\Sigma^*)$.
  - Accessibilité et co-accessibilité; afd émondé
  - Afd complet; complétion d'un afd
  - **Automate complémentaire : Si L est reconnaissable alors $\bar{L}$ l'est (*)**
  - Automate produit. **Si A_1 et A_2 sont 2 afd alors $\mathcal{L}(A_1 \times A_2) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$ (*)**.

- **Automates finis non déterministes (afnd)**
  - Définis comme un quadruplet (Q, I, F, T) avec Q ensemble des états, I états initiaux, F états finaux, $T \subset Q \times \Sigma \times Q$ ensemble des transitions.
  - Calcul dans un afnd. Calcul réussi.
  - Mots acceptés par un afnd. Langage reconnu.
  - Fonction de transition $\delta : Q \times \Sigma \to \mathfrak{P}(Q)$ et fonction de transition étendue $\delta : Q \times \Sigma^* \to \mathfrak{P}(Q)$ d'un afnd :

    $$\delta(q, c) = \{q' \,/\, (q, c, q') \in T\}.$$

    La notion de blocage se traduit par $\delta(q, c) = \varnothing$.
  - Automate des parties. Algorithme de déterminisation d'un afnd.
  - Les notions suivantes restent valides : états accessibles, co-accessibles, automate émondé, complétion. **L'algorithme de l'automate complétaire ne fonctionne pas sur un automate non déterministe** (déterminiser d'abord).
  - Définition de l'automate produit dans le cas d'un automate non déterministe. L'algorithme de construction pas à pas ne change pas : on fait simplement attention à ce que la lecture d'une lettre depuis un état peut conduire à plusieurs transitions.

- **Automates finis non déterministes à transitions spontanées (ε-afnd)**
  - Définis comme un quadruplet (Q, I, F, T) avec Q ensemble des états, I états initiaux, F états finaux, $T \subset Q \times \Sigma \cup \{\varepsilon\} \times Q$ ensemble des transitions.
  - Les notions de calcul, calcul réussi, mot accepté, langage reconnu, *etc* sont les mêmes que pour un afnd.
  - ε-clôture ou ε-fermeture, notée $\eta : Q \to \mathfrak{P}(Q)$ dans le cours, définie comme l'ensemble des états qu'on peut atteindre à partir d'un état q en n'empruntant que des ε-transitions.

  - Formules récursives pour le calcul de la fonction de transition étendue : $\delta : Q \times \Sigma^* \to \mathfrak{P}(Q)$
  - Algorithme de suppression des ε-transitions pour passer d'un ε-afnd à un afnd équivalent (sans avoir à déterminiser).
  - Algorithme de déterminisation d'un ε-afnd.

- **Langages non reconnaissables** : le lemme de l'étoile et sa démonstration doivent être sus, la méthode pour montrer qu'un langage est non reconnaissable doit être sue.

  - **Lemme de l'étoile (*)** : Soit L est un langage reconnu par un automate à n états et $u \in L$ tel que $|u| \geq n$ alors il existe une décomposition $u = xyz$ telle que $|xz| \leq n$, $y \neq \varepsilon$, $\forall k \in \mathbb{N}, xy^k z \in L$.

  - Application : **$L = \{a^n b^n, n \in \mathbb{N}\}$ n'est pas reconnaissable (*)**

  - Utilisation des propriétés de fermeture des langages reconnaissables pour montrer qu'un langage n'est pas reconnaissable.

3 Théorème de Kleene

On fait le lien entre langages réguliers et langages reconnnaissables par automate :

$$\mathrm{RAT}(\Sigma^*) = \mathrm{REC}(\Sigma^*)$$

La démonstration se fait de manière pratique en donnant deux algorithmes pour passer d'une expression régulière à un automate, et un algorithme pour passer d'un automate à une expression régulière :

- **Algorithme de Berry-Séthi** (à utiliser de préférence)

  - Pour un langage L, définition de $D(L)$ (premières lettres), $F(L)$ (dernières lettres) et $T(L)$ (paires de lettres consécutives existant dans les mots de L).

  - Un langage est local lorsque

$$L \setminus \{\varepsilon\} = \left(D(L)\Sigma^* \cap \Sigma^* F(L)\right) \setminus \left(\Sigma^*(\Sigma^2 \setminus T(L))\Sigma^*\right)$$

  - Pour tout langage local on peut construire **l'automate de Glushkov** qui est un automate fini déterministe à $|\Sigma| + 1$ états qui reconnaît L.

  - Algorithme de Berry-Séthi : on linéarise l'expression régulière en ajoutant des numéros aux lettres, on détermine $D(L')$, $F(L')$, $T(L')$ et si $\varepsilon \in L'$, on construit l'automate de Glushkov associé et enfin on supprime les numéros de lettres.

  - **Savoir appliquer l'algorithme de Berry-Séthi sur un exemple (*)** : par exemple $(a|b) * c(a|ab) *$.

- **Automate de Thomson**
  - Automates normalisés : un seul état initial n'ayant que des transitions sortantes et un seul état final n'ayant que des transitions entrantes.
  - Normalisation d'un automate fini quelconque.
  - **Pour toute expression régulière e, il existe un automate normalisé qui reconnaît le langage dénoté par e (*)** : on demande une preuve par induction, les élèves doivent connaître et respecter la construction vue en cours pour chacun des cas inductifs, on ne demande pas de justifier que cette construction est correcte.
- **Algorithme d'élimination des états**
  - Automates généralisés où les transitions peuvent être étiquetées par une expression régulière. Ils ne doivent être utilisés que dans le cadre de cet algorithme.
  - Algorithme d'élimination des états : on normalise l'automate de départ puis on supprime un à un les états (sauf l'état initial et l'état final). On en déduit l'expression régulière.
  - **Utiliser l'algorithme d'élimination des états pour obtenir une expression régulière dénotant : $\{u \in \{a,b\}^* / |u|_a$ pair$\}$ (*)**

4 Apprentissage supervisé (début)

- Principe de l'apprentissage supervisé : les élèves doivent savoir l'expliquer.
- **Cas des données qualitatives** : dans le programme les attributs et les étiquettes ont des valeurs des binaires, en cours on a vu le cas plus général des étiquettes dans un ensemble fini.
  - Arbres binaires de décision (ABD) : définition et écriture de la fonction de classification utilisant un ABD.
  - Éléments de théorie de l'information : définition de l'entropie, cas du pile ou face traité en cours, calcul de l'entropie dans le cas d'un ensemble d'exemples d'apprentissage

$$H(S) = -\sum_{x \in \mathcal{X}} \frac{n_x}{n_{\text{tot}}} \log_2 \frac{n_x}{n_{\text{tot}}}$$

  - Gain d'information par rapport à un attribut, utilisation pour déterminer l'attribut le plus discriminant.

$$G(S,A) = H(S) - \frac{n_{A=\text{non}}}{n_{\text{tot}}} H(S_{A=\text{non}}) - \frac{n_{A=\text{oui}}}{n_{\text{tot}}} H(S_{A=\text{oui}})$$

– **Algorithme ID3 (*)**: savoir écrire l'algorithme en pseudo-code et l'appliquer sur un exemple. Seul le cas d'apprentissage d'arbres **binaires** de décision est au programme (attributs binaires), les élèves peuvent utiliser une calculatrice pour calculer les entropies et les gains.