Programme de colle - Semaine 17 (semaine du 26 janvier au 30 janvier)

La démonstration des énoncés marqués d'une étoile est exigible

1 Logique propositionnelle (révisions 1A)

- Syntaxe de la logique propositionnelle : les formules sont définies par induction. Symboles utilisés : $\bot, \top, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Les formules sont des **données** (arbres) sur lesquelles on travaille.
- Sémantique : les valeurs de vérité ont été notées *false* et *true*. Valuations. Valeur de vérité d'une formule F dans le contexte de la valuation φ (notée $[\![F]\!]_\varphi$). Table de vérité d'une formule.
- Tautologies. Antilogies. Formules satisfisables. Utilisation de tables de vérité dans ce cadre.
- **Une formule est une tautologie si et seulement si sa négation est non satisfiable (*)**
- Algorithme de Quine pour tester la satisfiabilité d'une formule.
- Équivalence de deux formules propositionnelles (notation $\equiv$). Calculs par équivalents. Équivalences usuelles (distributivité de $\vee$ sur $\wedge$ et réciproquement, lois de De Morgan, $(p \rightarrow q) \equiv (\neg p \vee q)$, ...).
- Formes normales conjonctives et disjonctives. Littéral. Clause. Mise sous forme normale conjonctive (resp. disjonctive) d'une formule.
- Conséquence. $\Gamma \models F$ signifie que F est la conséquence sémantique de l'ensemble de formules Γ. Exemple : $(p \rightarrow q), (q \rightarrow r) \models (p \rightarrow r)$.

2 Logique des prédicats (1er ordre)

On s'intéresse surtout à la syntaxe. La sémantique est abordée de manière intuitive.

- Syntaxe des **termes** : constantes, variables, fonctions (avec arité)
- Syntaxe des **formules** : prédicats (avec arité), propositions atomiques, $\bot$, $\top$, $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$, quantificateurs $\forall$, $\exists$
- Portée des quantificateurs. Occurrence **liée** ou **libre** d'une variable. Formules closes.
- Substitution d'une variable par un terme (on en substitue que les occurrences libres). Notion de capture.
- À savoir faire :
  - Identifier une formule de la logique des prédicats.
  - Écrire une formule du 1er ordre en respectant la syntaxe.
  - Modélisation : traduire un énoncé en langue naturelle en formule de la logique des prédicats.

3 Déduction naturelle

La déduction naturelle permet de formaliser la notion de preuve. Elle permet de manipuler à la fois les formules propositionnelles et les formules de la logique des prédicats.

- Notion de **séquent** : $\Gamma \vdash F$. Γ est un ensemble fini de formules.
- **Arbres de preuves** : un **séquent** est dérivable/prouvable s'il existe un arbre de preuve dont il est la racine.
- **Règles de déduction** vue en cours – **les règles doivent être apprises** :
  - Logique minimale : axiome, affaiblissement, règles d'introduction et d'élimination pour $\wedge, \vee, \neg, \rightarrow$.
  - Logique intuitionniste : on ajoute *ex falso sequitur quodlibet* (aussi appelée élimination du $\bot$).
  - Logique classique : on a ajouté la règle de raisonnement par l'absurde (RAA) mais d'autres règles sont équivalentes (élimination de la double négation ou tiers-exclus).
- Les étudiants doivent savoir trouver des petits arbres de preuve (on évite la technicité excessive sauf groupe (*)).
- **Correction :** de la déduction naturelle : Si le séquent $\Gamma \vdash F$ est dérivable alors $\Gamma \models F$. **Savoir démontrer la correction de n'importe quelle régle de déduction dans le cadre de la logique propositionnelle uniquement (*)**
- **Complétude :** Si $\Gamma \models F$ alors il existe un arbre de preuve pour $\Gamma \vdash F$. Ce résultat est admis.
- Extension à la logique des prédicats : règles d'élimination et d'introduction du $\exists$ et du $\forall$, quelques exemples vus en cours.

Remarque : la classification logique minimale, intuitionniste et classique n'est pas à connaître mais l'ensemble des règles de déduction l'est.

4 Décidabilité et complexité : le début

*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 $B \in \mathbf{P}$ et $A \leq_P B$ alors $A \in \mathbf{P}$ (*)

- 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.
- $\mathbf{P} \subset \mathbf{NP}$ (*)

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 (seul SAT a été fait en cours)

À venir (prochain programme) : NP-complétude, ne pas interroger sur ce point cette semaine.