Logique et Fondements de l'Informatique
Attendus du cours en 2024–2025

David A. Madore

24 janvier 2025

INF110

Git : `04d6647 Fri Jan 24 12:33:01 2025 +0100`

Sont au programme du cours les notions suivantes :

— **Calculabilité :** fonctions primitives récursives, fonctions générales récursives, numérotation (notamment la notation $\varphi_e(i)$), théorème s-m-n, astuce de Quine, existence d'une fonction universelle pour les fonctions générales récursives (et inexistence pour les primitives récursives), théorème de la forme normale et possibilité de lancer des calculs en parallèle, théorème de récursion de Kleene ; indécidabilité du problème de l'arrêt, théorème de Rice ; machines de Turing et équivalence avec les fonctions générales récursives ; parties décidables et semi-décidables, équivalence entre semi-décidable et « image d'une fonction calculable » ; la notion de réduction many-to-one et de Turing ; le λ-calcul non typé, β-réduction, théorème de Church-Rosser, redex extérieur gauche, entiers de Church, équivalence du λ-calcul avec les fonctions générales récursives, combinateur Y.

— **Typage :** λ-calcul simplement typé, et sa version enrichie par les types produits, sommes, 1 et 0 ; terminaison des programmes écrits dans ce dernier (normalisation forte ; sans preuve ni détails) ; correspondance de Curry-Howard entre λ-calcul simplement typé enrichi et calcul propositionnel intuitionniste.

— **Calcul propositionnel :** règles de logique en déduction naturelle, et au moins une présentation des preuves (arbre de preuve, ou drapeau) ; écriture et vérification des λ-termes de preuve (sans entrer dans le détail pointilleux des notations) ; différence entre logique intuitionniste et logique classique ; propriété de la disjonction et décidabilité du calcul propositionnel intuitionniste (idée).

— **Sémantiques du calcul propositionnel intuitionniste :** sémantique des ouverts, sa correction et sa complétude ; définition de la sémantique de la rélisabilité propositionnelle, et sa correction ; savoir utiliser *au moins une* des sémantiques vues en cours (ouverts, réalisabilité propositionnelle, éventuellement Kripke ou les problèmes finis) pour montrer qu'une formule propositionnelle n'est pas démontrable.

— **Quantificateurs :** règles *générales* d'introduction et d'élimination du $\forall$ et $\exists$, et λ-termes de preuve correspondants (sans entrer dans le détail pointilleux des notations) ; logique du premier ordre pure, logique du premier ordre avec égalité.

— **Arithmétique du premier ordre :** les axiomes de Peano ; l'idée générale que Curry-Howard sur l'arithmétique de Heyting permet d'extraire des algorithmes des preuves ; le fait qu'il est

possible formaliser $\varphi_e(i)\downarrow$ en arithmétique de Heyting/Peano ; le fait que vérifier si une preuve est valable est décidable, mais que savoir si un énoncé est un théorème est seulement semi-décidable ; l'énoncé du théorème de Gödel et l'idée de sa démonstration.
— **Coq :** l'utilisation générale de Coq telle que pratiquée en TP, et notamment les principales tactiques de raisonnement (sur les connecteurs logiques, types inductifs, égalités, etc.). Le nom des tactiques est exigible donc on recommande de préparer un récapitulatif pour l'examen (un tel document est disponible sur la page eCampus du cours).

Ne sont explicitement pas exigibles les notions suivantes :
— Les détails de la fonction d'Ackermann ; les détails de la notion d'arbre de calcul (autre que l'énoncé du théorème de la forme normale) ; la notion de degré many-to-one ou de Turing ; les notions de β-réduction autres qu'extérieur gauche, les subtilités de l'ordre d'évaluation, le combinateur Z ou sa différence avec Y.
— Les détails du typage de quelque langage de programmation que ce soit (autres que les variantes du λ-calcul simplement typé vus en cours, et les parties de Coq vues en TP), notamment rien de ce qui concerne Scheme, Haskell ou quelque autre langage mentionné en passant dans le cours ; le sous-typage, le polymorphisme ad hoc, les types dépendants, ou les autres fonctionnalités de certains systèmes de typages mentionnés en passant dans le cours. L'algorithme de Hindley-Milner.
— Le calcul des séquents, le fonctionnement de l'élimination des coupures ou sa preuve. Le $\overline{\lambda}$-calcul. Les axiomes de Hilbert, et les combinateurs $\mathsf{S}, \mathsf{K}, \mathsf{I}$.
— La notion de continuation. La fonction call/cc. Le continuation-passing-style ou de son typage. Le $\lambda\mu$-calcul.
— Les subtilités de la réalisabilité propositionnelle (p.ex., la réalisabilité de la formule de Tseitin). La sémantique de Kripke et celle des problèmes finis ne sont pas exigibles (mais pourront être utilisées si on le souhaite).
— Le λ-cube de Barendregt, les subtilités de la différence entre $\exists$ et types sommes, la notion de prédicativité/imprédicativité.
— Les subtilités des différences et rapports entre l'arithmétique de Heyting et celle de Peano (sauf s'il s'agit, par exemple, de vérifier si une démonstration donnée utilise un raisonnement par l'absurde).
— Les détails de la démonstration du théorème de Gödel, la formalisation des systèmes précis auxquels il s'applique.
— Les détails de la syntaxe de Coq.