

SESSION 2024

AGREGATION CONCOURS EXTERNE

Section : INFORMATIQUE

ÉPREUVE SPÉCIFIQUE SELON L'OPTION CHOISIE :

- **ÉTUDE DE CAS INFORMATIQUE**
- **FONDEMENTS DE L'INFORMATIQUE**

Durée : 6 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique (y compris la calculatrice) est rigoureusement interdit.

Il appartient au candidat de vérifier qu'il a reçu un sujet complet et correspondant à l'épreuve à laquelle il se présente.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier. Le fait de rendre une copie blanche est éliminatoire.

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

► Etude de cas informatique :

Concours	Section/option	Epreuve	Matière
EAE	6200A	103	9424

► Fondement de l'informatique :

Concours	Section/option	Epreuve	Matière
EAE	6200A	103	9425

Fondements de l'informatique

Préliminaires généraux : Ce sujet porte sur le thème de la logique et s'intéresse plus particulièrement à la possibilité d'automatiser la recherche de preuves de formules logiques. Les travaux de Gentzen sur les systèmes formels de déduction se sont révélés fondamentaux en théorie de la démonstration. Nous nous intéressons ici à leur côté pratique pour l'automatisation du raisonnement. En particulier, l'aspect syntaxique des règles de déduction est un point positif quant à l'objectif d'un usage de la machine pour mener à bien les raisonnements.

Ce sujet se décompose en trois parties. La partie I justifie l'usage de systèmes déductifs formels via la démonstration du théorème de complétude de la déduction naturelle dans le cadre de la logique propositionnelle classique. La partie II porte sur l'implémentation d'un algorithme de recherche de preuve, toujours en logique propositionnelle classique, mais avec un système déductif plus adapté à la mécanisation que la déduction naturelle : le calcul des séquents. Enfin, la partie III porte sur le passage à la logique du premier ordre et l'usage de la méthode dite des tableaux pour la recherche de preuve. Les trois parties sont indépendantes, mais il est nécessaire de connaître le système du calcul des séquents, introduit en partie II, pour traiter la partie III.

Les figures et types OCaml de cet énoncé sont reproduits dans une annexe en fin de sujet afin de faciliter la composition.

Attendus : Les questions de programmation doivent être traitées en langage OCaml. On pourra utiliser toutes les fonctions du module `List`, ainsi que les fonctions de la bibliothèque standard (celles qui s'écrivent sans nom de module, comme `max`, `incr` ainsi que les opérateurs comme `@`). L'utilisation d'autres modules est interdite.

Dans l'écriture d'une fonction, on pourra faire appel à des fonctions définies dans les questions précédentes, même si ces dernières n'ont pas été traitées. On pourra également définir des fonctions auxiliaires, mais il faudra alors préciser leur rôle ainsi que le type et la signification de leurs arguments. Si les paramètres d'une fonction à coder sont supposés vérifier certaines hypothèses, il ne sera pas utile dans l'écriture de cette fonction de tester si les hypothèses sont bien satisfaites. Lorsque les choix d'implémentation ne découlent pas directement des spécifications de l'énoncé, il est conseillé de les expliquer.

On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple n) et en police à chasse fixe du point de vue informatique (par exemple `n`).

Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle dans le pire des cas. La complexité sera exprimée sous la forme $\mathcal{O}(f(n))$ où n est la taille de l'argument de l'algorithme, et f une expression la plus simple possible. Les calculs de complexité seront justifiés succinctement.

Il est attendu des candidates et des candidats des réponses construites. Leur évaluation portera aussi sur la précision, le soin et la clarté de la rédaction.

Partie I. Théorème de complétude

Dans cette partie, nous nous intéressons aux formules du calcul propositionnel. Nous supposons fixé un ensemble $\mathcal{P}$ dénombrable de variables propositionnelles. Nous considérons l'ensemble $\mathcal{F}_0$ des formules défini inductivement à partir des variables propositionnelles et à l'aide des connecteurs $\neg$ (négation), $\vee$ (disjonction), $\wedge$ (conjonction) et $\rightarrow$ (implication). Nous incluons une formule antilogique $\perp$ dans cette définition inductive, qui est rappelée en Figure 1a. Nous rappelons également les règles de la déduction naturelle pour le calcul propositionnel en logique classique en Figure 1b.

$$\begin{array}{c}
 \frac{p \in \mathcal{P}}{p \in \mathcal{F}_0} \qquad \frac{}{\perp \in \mathcal{F}_0} \qquad \frac{A \in \mathcal{F}_0}{\neg A \in \mathcal{F}_0} \\
 \\
 \frac{A \in \mathcal{F}_0 \quad B \in \mathcal{F}_0}{A \wedge B \in \mathcal{F}_0} \qquad \frac{A \in \mathcal{F}_0 \quad B \in \mathcal{F}_0}{A \vee B \in \mathcal{F}_0} \qquad \frac{A \in \mathcal{F}_0 \quad B \in \mathcal{F}_0}{A \rightarrow B \in \mathcal{F}_0} \\
 \\
 \text{(a) Définition inductive des formules du calcul propositionnel} \\
 \\
 \frac{}{\Gamma, A \vdash A} ax \qquad \frac{\Gamma, \neg A \vdash \perp}{\Gamma \vdash A} RAA \qquad \frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A} \neg_i \qquad \frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \perp} \neg_e \\
 \\
 \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge_{e,g} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge_{e,d} \\
 \\
 \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee_{i,g} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee_{i,d} \qquad \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee_e \\
 \\
 \frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow_i \qquad \frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \rightarrow_e \\
 \\
 \text{(b) Règles d'inférence de la déduction naturelle classique}
 \end{array}$$

FIGURE 1 – Formules et règles de la déduction naturelle pour le calcul propositionnel classique

Pour toute famille finie de formules $(A_i)_{i \in \llbracket 0, n \rrbracket}$ et tout entier $k \in \llbracket 0, n \rrbracket$, nous nous permettrons d'écrire $\bigwedge_{i=0}^k A_i$ pour désigner la formule définie par :

$$\bigwedge_{i=0}^k A_i = \begin{cases} A_0 & \text{si } k = 0 \\ \left(\bigwedge_{i=0}^{k-1} A_i \right) \wedge A_k & \text{sinon} \end{cases} .$$

Une définition similaire s'appliquera également au connecteur $\vee$, et si Γ est un ensemble fini de formules, nous nous permettrons d'écrire $\bigvee_{A \in \Gamma} A$ pour désigner $\bigvee_{i=0}^n A_i$, où $A_0, \dots, A_n$ est une numérotation arbitraire des éléments de Γ . Par convention, nous définirons $\bigvee_{A \in \emptyset} A = \perp$.

Nous rappelons qu'une valuation est une fonction $v : \mathcal{P} \rightarrow \{V, F\}$ donnant une valeur de vérité à chaque variable propositionnelle, et nous noterons $\llbracket A \rrbracket_v$ l'évaluation de la formule A selon la valuation v . Nous noterons $\Gamma \models A$ le fait que la formule A est une conséquence sémantique de l'ensemble de formules Γ , i.e. pour toute valuation v , si $\llbracket B \rrbracket_v = V$ pour toute formule $B \in \Gamma$, alors $\llbracket A \rrbracket_v = V$. Ainsi, $\models A$ signifie que A est une tautologie. Enfin, l'équivalence sémantique entre deux formules A et B sera notée $A \equiv B$.

L'objectif de cette partie est de démontrer la complétude du système déductif présenté en Figure 1b, c'est-à-dire le théorème suivant.

Théorème 1 (Complétude). *Soit $A \in \mathcal{F}_0$ une formule du calcul propositionnel.*

Si $\models A$, alors le séquent $\vdash A$ est dérivable.

I.1 Restriction de l'ensemble des formules

Nous allons commencer par démontrer certaines équivalences classiques, ce qui nous permettra de restreindre l'ensemble des formules possibles pour simplifier les démonstrations.

Question 1. Soit une formule $A \in \mathcal{F}_0$. On cherche une dérivation du séquent $\vdash \neg\neg A \leftrightarrow A$, où le connecteur $\leftrightarrow$ est défini comme suit : $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$.

En exploitant la règle appropriée pour se ramener aux dérivations des séquents $\vdash \neg\neg A \rightarrow A$ et $\vdash A \rightarrow \neg\neg A$, proposer une telle dérivation.

Indication : la règle *RAA* servira exactement une fois, dans la dérivation du séquent $\vdash \neg\neg A \rightarrow A$.

Question 2. Nous rappelons les lois de De Morgan : pour toutes formules $A, B \in \mathcal{F}_0$, on a

2.a) $\neg(A \vee B) \equiv \neg A \wedge \neg B$;

2.b) $\neg(A \wedge B) \equiv \neg A \vee \neg B$.

Donner une dérivation du séquent $\vdash \neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$, où $A, B \in \mathcal{F}_0$.

Question 3. Étant donné deux formules $A, B \in \mathcal{F}_0$, proposer une formule équivalente à $A \rightarrow B$ n'utilisant pas l'opérateur d'implication. À la manière des questions précédentes, donner un séquent représentant cette équivalence et le démontrer.

Les deux questions qui suivent donnent des outils permettant d'exploiter les équivalences précédentes dans le contexte de démonstrations.

Question 4. Soit deux formules $A, B \in \mathcal{F}_0$ telles que $\vdash A \leftrightarrow B$ est dérivable. Montrer que $\vdash A$ est dérivable si et seulement si $\vdash B$ l'est également.

Question 5. Démontrer le théorème d'affaiblissement, i.e. : si un séquent $\Gamma \vdash A$ est dérivable, alors pour tout ensemble de formules Δ le séquent $\Gamma, \Delta \vdash A$ est dérivable.

Nous admettons alors que pour toute formule $A \in \mathcal{F}_0$ il existe une formule $[A]_1 \in \mathcal{F}_1$ qui lui est équivalente sémantiquement et telle que le séquent $\vdash A \leftrightarrow [A]_1$ est dérivable, où l'ensemble de formules $\mathcal{F}_1$ est défini inductivement comme en Figure 2.

$$\frac{p \in \mathcal{P}}{p \in \mathcal{F}_1} \quad \frac{}{\perp \in \mathcal{F}_1} \quad \frac{A \in \mathcal{F}_1}{\neg A \in \mathcal{F}_1} \quad \frac{A \in \mathcal{F}_1 \quad B \in \mathcal{F}_1}{A \wedge B \in \mathcal{F}_1}$$

FIGURE 2 – Définition inductive de $\mathcal{F}_1$

Question 6. Montrer que l'on peut déduire le théorème de complétude du théorème de complétude restreint aux formules de $\mathcal{F}_1$.

Nous allons donc nous focaliser sur la démonstration du théorème de complétude restreint aux formules de $\mathcal{F}_1$.

I.2 Démonstration du théorème de complétude

L'ensemble $\mathcal{P}$ des variables propositionnelles étant dénombrable, nous nous donnons une énumération de ces variables : $\mathcal{P} = (p_n)_{n \in \mathbb{N}}$. Nous appelons alors valuation initiale de rang $n \in \mathbb{N}$ une valuation partielle donnant une valeur de vérité uniquement aux variables p_i pour $i \in \llbracket 0, n \rrbracket$. Nous disons qu'une valuation initiale est compatible avec une formule si elle permet de l'évaluer, i.e. si elle donne une valeur de vérité à toutes les variables de la formule.

Question 7. On considère la formule $A = (p_0 \rightarrow (p_0 \rightarrow p_2)) \rightarrow p_2$. A est-elle une tautologie ? Si oui, le justifier. Sinon, proposer une valuation initiale compatible avec A , qui réfute A , et de rang minimal.

Question 8. Montrer qu'une formule $A \in \mathcal{F}_1$ est une tautologie si et seulement s'il existe un $n \in \mathbb{N}$ tel que toute valuation initiale v de rang n est compatible avec A et telle que $\llbracket A \rrbracket_v = V$.

Nous voulons exploiter les valuations initiales afin de montrer que toute tautologie est démontrable. Soit v une valuation initiale de rang $n \in \mathbb{N}$. Nous définissons une formule représentant v par $h_v = \bigwedge_{i=0}^n l(v, i)$, où pour tout $i \in \llbracket 0, n \rrbracket$ le littéral $l(v, i)$ est défini par :

$$l(v, i) = \begin{cases} p_i & \text{si } v(p_i) = V \\ \neg p_i & \text{sinon} \end{cases}.$$

Question 9. Soit $A_0, \dots, A_n \in \mathcal{F}_1$ et $i \in \llbracket 0, n \rrbracket$. Montrer que $\bigwedge_{j=0}^n A_j \vdash A_i$ est dérivable.

Question 10. Soit $A \in \mathcal{F}_1$ et v une valuation initiale compatible avec A , dont on note n le rang. Montrer que

- si $\llbracket A \rrbracket_v = V$, alors le séquent $h_v \vdash A$ est dérivable ;
- si $\llbracket A \rrbracket_v = F$, alors le séquent $h_v \vdash \neg A$ est dérivable.

Nous voulons maintenant exploiter le résultat de la question 10 en démontrant par récurrence la propriété P_n : si pour toute valuation initiale v de rang n le séquent $h_v \vdash A$ est dérivable, alors $\vdash A$ est dérivable.

Question 11. Démontrer P_0 .

Question 12. Soit $n \in \mathbb{N}$ tel que P_n est vraie. Démontrer P_{n+1} .

Question 13. Démontrer le théorème de complétude pour les formules de $\mathcal{F}_1$.

Partie II. Mécanisation de la recherche de preuve

Le théorème de complétude étant démontré, nous voulons maintenant confier la recherche de preuve à un ordinateur. Cependant, le système déductif présenté en Figure 1b présente plusieurs freins à cet usage :

- certaines règles, comme $\vee_e$ et $\rightarrow_e$, nécessitent de trouver une formule qui n'apparaît pas nécessairement dans le contexte initial. Trouver une telle formule de manière automatique ne paraît pas aisé.
- il est possible de « se tromper » en construisant un arbre de preuve, par exemple si l'on applique la règle $\vee_{i,g}$ alors que c'était le membre droit de la disjonction qui était démontrable.

Nous allons commencer par formaliser cette notion d'erreur possible dans l'application des règles. Un séquent $\Gamma \vdash A$ est dit valide si et seulement si $\Gamma \models A$. Une règle d'inférence est dite inversible si, dès lors que sa conclusion est un séquent valide, ses prémisses le sont aussi. Une règle non inversible est donc une règle pour laquelle il est possible lors de la recherche de preuve d'obtenir des prémisses non démontrables (car non valides) à partir d'une conclusion démontrable (car valide).

Question 14. Donner, en justifiant à l'aide d'exemples, les règles non inversibles de la déduction naturelle. On ne justifiera pas que les autres règles sont inversibles.

Pour résoudre ces problèmes techniques de la déduction naturelle, Gentzen a introduit un système déductif appelé calcul des séquents et qui s'appuie sur la notion de séquent symétrique. Un séquent symétrique est un séquent dont les deux membres sont des ensembles de formules. Nous écrirons donc un séquent symétrique sous la forme $\Gamma \vdash \Delta$ où Γ et Δ sont tous deux des ensembles de formules, contrairement aux séquents asymétriques, de la forme $\Gamma \vdash A$ avec A une formule unique. Le séquent symétrique $\Gamma \vdash \Delta$ doit être interprété de la manière suivante : de la conjonction des formules de Γ , on peut déduire la disjonction des formules de Δ . Les règles du calcul des séquents classique sont présentées en Figure 3.

$$\begin{array}{c}
\frac{}{\Gamma, A \vdash A, \Delta}^{ax} \qquad \frac{}{\Gamma, \perp \vdash \Delta}^{\perp_g} \\
\\
\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}^{\neg_g} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}^{\neg_d} \\
\\
\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}^{\wedge_g} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}^{\wedge_d} \\
\\
\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}^{\vee_g} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}^{\vee_d} \\
\\
\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \rightarrow B \vdash \Delta}^{\rightarrow_g} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}^{\rightarrow_d}
\end{array}$$

FIGURE 3 – Règles d'inférence du calcul des séquents classique

Bien que le système déductif de la Figure 3 ne contienne aucune règle telle que la règle *RAA* de la déduction naturelle pour le raisonnement par l'absurde, c'est bien un système pour la logique classique, comme le démontre la question suivante.

Question 15. Donner une dérivation du tiers exclu : $\vdash A \vee \neg A$.

Nous admettons que le calcul des séquents est équivalent à la déduction naturelle, c'est-à-dire que :

- un séquent $\Gamma \vdash A$ est dérivable en déduction naturelle si et seulement s'il l'est également en calcul des séquents ;
- un séquent $\Gamma \vdash \Delta$ est dérivable en calcul des séquents si et seulement si le séquent $\Gamma \vdash \bigvee_{A \in \Delta} A$ l'est également en déduction naturelle.

Afin d'observer plus concrètement cette équivalence, et de constater le côté mécanique de la recherche de preuve en calcul des séquents, nous pouvons redémontrer des séquents démontrés précédemment en déduction naturelle.

Question 16. Donner des dérivations des séquents des questions 1 et 2 en calcul des séquents, à savoir :

16.a) $\vdash \neg \neg A \leftrightarrow A$

16.b) $\vdash \neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$.

Pour simplifier notre étude du calcul des séquents, nous nous limiterons aux formules de l'ensemble $\mathcal{F}_1$ défini en Figure 2, ce qui nous permettra de nous passer des règles pour la disjonction et l'implication ($\vee_g$, $\vee_d$, $\rightarrow_g$ et $\rightarrow_d$).

Nous étendons la notion de séquent valide aux séquents symétriques de la manière suivante : $\Gamma \vdash \Delta$ est valide si et seulement si $\Gamma \models \bigvee_{A \in \Delta} A$, ce que nous noterons également $\Gamma \models \Delta$. Nous

admettons que le calcul des séquents est un système déductif correct et complet, i.e. qu'un séquent $\Gamma \vdash \Delta$ est dérivable si et seulement s'il est valide.

Nous pouvons maintenant démontrer qu'il n'est plus possible de se tromper lors de la recherche de preuve.

Question 17. Démontrer que les règles du calcul des séquents sont inversibles.

Nous souhaitons maintenant implémenter un algorithme de recherche de preuve en calcul des séquents. Pour ce faire, nous commençons par nous donner un type pour les formules.

```
type 'a formule =  
  | Faux  
  | Var of 'a  
  | Non of 'a formule  
  | Et of 'a formule * 'a formule
```

Nous programmons tout d'abord une fonction utilitaire.

Question 18. Écrire une fonction `est_atomique : 'a formule -> bool` qui détermine si une formule est atomique, i.e. si elle ne contient aucun connecteur $\neg$ ou $\wedge$.

Nous nous donnons ensuite un type représentant les séquents : nous utilisons un enregistrement contenant une liste `hyp` de formules qui constituent les hypothèses du séquent et une liste `concl` de formules qui constituent sa conclusion.

```
type 'a sequent =  
  {hyp : 'a formule list; concl : 'a formule list}
```

Nous rappelons que la construction d'un élément de ce type dont les champs valent l_1 et l_2 se fait à l'aide de la syntaxe `{hyp =  $l_1$ ; concl =  $l_2$ }` et que si `s` est un élément de ce type, on peut lire ses champs à l'aide des expressions `s.hyp` et `s.concl`.

Nous supposons disposer des deux fonctions utilitaires suivantes sur les listes :

- une fonction `extraire : 'a -> 'a list -> 'a list` telle que `extraire x l` supprime la première occurrence de `x` dans `l` s'il en existe une et lève l'exception `Not_found` sinon ;
- une fonction `element_commun : 'a list -> 'a list -> 'a option` qui renvoie un élément commun aux deux listes passées en argument, sous la forme `Some x`, s'il en existe un, et `None` sinon.

Nous appelons formule principale d'une règle d'inférence la formule située dans la conclusion de cette règle et qui en permet l'instanciation. Le tableau suivant indique pour chaque règle de la figure 3 quelle est sa formule principale.

Règle	Formule principale
ax	A (à gauche et à droite de $\vdash$)
$\perp_g$	$\perp$
$\neg_g, \neg_d$	$\neg A$
$\wedge_g, \wedge_d$	$A \wedge B$
$\vee_g, \vee_d$	$A \vee B$
$\rightarrow_g, \rightarrow_d$	$A \rightarrow B$

Nous représentons enfin une dérivation de la manière suivante : nous utilisons un type somme dont chaque constructeur correspond à une règle d'inférence. Chaque constructeur est paramétré par la formule principale de la règle d'inférence, ainsi que par des dérivations de ses prémisses. Nous faisons une exception pour le constructeur **Faux_g**, car l'unique formule permettant d'instancier la règle $\perp_g$ est $\perp$.

```

type 'a derivation =
| Axiome of 'a formule
| Faux_g
| Non_g of 'a formule * 'a derivation
| Non_d of 'a formule * 'a derivation
| Et_g of 'a formule * 'a derivation
| Et_d of 'a formule * 'a derivation * 'a derivation

```

Par exemple, l'expression **Faux_g** représente n'importe quelle dérivation de la forme

$$\frac{}{\Gamma, \perp \vdash \Delta} \perp_g$$

où Γ et Δ sont des ensembles de formules quelconques, tandis que l'expression

```

Et_d (Et (Var "p", Var "q"),
      Axiome (Var "p"),
      Non_d (Non (Var "q"), Axiome (Var "q")))

```

représente les dérivations de la forme

$$\frac{\frac{}{\Gamma, p \vdash p, \neg q, \Delta} ax \quad \frac{\frac{}{\Gamma, p, q \vdash q, \Delta} ax}{\Gamma, p \vdash q, \neg q, \Delta} \neg_d}{\Gamma, p \vdash p \wedge q, \neg q, \Delta} \wedge_d$$

Une expression du type **'a derivation** peut représenter plusieurs dérivations différentes, puisqu'il peut exister dans le séquent dérivé des formules dont une telle expression ne fait pas mention (celles des ensembles Γ et Δ dans les exemples ci-avant). Nous nous proposons alors de vérifier si un séquent donné admet bien une dérivation représentée par une telle expression.

Question 19. Écrire une fonction `est_valide : 'a derivation -> 'a sequent -> bool` qui détermine si une dérivation est un arbre de preuve valide pour un séquent donné. On veillera à écrire une fonction de complexité $\mathcal{O}(nm)$, où n est la taille de la dérivation et m la taille du séquent.

Nous proposons enfin l'algorithme récursif suivant, qui construit une dérivation du séquent $\Gamma \vdash \Delta$ passé en argument s'il en existe une et provoque une erreur sinon :

1. S'il existe dans Γ une formule A non atomique, appliquer la règle correspondant au connecteur logique principal de A , puis construire récursivement une dérivation des prémisses de cette règle.
2. Sinon, s'il existe dans Δ une formule non atomique, procéder de même en appliquant la règle correspondante et en démontrant récursivement les prémisses de cette règle.
3. Sinon, appliquer l'une des deux règles $\perp_g$ et ax si possible et échouer sinon.

Question 20. Écrire une fonction `prouve : 'a sequent -> 'a derivation` qui implémente cet algorithme de recherche de preuve. L'échec de l'algorithme sera signifié par la levée d'une exception.

Question 21. Démontrer la terminaison de cet algorithme.

Question 22. Démontrer la correction de cet algorithme, c'est-à-dire qu'il parvient bien à construire une dérivation du séquent $\Gamma \vdash \Delta$ en entrée s'il est dérivable et qu'il échoue sinon.

Question 23. Donner un majorant de la complexité en termes de nombre d'appels récursifs de cet algorithme et montrer que cette borne est atteinte.

Partie III. Passage à la logique du premier ordre

Nous souhaitons maintenant étudier la mécanisation de la recherche de preuve en logique du premier ordre. Nous fixons donc un langage du premier ordre $\mathcal{L} = (\mathcal{S}_F, \mathcal{S}_R)$, où $\mathcal{S}_F$ est l'ensemble des symboles de fonction et $\mathcal{S}_R$ est l'ensemble des symboles de relation. Nous rappelons qu'un symbole de fonction d'arité 0 est parfois appelé constante et qu'un symbole de relation d'arité 0 est parfois appelé constante propositionnelle. Nous fixons un ensemble dénombrable $\mathcal{V}$ de variables, puis nous définissons inductivement l'ensemble $\mathcal{T}$ des termes en Figure 4a, et l'ensemble $\mathcal{F}$ des formules de la logique du premier ordre en Figure 4b.

Dans la suite, nous noterons $FV(t)$ l'ensemble des variables (libres) d'un terme $t \in \mathcal{T}$, notation que nous étendrons aux formules ($FV(A)$) et aux ensembles de formules ($FV(\Gamma)$). De même, nous noterons $t[\sigma]$ le résultat de l'application de la substitution σ aux variables de t et nous étendrons également cette notation aux formules et aux ensembles de formules. Nous noterons $x := t$ la substitution σ telle que $\sigma(x) = t$ et $\sigma(y) = y$ pour tout $y \neq x$.

Nous rappelons que, dans le calcul de $A[x := t]$, si une variable de t apparaît liée dans A , alors il est nécessaire de renommer cette variable liée avant d'appliquer la substitution afin d'éviter le phénomène de capture de variable. De manière générale, deux formules égales au renommage

$$\frac{x \in \mathcal{V}}{x \in \mathcal{T}} \qquad \frac{f \in \mathcal{S}_F \quad f \text{ d'arité } n \quad t_1, \dots, t_n \in \mathcal{T}}{f(t_1, \dots, t_n) \in \mathcal{T}}$$

(a) Définition inductive de l'ensemble des termes

$$\begin{array}{c} \frac{R \in \mathcal{S}_R \quad R \text{ d'arité } n \quad t_1, \dots, t_n \in \mathcal{T}}{R(t_1, \dots, t_n) \in \mathcal{F}} \qquad \frac{}{\perp \in \mathcal{F}} \qquad \frac{A \in \mathcal{F}}{\neg A \in \mathcal{F}} \\[10pt] \frac{A \in \mathcal{F} \quad B \in \mathcal{F}}{A \wedge B \in \mathcal{F}} \qquad \frac{A \in \mathcal{F} \quad B \in \mathcal{F}}{A \vee B \in \mathcal{F}} \qquad \frac{A \in \mathcal{F} \quad B \in \mathcal{F}}{A \rightarrow B \in \mathcal{F}} \\[10pt] \frac{A \in \mathcal{F} \quad x \in \mathcal{V}}{\forall x A \in \mathcal{F}} \qquad \frac{A \in \mathcal{F} \quad x \in \mathcal{V}}{\exists x A \in \mathcal{F}} \end{array}$$

(b) Définition inductive des formules de la logique du premier ordre

FIGURE 4 – Termes et formules de la logique du premier ordre

près de leurs variables liées seront considérées identiques. Par exemple, si $P \in \mathcal{S}_R$ est d'arité 2, alors le calcul de $(\forall x P(x, y)) [y := x]$ peut passer par le renommage de x en z , ce qui donne

$$(\forall x P(x, y)) [y := x] = (\forall z P(z, y)) [y := x] = \forall z P(z, x).$$

III.1 Calcul des séquents pour la logique du premier ordre

Nous ajoutons les règles de la figure 5 aux règles de la figure 3 afin de définir le système déductif du calcul des séquents pour la logique du premier ordre.

$$\begin{array}{c} \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \text{contr}_g \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta} \text{contr}_d \\[10pt] \frac{\Gamma, A[x := t] \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g \qquad \frac{\Gamma \vdash A, \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma \vdash \forall x A, \Delta} \forall_d \\[10pt] \frac{\Gamma, A \vdash \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma, \exists x A \vdash \Delta} \exists_g \qquad \frac{\Gamma \vdash A[x := t], \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d \end{array}$$

FIGURE 5 – Règles supplémentaires du calcul des séquents pour la logique du premier ordre

Question 24. Proposer une dérivation du séquent $\vdash \exists x \forall y (R(y) \rightarrow R(x))$, où $R \in \mathcal{S}_R$ est d'arité 1. On pourra utiliser une règle de contraction contr_d .

Question 25. Justifier qu'il est impossible de dériver le séquent $\vdash \exists x \forall y (R(y) \rightarrow R(x))$ sans utiliser de règle de contraction.

En fait, les contractions ne sont utiles que pour l'usage des règles des quantificateurs et nous admettrons qu'il est possible de remplacer les quatre règles $\forall_g$, $\exists_d$, $contr_g$ et $contr_d$ par les deux règles suivantes :

$$\frac{\Gamma, A[x := t], \{\forall x A\} \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g^c \qquad \frac{\Gamma \vdash A[x := t], \{\exists x A\}, \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d^c$$

Dans les règles ci-avant, une formule est mise entre accolades si l'on peut choisir de ne pas la conserver dans une instance de la règle. Par exemple, la règle $\forall_g^c$ admet les deux instances suivantes :

$$\frac{\Gamma, A[x := t] \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \qquad \frac{\Gamma, A[x := t], \forall x A \vdash \Delta}{\Gamma, \forall x A \vdash \Delta}$$

Ainsi, dans toute la suite, un séquent sera dit dérivable en calcul des séquents s'il est dérivable dans le système déductif constitué des règles de la Figure 3 et de la Figure 6.

$$\begin{array}{cc} \frac{\Gamma, A[x := t], \{\forall x A\} \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g^c & \frac{\Gamma \vdash A, \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma \vdash \forall x A, \Delta} \forall_d \\ \frac{\Gamma, A \vdash \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma, \exists x A \vdash \Delta} \exists_g & \frac{\Gamma \vdash A[x := t], \{\exists x A\}, \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d^c \end{array}$$

FIGURE 6 – Nouvelles règles pour les quantificateurs

Le système obtenu n'est cependant pas très pratique dans le cadre de l'automatisation de la recherche de preuve pour deux raisons : les nouvelles règle nécessitent de décider si l'on conserve la formule avec son quantificateur via une contraction, et il est nécessaire de choisir un terme pour la substitution. Le principe de la méthode des tableaux est de retarder le choix de ce terme jusqu'à l'usage des axiomes pour clore les branches de l'arbre de dérivation, afin d'exploiter les informations supplémentaires obtenues au cours de la dérivation. L'idée est de choisir la substitution de sorte à égaliser des formules à gauche et à droite des séquents afin d'obtenir des instances de la règle ax . Sa réalisation repose sur un outil appelé unification.

III.2 Unification

Pour simplifier, nous limiterons notre étude dans cette section à l'unification de termes (la généralisation aux formules est assez directe). Le principe de l'unification est alors d'égaliser des termes à l'aide d'une substitution. Nous disons que deux termes $t, u \in \mathcal{T}$ sont unifiables si et seulement s'il existe une substitution σ telle que $t[\sigma] = u[\sigma]$. Une telle substitution est appelée

unificateur pour t et u . Plus généralement, puisque nous voulons égaliser des formules atomiques pouvant contenir plusieurs termes, nous nous intéresserons au problème de l'unification simultanée de plusieurs couples de termes. Nous considérons alors le problème de décision suivant, que nous appelons UNIF : étant donné une liste $L = [t_i \stackrel{?}{=} u_i, i \in \llbracket 1, n \rrbracket]$ d'équations entre termes, existe-t-il une substitution σ telle que pour tout $i \in \llbracket 1, n \rrbracket$, $t_i[\sigma] = u_i[\sigma]$? Nous dirons encore qu'une telle substitution est un unificateur pour L .

Question 26.

26.a) Montrer que les termes $t = f(x, y)$ et $u = f(f(y, z), a)$, où $a \in \mathcal{S}_F$ est une constante et $f \in \mathcal{S}_F$ est d'arité 2, sont unifiables.

26.b) Que peut-on dire de la réponse au problème UNIF si l'on considère l'instance

$$L = [f(x, y) \stackrel{?}{=} f(f(y, z), a), f(z, y) \stackrel{?}{=} f(f(y, z), a)]?$$

Nous voulons maintenant démontrer que le problème UNIF est décidable. Nous nous proposons d'étudier l'algorithme d'unification de Martelli et Montanari. À partir de la liste L d'équations, cet algorithme construit une suite L_n de listes et une suite σ_n de substitutions de la manière suivante :

- $L_0 = L$ et σ_0 est la fonction identité ;
- L_{n+1} et σ_{n+1} sont calculés comme suit :
 - Si $L_n = []$: l'algorithme se termine et σ_n est un unificateur pour L .
 - Si $L_n = (f(t_1, \dots, t_p) \stackrel{?}{=} g(u_1, \dots, u_q)) :: L'$ avec $f, g \in \mathcal{S}_F$ et $t_1, \dots, t_p, u_1, \dots, u_q \in \mathcal{T}$:
 - si $f \neq g$, alors l'algorithme échoue : il n'y a pas d'unificateur pour L ;
 - si $f = g$ (et donc $p = q$), alors $L_{n+1} = [t_1 \stackrel{?}{=} u_1, \dots, t_p \stackrel{?}{=} u_p] @ L'$ et $\sigma_{n+1} = \sigma_n$.
 - Si $L_n = (x \stackrel{?}{=} x) :: L'$ avec $x \in \mathcal{V}$: $L_{n+1} = L'$ et $\sigma_{n+1} = \sigma_n$;
 - Si $L_n = (x \stackrel{?}{=} t) :: L'$ avec $x \in \mathcal{V}$ et $t \in \mathcal{T} \setminus \{x\}$, à symétrie de l'équation près :
 - si $x \in FV(t)$, alors l'algorithme échoue : il n'y a pas d'unificateur pour L ;
 - sinon, $\sigma_{n+1} = (x := t) \circ \sigma_n$ et $L_{n+1} = L'[x := t]$, où $[t_i \stackrel{?}{=} u_i, i \in \llbracket 1, n \rrbracket][x := t] = [t_i[x := t] \stackrel{?}{=} u_i[x := t], i \in \llbracket 1, n \rrbracket]$.

Question 27. Appliquer cet algorithme aux listes suivantes :

27.a) la liste L de la question **26.b)** ;

27.b) $[S(f(x, S(y))) \stackrel{?}{=} S(f(S(t), z))]$, où $S \in \mathcal{S}_F$ est d'arité 1.

Nous nous proposons d'implémenter cet algorithme avant d'en étudier les propriétés. Nous définissons le type des termes de la manière suivante :

```
type ('a, 'b) terme =
  | Var of 'a
  | App of 'b * ('a, 'b) terme list
```

Ainsi, un terme est soit une variable, représentée par un élément du type `'a`, soit l'application d'un symbole de fonction, représenté par un élément du type `'b`, à une liste de termes. Par exemple, l'expression `Var "x"` pourra représenter la variable x et le terme $f(x, y)$ pourra être représenté par `App ("f", [Var "x"; Var "y"])`.

Nous représentons alors une substitution comme une liste associant des termes à des éléments du type 'a et une instance du problème UNIF comme une liste de couple de termes à unifier.

```
type ('a, 'b) subst = ('a * ('a, 'b) terme) list
```

```
type ('a, 'b) instance = (('a, 'b) terme * ('a, 'b) terme) list
```

Nous commençons par implémenter deux fonctions utilitaires.

Question 28. Écrire des fonctions :

28.a) `est_variable : 'a -> ('a, 'b) terme -> bool` telle que `est_variable x t` calcule le booléen $x \in FV(t)$;

28.b) `substitue : 'a -> ('a, 'b) terme -> ('a, 'b) terme -> ('a, 'b) terme` telle que `substitue x u t` calcule le terme $t[x := u]$.

Ces deux fonctions devront être de complexité linéaire en la taille du terme t passé en paramètre.

Nous pouvons alors implémenter l'algorithme d'unification.

Question 29. Écrire une fonction `unifie : ('a, 'b) instance -> ('a, 'b) subst` qui implémente l'algorithme de Martelli et Montanari. Cette fonction lèvera une exception dans les cas d'échec de l'algorithme.

Question 30. Montrer que l'algorithme d'unification termine toujours, soit en signalant l'échec de l'unification, soit avec $L_n = []$.

Question 31. Soit $(x_n)_{n \in \mathbb{N}} \in \mathcal{V}^{\mathbb{N}}$ une suite de variables deux à deux distinctes et $(t_n)_{n \in \mathbb{N}}, (u_n)_{n \in \mathbb{N}} \in \mathcal{T}^{\mathbb{N}}$ les suites de termes définies par :

- $t_0 = u_0 = x_0$;
- pour tout $n \in \mathbb{N}$, $t_{n+1} = f(t_n, x_n)$ et $u_{n+1} = f(x_n, u_n)$.

Montrer que l'algorithme d'unification appliqué à l'équation $t_n \stackrel{?}{=} u_n$ produit une substitution σ telle que la taille de $t_n[\sigma]$ soit en $\Theta(2^n)$. En déduire une instance telle que l'algorithme d'unification est de complexité exponentielle en la taille de son entrée.

Question 32. Démontrer que pour tout $n \in \mathbb{N}$ tel que L_n est définie et pour toute substitution σ , σ est un unificateur pour L si et seulement s'il existe une substitution σ' qui est un unificateur pour L_n et telle que $\sigma = \sigma' \circ \sigma_n$.

Question 33. Conclure quant à la correction de l'algorithme d'unification.

III.3 Méthode des tableaux

Nous revenons maintenant à la recherche de preuve en calcul des séquents. Comme énoncé précédemment, le principe de la méthode des tableaux est de retarder l'instanciation des variables dans les règles $\forall_g^c$ et $\exists_d^c$ afin d'appliquer un algorithme d'unification à l'ensemble des feuilles de l'arbre de dérivation pour vérifier s'il est possible d'utiliser la règle ax au niveau de ces feuilles.

Cela nécessite toutefois d'être prudent. En effet, une même formule quantifiée peut être exploitée plusieurs fois via les règles $\forall_g^c$ et $\exists_d^c$, via la contraction intégrée à ces règles. Ainsi, si l'on veut pouvoir faire plusieurs instanciations avec des termes différents, il ne faut pas conserver la variable associée au quantificateur que l'on retire dans cette règle, mais la remplacer par une nouvelle variable n'apparaissant pas ailleurs dans l'arbre de dérivation. Cela revient donc à utiliser les deux règles suivantes à la place des règles $\forall_g^c$ et $\exists_d^c$:

$$\frac{\Gamma, A[x := ?x], \{\forall x A\} \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g^d \qquad \frac{\Gamma \vdash A[x := ?x], \{\exists x A\}, \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d^d$$

où $?x$ est une variable en attente d'instanciation, appelée métavariable, et qui ne doit apparaître nulle part ailleurs dans la dérivation en cours de construction.

Par exemple, si $?x$ et $?y$ sont deux noms différents, il sera possible d'enchaîner deux applications de la règle $\forall_g^d$ portant sur la même formule $\forall x A$ de la façon suivante :

$$\frac{\frac{\Gamma, A[x := ?x], A[x := ?y] \vdash \Delta}{\Gamma, A[x := ?x], \forall x A \vdash \Delta}}{\Gamma, \forall x A \vdash \Delta}$$

La méthode des tableaux se résume alors en ces deux étapes :

1. appliquer des règles d'inférences jusqu'à l'obtention de séquents ne contenant que des formules atomiques dans toutes les branches (nous mettons de côté le problème de la contraction dans les règles $\forall_g^d$ et $\exists_d^d$, que nous évoquerons plus tard), en fermant les branches avec la règle $\perp_g$ si possible ;
2. utiliser un algorithme d'unification pour essayer d'obtenir une substitution σ ne s'appliquant qu'aux métavariables et telle que, pour tout séquent $\Gamma \vdash \Delta$ dans les branches encore ouvertes de l'arbre obtenu à l'étape précédente, la règle ax peut être appliquée au séquent $\Gamma[\sigma] \vdash \Delta[\sigma]$. En cas de succès, le séquent initial est considéré comme démontré. Sinon, il n'est pas démontrable.

Question 34. Appliquer la méthode des tableaux au séquent $\exists y \forall x P(x, y) \vdash \forall x \exists y P(x, y)$.

Cependant, avec les règles d'inférence dont nous disposons, la méthode des tableaux peut valider un séquent qui n'est pas démontrable.

Question 35. Appliquer la méthode des tableaux au séquent $\forall x \exists y P(x, y) \vdash \exists y \forall x P(x, y)$. Pourquoi la substitution produite est-elle invalide ?

Pour résoudre ce problème, nous proposons d'utiliser une technique appelée skolémisation et dont le principe est le suivant : dans une règle $\forall_d$ ou $\exists_g$, au lieu de vérifier une contrainte supplémentaire sur la variable quantifiée, nous encodons sa possible dépendance aux autres variables libres de la formule en lui substituant un terme dépendant de ces variables, c'est-à-dire que nous remplaçons les règles $\forall_d$ et $\exists_g$ par les règles suivantes :

$$\frac{\Gamma \vdash A[x := f(?x_1, \dots, ?x_n)], \Delta}{\Gamma \vdash \forall x A, \Delta} \forall_d^s \qquad \frac{\Gamma, A[x := f(?x_1, \dots, ?x_n)] \vdash \Delta}{\Gamma, \exists x A \vdash \Delta} \exists_g^s$$

où $?x_1, \dots, ?x_n$ sont les métavariabiles de $\forall x A$ (ou $\exists x A$ dans $\exists_g^s$) et f est un symbole de fonction qui n'apparaît nulle part ailleurs dans la dérivation en cours de construction. S'il n'y a pas de métavariabiles, le terme $f(?x_1, \dots, ?x_n)$ est bien-sûr remplacé par une constante c qui répond à la même contrainte que f .

Ainsi, les règles de dérivation utilisées dans la méthode des tableaux sont les règles de la Figure 3 et celles de la Figure 7.

$$\begin{array}{cc} \frac{\Gamma, A[x := ?x], \{\forall x A\} \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g^d & \frac{\Gamma \vdash A[x := f(?x_1, \dots, ?x_n)], \Delta}{\Gamma \vdash \forall x A, \Delta} \forall_d^s \\ \frac{\Gamma, A[x := f(?x_1, \dots, ?x_n)] \vdash \Delta}{\Gamma, \exists x A \vdash \Delta} \exists_g^s & \frac{\Gamma \vdash A[x := ?x], \{\exists x A\}, \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d^d \end{array}$$

FIGURE 7 – Règles des quantificateurs pour la méthode des tableaux

Question 36. Reprendre les questions 34 et 35 avec ces nouvelles règles. Le problème de la question 35 est-il résolu ?

Question 37. En supposant que l'on veuille utiliser l'algorithme de Martelli et Montanari pour l'unification, expliquer comment réaliser la seconde étape de la méthode des tableaux. Donner l'ordre de grandeur du nombre d'appels à l'algorithme de Martelli et Montanari en fonction de la taille du séquent initial, dans le pire cas si l'on n'utilise pas la contraction dans les règles $\forall_g^d$ et $\exists_d^d$.

Nous voulons maintenant démontrer la complétude de la méthode des tableaux : si un séquent est prouvable en calcul des séquents, alors la méthode des tableaux appliquée à ce séquent réussit. Nous nous limiterons pour cela, suivant le même principe qu'en section 1, à des formules n'utilisant pas les connecteurs $\vee$, $\rightarrow$ et $\exists$.

Question 38. Montrer que si un séquent $\Gamma \vdash \Delta$ admet une dérivation, alors pour toute substitution σ , le séquent $\Gamma[\sigma] \vdash \Delta[\sigma]$ est dérivable avec une dérivation de même taille.

Question 39. Soit $\Gamma \vdash \Delta$ un séquent et σ une substitution telle que $\Gamma[\sigma] \vdash \Delta[\sigma]$ est dérivable en calcul des séquents. Montrer que pour tout ensemble fini E de formules, il existe une exécution réussie de la méthode des tableaux appliquée à $\Gamma \vdash \Delta$ et que parmi toutes les substitutions possibles pour l'unification dans les feuilles de l'arbre construit lors de cette exécution, il en existe une de la forme $\sigma \circ \tau$, où τ est une substitution laissant invariantes les formules de E .

Question 40. Conclure quant à la complétude de la méthode des tableaux.

Pour résoudre le problème de l'usage ou non de la contraction dans les règles $\forall_g^d$ et $\exists_d^d$, une technique classique consiste à décorer les quantificateurs avec des entiers strictement positifs : un quantificateur Q^n , où $Q \in \{\forall, \exists\}$ signifie que les $n-1$ premiers usages de la règle associée à ce quantificateur se feront avec contraction et que le suivant se fera sans contraction. Par exemple, la règle $\forall_g^d$ est remplacée par les deux règles suivantes :

$$\frac{\Gamma, A[x := ?x], \forall^{n-1}x A \vdash \Delta \quad n > 1}{\Gamma, \forall^n x A \vdash \Delta} \qquad \frac{\Gamma, A[x := ?x] \vdash \Delta}{\Gamma, \forall^1 x A \vdash \Delta}$$

Une idée simple pour la recherche de preuve avec la méthode des tableaux consiste alors à faire une recherche avec tous les quantificateurs décorés par l'entier 1 et, en cas d'échec, à recommencer en incrémentant les décorations. Il est évidemment possible d'utiliser des approches plus subtiles afin de gagner en efficacité.

* *

*

Annexe : formules et règles de la déduction naturelle pour le calcul propositionnel classique

$$\begin{array}{ccc}
 \frac{p \in \mathcal{P}}{p \in \mathcal{F}_0} & \frac{}{\perp \in \mathcal{F}_0} & \frac{A \in \mathcal{F}_0}{\neg A \in \mathcal{F}_0} \\
 \\
 \frac{A \in \mathcal{F}_0 \quad B \in \mathcal{F}_0}{A \wedge B \in \mathcal{F}_0} & \frac{A \in \mathcal{F}_0 \quad B \in \mathcal{F}_0}{A \vee B \in \mathcal{F}_0} & \frac{A \in \mathcal{F}_0 \quad B \in \mathcal{F}_0}{A \rightarrow B \in \mathcal{F}_0}
 \end{array}$$

(a) Définition inductive des formules du calcul propositionnel

$$\begin{array}{ccc}
 \frac{}{\Gamma, A \vdash A} ax & \frac{\Gamma, \neg A \vdash \perp}{\Gamma \vdash A} RAA & \frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A} \neg_i \quad \frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \perp} \neg_e \\
 \\
 \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i & \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge_{e,g} & \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge_{e,d} \\
 \\
 \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee_{i,g} & \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee_{i,d} & \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee_e \\
 \\
 \frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow_i & \frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \rightarrow_e
 \end{array}$$

(b) Règles d'inférence de la déduction naturelle classique

FIGURE 1 – Formules et règles de la déduction naturelle pour le calcul propositionnel classique

$$\begin{array}{ccc}
 \frac{p \in \mathcal{P}}{p \in \mathcal{F}_1} & \frac{}{\perp \in \mathcal{F}_1} & \frac{A \in \mathcal{F}_1}{\neg A \in \mathcal{F}_1} \quad \frac{A \in \mathcal{F}_1 \quad B \in \mathcal{F}_1}{A \wedge B \in \mathcal{F}_1}
 \end{array}$$

FIGURE 2 – Définition inductive de $\mathcal{F}_1$

Annexe : règles du calcul des séquent pour le calcul propositionnel classique et types OCaml associés

$$\begin{array}{c}
\frac{}{\Gamma, A \vdash A, \Delta}^{ax} \qquad \frac{}{\Gamma, \perp \vdash \Delta}^{\perp_g} \\
\\
\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}^{\neg_g} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}^{\neg_d} \\
\\
\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}^{\wedge_g} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}^{\wedge_d} \\
\\
\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}^{\vee_g} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}^{\vee_d} \\
\\
\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \rightarrow B \vdash \Delta}^{\rightarrow_g} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta}^{\rightarrow_d}
\end{array}$$

FIGURE 3 – Règles d'inférence du calcul des séquents classique

```

type 'a formule =
| Faux
| Var of 'a
| Non of 'a formule
| Et of 'a formule * 'a formule

type 'a sequent =
{hyp : 'a formule list; concl : 'a formule list}

type 'a derivation =
| Axiome of 'a formule
| Faux_g
| Non_g of 'a formule * 'a derivation
| Non_d of 'a formule * 'a derivation
| Et_g of 'a formule * 'a derivation
| Et_d of 'a formule * 'a derivation * 'a derivation

```

Annexe : termes, formules et règles du calcul des séquents pour la logique du premier ordre

$$\frac{x \in \mathcal{V}}{x \in \mathcal{T}} \qquad \frac{f \in \mathcal{S}_F \quad f \text{ d'arité } n \quad t_1, \dots, t_n \in \mathcal{T}}{f(t_1, \dots, t_n) \in \mathcal{T}}$$

(a) Définition inductive de l'ensemble des termes

$$\begin{array}{c} \frac{R \in \mathcal{S}_R \quad R \text{ d'arité } n \quad t_1, \dots, t_n \in \mathcal{T}}{R(t_1, \dots, t_n) \in \mathcal{F}} \qquad \frac{}{\perp \in \mathcal{F}} \qquad \frac{A \in \mathcal{F}}{\neg A \in \mathcal{F}} \\[10pt] \frac{A \in \mathcal{F} \quad B \in \mathcal{F}}{A \wedge B \in \mathcal{F}} \qquad \frac{A \in \mathcal{F} \quad B \in \mathcal{F}}{A \vee B \in \mathcal{F}} \qquad \frac{A \in \mathcal{F} \quad B \in \mathcal{F}}{A \rightarrow B \in \mathcal{F}} \\[10pt] \frac{A \in \mathcal{F} \quad x \in \mathcal{V}}{\forall x A \in \mathcal{F}} \qquad \frac{A \in \mathcal{F} \quad x \in \mathcal{V}}{\exists x A \in \mathcal{F}} \end{array}$$

(b) Définition inductive des formules de la logique du premier ordre

FIGURE 4 – Termes et formules de la logique du premier ordre

$$\begin{array}{c} \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \text{contr}_g \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta} \text{contr}_d \\[10pt] \frac{\Gamma, A[x := t] \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g \qquad \frac{\Gamma \vdash A, \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma \vdash \forall x A, \Delta} \forall_d \\[10pt] \frac{\Gamma, A \vdash \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma, \exists x A \vdash \Delta} \exists_g \qquad \frac{\Gamma \vdash A[x := t], \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d \end{array}$$

FIGURE 5 – Règles supplémentaires du calcul des séquents pour la logique du premier ordre

Annexe : variantes pour les règles du calcul des séquents relatives aux quantificateurs

$$\begin{array}{cc}
\frac{\Gamma, A[x := t], \{\forall x A\} \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g^c & \frac{\Gamma \vdash A, \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma \vdash \forall x A, \Delta} \forall_d \\
\frac{\Gamma, A \vdash \Delta \quad x \notin FV(\Gamma, \Delta)}{\Gamma, \exists x A \vdash \Delta} \exists_g & \frac{\Gamma \vdash A[x := t], \{\exists x A\}, \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d^c
\end{array}$$

FIGURE 6 – Nouvelles règles pour les quantificateurs

$$\begin{array}{cc}
\frac{\Gamma, A[x := ?x], \{\forall x A\} \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \forall_g^d & \frac{\Gamma \vdash A[x := f(?x_1, \dots, ?x_n)], \Delta}{\Gamma \vdash \forall x A, \Delta} \forall_d^s \\
\frac{\Gamma, A[x := f(?x_1, \dots, ?x_n)] \vdash \Delta}{\Gamma, \exists x A \vdash \Delta} \exists_g^s & \frac{\Gamma \vdash A[x := ?x], \{\exists x A\}, \Delta}{\Gamma \vdash \exists x A, \Delta} \exists_d^d
\end{array}$$

FIGURE 7 – Règles des quantificateurs pour la méthode des tableaux