*Les calculatrices sont interdites.*

N.B. : Le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction.
Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

PRÉAMBULE : Les deux parties qui composent ce sujet sont indépendantes et peuvent être traitées par les candidats dans un ordre quelconque.

# Partie I : Automates et langages

Le but de cet exercice est l'étude des propriétés des opérations de dérivation à gauche $m^{-1}.\mathcal{A}$ et à droite $\mathcal{A}.m^{-1}$ d'un automate fini $\mathcal{A}$ selon un mot $m$.

## 1 Automate fini

Pour simplifier les preuves, nous nous limiterons au cas des automates finis semi-indéterministes, c'est-à-dire les automates finis non déterministes qui ne contiennent pas de transitions instantanées (ou $\epsilon$-transitions). Les résultats étudiés s'étendent au cadre des automates finis quelconques.

### 1.1 Représentation d'un automate fini

**Déf. I.1 (Automate fini semi-indéterministe)** *Soit l'alphabet $X$ (un ensemble de symboles), soit $\Lambda$ le symbole représentant le mot vide ($\Lambda \notin X$), soit $X^\star$ l'ensemble contenant $\Lambda$ et les mots composés de symboles de $X$ (donc $\Lambda \in X^\star$), un automate fini semi-indéterministe sur $X$ est un quintuplet $A = (Q,X,I,T,\gamma)$ composé de :*

- *Un ensemble fini d'états : $Q$ ;*
- *Un ensemble d'états initiaux : $I \subseteq Q$ ;*
- *Un ensemble d'états terminaux : $T \subseteq Q$ ;*
- *Une relation de transition confondue avec son graphe : $\gamma \subseteq Q \times X \times Q$.*

Pour une transition $(o,e,d)$ donnée, nous appelons $o$ l'origine de la transition, $e$ l'étiquette de la transition et $d$ la destination de la transition.

Remarquons que $\gamma$ est le graphe d'une application de transition $\delta : Q \times X \to \mathcal{P}(Q)$ dont les valeurs sont définies par :

$$\forall o \in Q, \forall e \in X, \delta(o,e) = \{d \in Q \mid (o,e,d) \in \gamma\}$$

La notation $\gamma$ est plus adaptée que $\delta$ à la formalisation et la construction des preuves dans le cadre des automates indéterministes.



### 1.2 Représentation graphique d'un automate

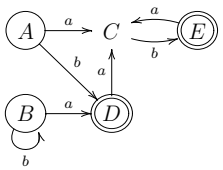
Les automates peuvent être représentés par un schéma suivant les conventions :

- les valeurs de la relation de transition $\gamma$ sont représentées par un graphe orienté dont les nœuds sont les états et les arêtes sont les transitions ;
- un état initial est entouré d'un cercle $(i)$ ;
- un état terminal est entouré d'un double cercle $((t))$ ;
- un état qui est à la fois initial et terminal est entouré d'un triple cercle $(((it)))$ ;
- une arête étiquetée par le symbole $e \in X$ va de l'état $o$ à l'état $d$ si et seulement si $(o,e,d) \in \gamma$.

**Exemple I.1** *L'automate $\mathcal{E} = (Q,X,I,T,\gamma)$ avec :*

$$\begin{array}{l} Q = \{A,B,C,D,E\} \\ X = \{a,b\} \\ I = \{A,B\} \\ T = \{D,E\} \\ \gamma = \{(A,a,C),(A,b,D),(B,a,D),(B,b,B),(C,b,E),(D,a,C),(E,a,C)\} \end{array}$$

*est représenté par le graphe suivant :*




### 1.3 Langage reconnu par un automate fini

Soit $\gamma^\star$ l'extension de $\gamma$ à $Q \times X^\star \times Q$ définie par :

$$\begin{cases} \forall q \in Q, (q,\Lambda,q) \in \gamma^\star \\ \forall e \in X, \forall m \in X^\star, \forall o \in Q, \forall d \in Q, (o,e.m,d) \in \gamma^\star \Leftrightarrow \exists q \in Q, ((o,e,q) \in \gamma) \wedge ((q,m,d) \in \gamma^\star) \end{cases}$$

Le langage sur $X^\star$ reconnu par un automate fini est :

$$L(A) = \{m \in X^\star \mid \exists o \in I, \exists d \in T, (o,m,d) \in \gamma^\star\}$$

**Question I.1** *Donner sans la justifier une expression régulière ou ensembliste représentant le langage reconnu par l'automate $\mathcal{E}$ de l'exemple I.1.*

## 2 Opérations de dérivation

### 2.1 Définitions

Soient les opérations internes sur les automates finis semi-indéterministes définies par :

**Déf. I.2 (Dérivées selon un mot)** *Soient $\mathcal{A} = (Q,X,I,T,\gamma)$ un automate fini semi-indéterministe et $m \in X^\star$, les automates $m^{-1}.\mathcal{A}$ (dérivation à gauche selon $m$) et $\mathcal{A}.m^{-1}$ (dérivation à droite selon $m$) sont définis par :*

$$m^{-1}.\mathcal{A} = (Q,X,\{q \in Q \mid \exists i \in I,(i,m,q) \in \gamma^\star\},T,\gamma)$$
$$\mathcal{A}.m^{-1} = (Q,X,I,\{q \in Q \mid \exists t \in T,(q,m,t) \in \gamma^\star\},\gamma)$$

**Question I.2** *En considérant l'exemple I.1, construire les automates $a^{-1}.\mathcal{E}$, $b^{-1}.\mathcal{E}$, $\mathcal{E}.a^{-1}$ et $\mathcal{E}.b^{-1}$ (seuls les états et les transitions utiles, c'est-à-dire accessibles depuis les états initiaux, devront être construits).*

**Question I.3** *Caractériser les langages reconnus par $a^{-1}.\mathcal{E}$, $b^{-1}.\mathcal{E}$, $\mathcal{E}.a^{-1}$ $\mathcal{E}.b^{-1}$, par une expression régulière ou ensembliste.*

### 2.2 Propriétés

**Question I.4** *Montrer que : si $\mathcal{A}$ est un automate fini semi-indéterministe et $m \in X^\star$ un mot, alors $m^{-1}.\mathcal{A}$ et $\mathcal{A}.m^{-1}$ sont des automates finis semi-indéterministes.*

**Question I.5** *Montrer que :*

$$\forall m \in X^\star, \forall n \in X^\star, \forall o \in Q, \forall q \in Q, \forall d \in Q, (o,m,q) \in \gamma^\star \wedge (q,n,d) \in \gamma^\star \Leftrightarrow (o,m.n,d) \in \gamma^\star$$

**Question I.6** *Soit $\mathcal{A}$ un automate fini semi-indéterministe, montrer que :*

$$\begin{cases} \forall m \in X^\star, \forall n \in X^\star, n \in L(m^{-1}.\mathcal{A}) \Leftrightarrow m.n \in L(\mathcal{A}) \\ \forall m \in X^\star, \forall n \in X^\star, n \in L(\mathcal{A}.m^{-1}) \Leftrightarrow n.m \in L(\mathcal{A}) \end{cases}$$



# Partie II : Algorithmique et programmation en CaML

Cette partie doit être traitée par les étudiants qui ont utilisé le langage CaML dans le cadre des enseignements d'informatique. Les fonctions écrites devront être récursives ou faire appel à des fonctions auxiliaires récursives. Elles ne devront pas utiliser d'instructions itératives (`for`, `while`, ...) ni de références.

Format de desciption d'une fonction : La description d'une fonction, lorsque celle-ci est demandée, c'est-à-dire aux questions II.23 et II.29, doit au moins contenir :

1. L'objectif général ;
2. Le rôle des paramètres de la fonction ;
3. Les contraintes sur les valeurs des paramètres ;
4. Les caractéristiques du résultat renvoyé par la fonction ;
5. Le rôle des variables locales à la fonction ;
6. Le principe de l'algorithme ;
7. Des arguments de terminaison du calcul pour toutes les valeurs des paramètres qui vérifient les contraintes présentées au point 3.

L'objectif de ce problème est l'étude d'une stratégie complète et cohérente d'interrogation d'une base de connaissances. Une base de connaissances est une représentation de relations logiques existant entre des faits. Cette stratégie repose sur un algorithme de construction d'une base complète, cohérente et minimale. Cet algorithme qui préserve le contenu logique de la base est généralement appelé « *complétion* » de la base.

## 1 Préliminaire : Calcul des propositions

La représentation d'une base de connaissances repose sur le calcul des propositions.

**Déf. II.1 (Propositions)** *Soit $\mathcal{F} = \{f_0,f_1,\ldots\}$ un ensemble dénombrable de symboles, appelés faits (ou variables propositionnelles), soient les constantes vrai et faux notées $\top,\bot$, soit l'opérateur unaire $\neg$ (négation), soient les opérateurs binaires $\vee$ (disjonction), $\wedge$ (conjonction), $\Rightarrow$ (implication); l'ensemble $\mathcal{P}(\mathcal{F})$ des propositions est le plus petit ensemble tel que :*

- *$\mathcal{F} \subseteq \mathcal{P}(\mathcal{F})$ ;*
- *$\{\top,\bot\} \subseteq \mathcal{P}(\mathcal{F})$ ;*
- *si $P \in \mathcal{P}(\mathcal{F})$ alors $\neg P \in \mathcal{P}(\mathcal{F})$ ;*
- *si $P_1,P_2 \in \mathcal{P}^2(\mathcal{F})$ et $op \in \{\vee,\wedge,\Rightarrow\}$ alors $P_1\, op\, P_2 \in \mathcal{P}(\mathcal{F})$ ;*
- *les propositions de $\mathcal{P}(\mathcal{F})$ sont finies, c'est-à-dire obtenues par application d'un nombre fini de fois des règles précédentes.*

Nous noterons les faits en utilisant les minuscules de l'alphabet romain. Nous noterons les propositions et les ensembles de faits en utilisant les majuscules de l'alphabet romain.

**Déf. II.2 (Valuation)** *Soit $\mathbb{B} = \{0,1\}$ les valeurs de vérité, une valuation est une application $\sigma : \mathcal{F} \to \mathbb{B}$. Cette application est étendue en une application unique $\hat{\sigma} : \mathcal{P}(\mathcal{F}) \to \mathbb{B}$ par les égalités :*

$$\begin{aligned}
&\hat{\sigma}(\top) = 1\\
&\hat{\sigma}(\bot) = 0\\
&\hat{\sigma}(f_i) = \sigma(f_i)\\
&\hat{\sigma}(\neg P) = 1 - \hat{\sigma}(P)\\
&\hat{\sigma}(P_1 \vee P_2) = 1 - (1 - \hat{\sigma}(P_1)) \times (1 - \hat{\sigma}(P_2))\\
&\hat{\sigma}(P_1 \wedge P_2) = \hat{\sigma}(P_1) \times \hat{\sigma}(P_2)\\
&\hat{\sigma}(P_1 \Rightarrow P_2) = 1 - \hat{\sigma}(P_1) \times (1 - \hat{\sigma}(P_2))
\end{aligned}$$

**Déf. II.3 (Équivalence)** *Soient deux propositions $P_1$,$P_2$ éléments de $\mathcal{P}(\mathcal{F})$, $P_1$ est équivalente à $P_2$ (notée $P_1 \equiv P_2$) si et seulement si pour toute valuation $\sigma$, $\hat{\sigma}(P_1) = \hat{\sigma}(P_2)$.*

**Question II.1** *Montrer que $a \Rightarrow b \equiv \neg a \vee b$.*

**Déf. II.4 (Tautologie)** *Soit une proposition $P \in \mathcal{P}(\mathcal{F})$, $P$ est une tautologie si et seulement si $P \equiv \top$.*

**Déf. II.5 (Proposition absurde)** *Soit une proposition $P \in \mathcal{P}(\mathcal{F})$, $P$ est absurde si et seulement si $P \equiv \bot$.*

**Déf. II.6 (Littéral)** *Un littéral est une proposition qui prend l'une des formes suivantes :*

- *un fait (littéral positif) ;*
- *la négation d'un fait (littéral négatif).*

**Déf. II.7 (Clause, clause duale)** *Une clause est une disjonction de littéraux deux à deux distincts. Une clause duale est une conjonction de littéraux deux à deux distincts.*

**Exemple II.1** *$c \vee \neg b \vee a$ est une clause. $b \wedge c \wedge \neg a$ est une clause duale.*

**Déf. II.8 (Forme conjonctive, disjonctive)** *Une forme conjonctive est une conjonction de clauses. Une forme disjonctive est une disjonction de clauses duales.*

**Exemple II.2** *$(a \vee \neg c) \wedge \neg b$ est une forme conjonctive. $c \vee (\neg a \wedge b)$ est une forme disjonctive.*

**Question II.2** *Soit la proposition $P = (a \wedge c \Rightarrow \bot) \wedge (a \Rightarrow b) \wedge (\top \Rightarrow b \vee d)$, transformer $P$ en une forme conjonctive $C$, puis en une forme disjonctive $D$ telles que $P \equiv C \equiv D$. Vous utiliserez pour cela les formules de De Morgan.*

# 2 Représentation et codage d'un ensemble de faits

La représentation et les opérations de manipulation des bases de connaissances reposent sur la structure d'ensemble. Nous allons dans une première étape étudier un codage de cette structure qui contiendra des faits. Cette structure sera ensuite adaptée aux connaissances.



## 2.1 Codage d'un fait

Un fait est représenté par le type de base `string`.

```
type fait == string;;
```

## 2.2 Codage d'un ensemble de faits

Nous allons utiliser un codage extrêmement simple : un ensemble est une liste contenant exactement un exemplaire de chaque élément contenu dans l'ensemble. Les opérations de manipulation d'un ensemble devront préserver cette propriété.

Un ensemble de faits est représenté par le type `faits` équivalent à une liste de `fait`.

```
type faits == fait list;;
```

Le parcours d'un ensemble sera donc effectué de la même manière que celui d'une liste.

Nous allons maintenant définir plusieurs opérations sur les ensembles de faits qui pourront être utilisées dans la suite du sujet. Ces opérations seront ensuite étendues aux ensembles de connaissances.

Nous supposons que toutes les listes représentant des ensembles, qui sont passées comme paramètres des fonctions, ne contiennent au plus qu'une fois chaque élément.

## 2.3 Opération sur la structure d'ensemble

Pour les calculs de complexité, nous noterons $\mid E \mid$ la taille de l'ensemble $E$, c'est-à-dire le nombre de faits qu'il contient.

### 2.3.1 Appartenance à un ensemble

Une première opération consiste à tester l'appartenance d'un fait à un ensemble.

**Question II.3** *Écrire en CaML une fonction `appartenance` de type `fait -> faits -> bool` telle que l'appel `(appartenance f E)` renvoie la valeur `true` si l'ensemble `E` contient le fait `f` et la valeur `false` sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

**Question II.4** *Donner un exemple de valeurs des paramètres `f` et `E` de la fonction `appartenance` qui correspond au pire cas en nombre d'appels récursifs effectués.*

*Calculer une estimation de la complexité dans le pire cas de la fonction `appartenance` en fonction de la taille de l'ensemble `E`. Cette estimation ne prendra en compte que le nombre d'appels récursifs effectués.*

#### 2.3.2 Ajout dans un ensemble

La deuxième opération est l'ajout d'un fait à un ensemble.

**Question II.5** *Écrire en CaML une fonction* `ajout` *de type* `fait -> faits -> faits` *telle que l'appel* `(ajout f E)` *renvoie un ensemble contenant les mêmes faits que l'ensemble* `E` *ainsi que le fait* `f` *s'il ne figurait pas déjà dans* `E`*. L'ensemble renvoyé contiendra exactement une fois le fait* `f`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

#### 2.3.3 Inclusion entre deux ensembles

La troisième opération est le test d'inclusion du contenu de deux ensembles.

**Question II.6** *Écrire en CaML une fonction* `inclusion` *de type* `faits -> faits -> bool` *telle que l'appel* `(inclusion E1 E2)` *renvoie la valeur* `true` *si l'ensemble* `E2` *contient au moins les mêmes faits que l'ensemble* `E1` *et la valeur* `false` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

**Question II.7** *Donner un exemple de valeurs des paramètres* `E1` *et* `E2` *de la fonction* `inclusion` *qui correspond au pire cas en nombre d'appels récursifs effectués.*

*Calculer une estimation de la complexité dans le pire cas de la fonction* `inclusion` *en fonction des tailles des ensembles* `E1` *et* `E2`*. Cette estimation ne prendra en compte que le nombre d'appels récursifs effectués.*

#### 2.3.4 Égalité entre deux ensembles

La quatrième opération est le test d'égalité du contenu de deux ensembles.

**Question II.8** *Écrire en CaML une fonction* `egalite` *de type* `faits -> faits -> bool` *telle que l'appel* `(egalite E1 E2)` *renvoie la valeur* `true` *si les deux ensembles* `E1` *et* `E2` *contiennent exactement les mêmes faits et la valeur* `false` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

#### 2.3.5 Soustraction de deux ensembles

La dernière opération étudiée est la construction d'un ensemble résultant de la soustraction d'un ensemble depuis un autre ensemble.

**Question II.9** *Écrire en CaML une fonction* `soustraction` *de type* `faits -> faits -> faits` *telle que l'appel* `(soustraction E1 E2)` *renvoie un ensemble contenant les faits qui sont contenus dans* `E1` *et ne sont pas contenus dans* `E2`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*



#### 2.3.6 Autres opérations prédéfinies

Nous supposons prédéfinies les fonctions suivantes pour les ensembles, dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `union : faits -> faits -> faits` telle que l'appel `(union E1 E2)` renvoie un ensemble contenant les faits contenus dans `E1` ainsi que les faits contenus dans `E2` ;
- `intersection : faits -> faits -> faits` telle que l'appel `(intersection E1 E2)` renvoie un ensemble contenant les faits contenus à la fois dans `E1` et dans `E2`.

# 3 Représentation et codage d'une connaissance

## 3.1 Représentation d'une connaissance

**Déf. II.9 (Connaissance)** *Une connaissance $\gamma$ notée $\gamma = H \vdash C$ est composée de deux ensembles finis de faits (ou variables propositionnelles) : les hypothèses $H = \{h_i\}_{i\in I}$ et les conclusions $C = \{c_j\}_{j\in J}$.*

| Nous noterons les connaissances en utilisant les minuscules de l'alphabet grec. |
|---|

**Déf. II.10 (Interprétation logique)** *L'interprétation logique, notée $\mathcal{I}(\gamma)$, d'une connaissance $\gamma = \{h_i\}_{i\in I} \vdash \{c_j\}_{j\in J}$ est :*

$$\mathcal{I}(\gamma) = (\top \wedge \bigwedge_{i\in I} h_i) \Rightarrow (\bot \vee \bigvee_{j\in J} c_j)$$

*Intuitivement, elle signifie que sous les hypothèses de $\{h_i\}_{i\in I}$, nous pouvons déduire une des conclusions de $\{c_j\}_{j\in J}$. $\top$ permet de traiter le cas $I = \emptyset$. $\bot$ permet de traiter le cas $J = \emptyset$. Notons que :*

- $\mathcal{I}(\emptyset \vdash \{c_j\}_{j\in J}) = \top \Rightarrow (\bot \vee \bigvee_{j\in J} c_j)$ ;
- $\mathcal{I}(\{h_i\}_{i\in I} \vdash \emptyset) = (\top \wedge \bigwedge_{i\in I} h_i) \Rightarrow \bot$ ;
- $\mathcal{I}(\emptyset \vdash \emptyset) = \top \Rightarrow \bot$.

**Exemple II.3** *Les formules logiques suivantes sont des connaissances :*

- $\{ a \} \vdash \{ b \}$ ;
- $\{ a, c \} \vdash \emptyset$ ;
- $\emptyset \vdash \{ b, d \}$.

**Question II.10** *Soit $\gamma = \{h_i\}_{i\in I} \vdash \{c_j\}_{j\in J}$ une connaissance quelconque, donner une clause $P$ telle que $\mathcal{I}(\gamma) \equiv P$.*

**Déf. II.11 (Connaissance absurde)** *Une connaissance $\gamma$ est absurde si et seulement si son interprétation $\mathcal{I}(\gamma)$ est absurde.*

**Question II.11** *Montrer que la seule connaissance absurde est $\emptyset \vdash \emptyset$.*

### 3.2 Codage d'une connaissance

Pour les variables ou les constantes dont le nom est pris dans l'alphabet grec ($\gamma$, ...), l'identifiant en CaML sera leur nom en toute lettre (`gamma`, ...).

Une connaissance est représentée par le type `connaissance` équivalent à un couple composé de deux ensembles de faits.

```
type connaissance == faits * faits ;;
```

**Question II.12** *Écrire en CaML une fonction* `comparaison` *de type* `connaissance -> connaissance -> bool` *telle que l'appel* `(comparaison gamma1 gamma2)` *renvoie la valeur* `true` *si les deux connaissances* `gamma1` *et* `gamma2` *contiennent exactement les mêmes hypothèses et conclusions et la valeur* `false` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

## 4 Représentation et codage d'une base de connaissances

### 4.1 Représentation d'une base de connaissances

**Déf. II.12 (Base de connaissances)** *Une base de connaissances $\Omega$ est un ensemble fini de connaissances $\Omega = \{\gamma_k\}_{k\in K}$.*

Nous noterons les bases de connaissances en utilisant les majuscules de l'alphabet grec.

**Déf. II.13 (Interprétation logique)** *L'interprétation logique, notée $\mathcal{I}(\Omega)$, d'une base $\Omega = \{\gamma_k\}_{k\in K}$ est définie par :*

$$\mathcal{I}(\Omega) = \top \wedge \bigwedge_{k\in K} \mathcal{I}(\gamma_k)$$

*Elle correspond à la conjonction des interprétations logiques de chaque connaissance.*
*Notons que : $\mathcal{I}(\emptyset) = \top$.*

**Question II.13** *Soit $\Omega = \{\gamma_k\}_{k\in K}$ avec $\gamma_k = \{h_i\}_{i\in I_k} \vdash \{c_j\}_{j\in J_k}$ une base de connaissances quelconque, donner une forme conjonctive $C$ telle que $\mathcal{I}(\Omega) \equiv C$.*

**Déf. II.14 (Équivalence)** *Soient deux bases $\Omega_1$ et $\Omega_2$, $\Omega_1$ est équivalente à $\Omega_2$ (notée $\Omega_1 \equiv \Omega_2$) si et seulement si $\mathcal{I}(\Omega_1) \equiv \mathcal{I}(\Omega_2)$.*

### 4.2 Codage d'une base de connaissances

Une base de connaissances est représentée par le type `base` équivalent à une liste de connaissances.

```
type base == connaissance list;;
```



**Exemple II.4** *La base composée des connaissances de l'exemple II.3 sera représentée par la valeur :*

```
[
  ( [ "a" ] , [ "b" ] ) ;
  ( [ "a" ; "c" ] , [] ) ;
  ( [ ] , [ "b" ; "d" ] )
]
```

Une base est un ensemble de connaissances. Il est donc nécessaire d'adapter les opérations définies dans la sous-section 2.2 sur les ensembles de faits.

Nous supposons prédéfinies les fonctions suivantes pour les bases, dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `appartenanceB : connaissance -> base -> bool`
- `ajoutB : connaissance -> base -> base`
- `inclusionB : base -> base -> bool`
- `egaliteB : base -> base -> bool`
- `unionB : base -> base -> base`
- `intersectionB : base -> base -> base`
- `soustractionB : base -> base -> base`

## 5 Élimination des tautologies

Une base de connaissances peut contenir des connaissances inutiles, c'est-à-dire qui n'apportent aucune information pertinente lors d'une interrogation, par exemple les tautologies. Pour réduire la taille de la base et les coûts de l'opération d'interrogation, nous nous intéressons à l'élimination des tautologies.

**Déf. II.15 (Tautologie)** *Une connaissance $\gamma$ est une tautologie si et seulement si son interprétation $\mathcal{I}(\gamma)$ est une tautologie.*

**Question II.14** *Quelles sont les tautologies parmi les connaissances suivantes (justifier vos réponses) :*

- $\gamma_1 = \{a,b\} \vdash \{c\}$ ;
- $\gamma_2 = \{a\} \vdash \{a\}$ ;
- $\gamma_3 = \{b\} \vdash \{b,c\}$ ;
- $\gamma_4 = \{a,c\} \vdash \{c\}$ ;
- $\gamma_5 = \{b\} \vdash \emptyset$ ;
- $\gamma_6 = \emptyset \vdash \{c\}$.

**Question II.15** *Donner une relation entre les hypothèses et les conclusions d'une connaissance qui soit une condition nécessaire et suffisante pour que cette connaissance soit une tautologie. Donner une preuve de cette condition.*

Nous supposons prédéfinie la fonction `tautologie` de type `connaissance -> bool` telle que l'appel `(tautologie gamma)` renvoie la valeur `true` si la connaissance `gamma` est une tautologie et la valeur `false` sinon. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

**Question II.16** *Soit $\Omega$ une base et $\gamma$ une tautologie, montrer que $\Omega \cup \{\gamma\} \equiv \Omega$.*

**Déf. II.16** *Soit $\Omega$ une base, nous noterons $]\,\Omega\,[$ la base $\Omega$ privée de ses tautologies, c'est-à-dire :*

$$]\,\Omega\,[ = \{\gamma \in \Omega \mid \gamma \not\equiv \top\}$$

*Nous pouvons déduire de la question II.16 que $]\,\Omega\,[ \equiv \Omega$.*

**Question II.17** *Écrire en CaML une fonction `elimination` de type `base -> base` telle que l'appel `(elimination Omega)` renvoie une base de connaissances contenant les mêmes connaissances que $]\,\Omega\,[$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

# 6 Minimisation d'une base de connaissances

Une base de connaissances peut contenir des connaissances redondantes, en particulier, elle peut contenir des connaissances plus générales que d'autres. Pour réduire la taille de la base et les coûts de l'opération d'interrogation, nous nous intéressons à la minimisation de la base, c'est-à-dire à ne conserver que les connaissances les plus générales.

**Déf. II.17** *Une connaissance $\gamma_1 = H_1 \vdash C_1$ est dite plus générale qu'une connaissance $\gamma_2 = H_2 \vdash C_2$ si et seulement si $H_1 \subseteq H_2$ et $C_1 \subseteq C_2$. Cette relation sera notée $\gamma_2 \prec \gamma_1$.*

**Question II.18** *Donner les relations $\prec$ entre les connaissances suivantes :*

- $\gamma_1 = \{a,b\} \vdash \{c,d\}$ ;
- $\gamma_2 = \{a,c\} \vdash \{b,d\}$ ;
- $\gamma_3 = \{a\} \vdash \{d\}$ ;
- $\gamma_4 = \{c\} \vdash \{b,d\}$ ;
- $\gamma_5 = \emptyset \vdash \emptyset$.

**Question II.19** *Soient deux connaissances $\gamma_1$ et $\gamma_2$, montrer que les bases $\{\gamma_1,\gamma_2\}$ et $\{\gamma_1\}$ sont équivalentes si et seulement si $\gamma_2 \prec \gamma_1$. Pour cela, vous pouvez considérer une valuation $\sigma$ et envisager les différents cas possibles.*

**Déf. II.18 (Base minimale)** *Soit $\Omega$ une base de connaissance, la base minimale $]\!]\,\Omega\,[\![$ associée à $\Omega$ est définie par :*

$$]\!]\,\Omega\,[\![ = \{\beta \in \Omega \mid \forall \gamma \in \Omega, \gamma \neq \beta \Rightarrow \beta \not\prec \gamma\}$$

*Nous pouvons déduire de la question II.19 que $]\!]\,\Omega\,[\![ \equiv \Omega$.*



**Question II.20** *Écrire en CaML une fonction `absorbable` de type `connaissance -> base -> bool` telle que l'appel `(absorbable gamma Omega)` renvoie la valeur `true` si la base `Omega` contient une connaissance plus générale que `gamma` et la valeur `false` sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Nous supposons prédéfinie la fonction `minimisation` de type `base -> base` telle que l'appel `(minimisation Omega)` renvoie $]\!]\,\Omega\,[\![$. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

# 7 Complétion d'une base de connaissances

La *complétion* d'une base de connaissances consiste à construire l'ensemble de toutes les connaissances qui peuvent être déduites d'une base donnée. Cette opération permet ensuite de réduire les coûts d'interrogation de la base.

**Déf. II.19 (Déduction de connaissances)** *Soient les connaissances $\gamma_1 = H_1 \vdash C_1$ et $\gamma_2 = H_2 \vdash C_2$, l'opérateur $\rhd$ de déduction de connaissances construit une base notée $\gamma_1 \rhd \gamma_2$ et définie par :*

$$\gamma_1 \rhd \gamma_2 = \{(H_1 \setminus \{f\}) \cup H_2 \vdash C_1 \cup (C_2 \setminus \{f\}) \mid f \in H_1 \cap C_2\}$$

*Notons que $\gamma_1 \rhd \gamma_2 = \emptyset$ si $H_1 \cap C_2 = \emptyset$.*

*L'ensemble des connaissances qui peuvent être déduites d'une base $\Omega$ est l'ensemble des connaissances construites par application d'un nombre quelconque de fois de l'opérateur $\rhd$ à partir des connaissances de $\Omega$.*

**Question II.21** *Appliquer l'opérateur $\rhd$ sur les connaissances suivantes (ne donner que les résultats différents de $\emptyset$) :*

- $\gamma_1 = \{a,b\} \vdash \{c,d\}$ ;
- $\gamma_2 = \{b,c\} \vdash \{e\}$ ;
- $\gamma_3 = \{e\} \vdash \emptyset$ ;
- $\gamma_4 = \emptyset \vdash \{b,c\}$.

**Question II.22** *Soient les connaissances $\gamma_1 = H_1 \vdash C_1$ et $\gamma_2 = H_2 \vdash C_2$, montrer que si $\mid H_1 \cap C_2 \mid > 1$ alors $\gamma_1 \rhd \gamma_2$ ne contient que des tautologies. Que peut-on en conclure ?*

La composition d'une connaissance $\gamma$ et d'une base $\Omega$ consiste à appliquer l'opérateur de déduction $\rhd$ sur $\gamma$ et sur chaque connaissance de $\Omega$.

Nous souhaitons écrire une fonction `composition` qui prend en paramètre une connaissance $\gamma$ et une base $\Omega$ et qui construit une nouvelle base contenant les connaissances résultant de la composition de `gamma` avec chaque connaissance de la base `Omega`. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.

**Question II.23** *Décrire cette fonction `composition` et expliquer son algorithme selon le format présenté en page 4.*

**Question II.24** *Écrire en CaML cette fonction* `composition` *de type* `connaissance -> base -> base` *telle que l'appel* `(composition gamma Omega)` *renvoie une base contenant les connaissances résultant de la composition de* `gamma` *avec les connaissances de la base* `Omega`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Nous supposons prédéfinie la fonction `deduction` de type `base -> base` telle que l'appel `(deduction Omega)` renvoie une base de connaissances qui contient les connaissances de la base `Omega` ainsi que le résultat des compositions deux à deux des connaissances de `Omega`. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

**Question II.25** *Soient deux connaissances $\gamma_1 = H_1 \vdash C_2$ et $\gamma_2 = H_2 \vdash C_2$ telles que $H_1 \cap C_2 = \{f\}$, montrer que les bases $\{\gamma_1,\gamma_2\} \cup \gamma_1 \rhd \gamma_2$ et $\{\gamma_1,\gamma_2\}$ sont équivalentes.*

**Question II.26** *Soit une base de connaissances quelconque $\Omega$, soit la suite $\{\Omega_i\}$ définie par :*

$$\begin{cases} \Omega_0 = \Omega \\ \Omega_{i+1} = \Omega_i \cup \bigcup\limits_{\gamma_i,\gamma_j \in {\Omega_i}^2} \gamma_i \rhd \gamma_j \end{cases}$$

*Nous admettrons que le résultat de la question II.25 peut être étendu à : $\forall i \in \mathbb{N}, \Omega_{i+1} \equiv \Omega_i$.*

1. *Montrer que la suite $\{\Omega_i\}$ est croissante ;*
2. *Montrer que : $\exists k \in \mathbb{N}, \Omega_{k+1} = \Omega_k$ (soit $l = \min\{k \in \mathbb{N} \mid \Omega_{k+1} = \Omega_k\}$, nous noterons alors $\overline{\Omega} = \Omega_l$ et nous appelerons $\overline{\Omega}$ la complétion de la base $\Omega$) ;*
3. *Montrer que $\overline{\Omega} \equiv \Omega$.*

**Question II.27** *Calculer la complétion $\overline{\Omega}$ de la base $\Omega$ contenant les connaissances suivantes :*

- $\gamma_1 = \{a,b\} \vdash \{c,d\}$ ;
- $\gamma_2 = \{b,c\} \vdash \{e\}$ ;
- $\gamma_3 = \{e\} \vdash \emptyset$.

L'élimination des tautologies et la minimisation de la base obtenue permettent ensuite de réduire la taille de la base et les coûts d'interrogation.

**Question II.28** *Éliminer les tautologies et minimiser la réponse que vous avez proposée pour la question précédente.*

Nous souhaitons écrire une fonction `completion` qui prend en paramètre une base $\Omega$ et qui construit une nouvelle base contenant les mêmes connaissances que $\overline{\Omega}$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.

**Question II.29** *Décrire cette fonction* `completion` *et expliquer son algorithme selon le format présenté en page 4.*

**Question II.30** *Écrire en CaML cette fonction* `completion` *de type* `base -> base` *telle que l'appel* `(completion Omega)` *renvoie une base de connaissances contenant les mêmes connaissances que $\overline{\Omega}$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*



# 8 Interrogation d'une base

Les connaissances contenues dans la base établissent un lien entre des faits « *hypothèses* » et des faits « *conclusions* ». Intuitivement, l'interrogation de la base consiste à :

- compléter la base (voir section 7), minimiser la base complétée (voir section 6) et en éliminer les tautologies (voir section 5) ;
- fournir une liste de faits « vrais » et une liste de faits « *faux* » ;
- intégrer ces faits dans la base complétée ;
- rendre la base obtenue minimale (voir section 6) pour obtenir la réponse à la requête.

Définissons ceci formellement :

**Déf. II.20 (Interrogation d'une base)** *Soit une base $\Omega$, la réponse à la requête composée des faits vrais $V$ et des faits faux $F$ est la base $\frac{\Omega}{V,F}$ définie par :*

$$\frac{\Omega}{V,F} = ]\!] \ \{ \ (H \setminus V) \vdash (C \setminus F) \ \mid \ H \vdash C \in \ ]\!] \overline{\Omega} [\![ \ [ \ \} \ [\![$$

**Question II.31** *Soit la base de connaissances proposée à la question II.27, construire la réponse à la requête $V = \{b\}, F = \{d\}$.*

**Question II.32** *Montrer que* $]\,(\frac{\Omega}{V,F})\,[ \equiv \frac{\Omega}{V,F}$.

**Question II.33** *Montrer que* $\overline{(\frac{\Omega}{V,F})} \equiv \frac{\Omega}{V,F}$.

**Question II.34** *Écrire en CaML une fonction* `interrogation` *de type* `faits -> faits -> base -> base` *telle que l'appel* `(interrogation V F Omega)` *renvoie une base de connaissances contenant les mêmes connaissances que $\frac{\Omega}{V,F}$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

# Partie II : Algorithmique et programmation en PASCAL

Cette partie doit être traitée par les étudiants qui ont utilisé le langage PASCAL dans le cadre des enseignements d'informatique. Les fonctions écrites devront être récursives ou faire appel à des fonctions auxiliaires récursives. Elles ne devront pas utiliser d'instructions itératives (`for`, `while`, `repeat`, ...).

Format de desciption d'une fonction : La description d'une fonction, lorsque celle-ci est demandée, c'est-à-dire aux questions II.23 et II.29, doit au moins contenir :

1. L'objectif général ;
2. Le rôle des paramètres de la fonction ;
3. Les contraintes sur les valeurs des paramètres ;
4. Les caractéristiques du résultat renvoyé par la fonction ;
5. Le rôle des variables locales à la fonction ;
6. Le principe de l'algorithme ;
7. Des arguments de terminaison du calcul pour toutes les valeurs des paramètres qui vérifient les contraintes présentées au point 3.

L'objectif de ce problème est l'étude d'une stratégie complète et cohérente d'interrogation d'une base de connaissances. Une base de connaissances est une représentation de relations logiques existant entre des faits. Cette stratégie repose sur un algorithme de construction d'une base complète, cohérente et minimale. Cet algorithme qui préserve le contenu logique de la base est généralement appelé « *complétion* » de la base.

## 1 Préliminaire : Calcul des propositions

La représentation d'une base de connaissances repose sur le calcul des propositions.

**Déf. II.1 (Propositions)** *Soit $\mathcal{F} = \{f_0, f_1, \ldots\}$ un ensemble dénombrable de symboles, appelés faits (ou variables propositionnelles), soient les constantes vrai et faux notées $\top, \bot$, soit l'opérateur unaire $\neg$ (négation), soient les opérateurs binaires $\vee$ (disjonction), $\wedge$ (conjonction), $\Rightarrow$ (implication); l'ensemble $\mathcal{P}(\mathcal{F})$ des propositions est le plus petit ensemble tel que :*

- *$\mathcal{F} \subseteq \mathcal{P}(\mathcal{F})$ ;*
- *$\{\top, \bot\} \subseteq \mathcal{P}(\mathcal{F})$ ;*
- *si $P \in \mathcal{P}(\mathcal{F})$ alors $\neg P \in \mathcal{P}(\mathcal{F})$ ;*
- *si $P_1, P_2 \in \mathcal{P}^2(\mathcal{F})$ et $op \in \{\vee, \wedge, \Rightarrow\}$ alors $P_1 \, op \, P_2 \in \mathcal{P}(\mathcal{F})$ ;*
- *les propositions de $\mathcal{P}(\mathcal{F})$ sont finies, c'est-à-dire obtenues par application d'un nombre fini de fois des règles précédentes.*

Nous noterons les faits en utilisant les minuscules de l'alphabet romain. Nous noterons les propositions et les ensembles de faits en utilisant les majuscules de l'alphabet romain.



**Déf. II.2 (Valuation)** *Soit $\mathbb{B} = \{0,1\}$ les valeurs de vérité, une valuation est une application $\sigma : \mathcal{F} \to \mathbb{B}$. Cette application est étendue en une application unique $\hat{\sigma} : \mathcal{P}(\mathcal{F}) \to \mathbb{B}$ par les égalités :*

$$\begin{aligned}
&\hat{\sigma}(\top) = 1 \\
&\hat{\sigma}(\bot) = 0 \\
&\hat{\sigma}(f_i) = \sigma(f_i) \\
&\hat{\sigma}(\neg P) = 1 - \hat{\sigma}(P) \\
&\hat{\sigma}(P_1 \vee P_2) = 1 - (1 - \hat{\sigma}(P_1)) \times (1 - \hat{\sigma}(P_2)) \\
&\hat{\sigma}(P_1 \wedge P_2) = \hat{\sigma}(P_1) \times \hat{\sigma}(P_2) \\
&\hat{\sigma}(P_1 \Rightarrow P_2) = 1 - \hat{\sigma}(P_1) \times (1 - \hat{\sigma}(P_2))
\end{aligned}$$

**Déf. II.3 (Équivalence)** *Soient deux propositions $P_1, P_2$ éléments de $\mathcal{P}(\mathcal{F})$, $P_1$ est équivalente à $P_2$ (notée $P_1 \equiv P_2$) si et seulement si pour toute valuation $\sigma$, $\hat{\sigma}(P_1) = \hat{\sigma}(P_2)$.*

**Question II.1** *Montrer que $a \Rightarrow b \equiv \neg a \vee b$.*

**Déf. II.4 (Tautologie)** *Soit une proposition $P \in \mathcal{P}(\mathcal{F})$, $P$ est une tautologie si et seulement si $P \equiv \top$.*

**Déf. II.5 (Proposition absurde)** *Soit une proposition $P \in \mathcal{P}(\mathcal{F})$, $P$ est absurde si et seulement si $P \equiv \bot$.*

**Déf. II.6 (Littéral)** *Un littéral est une proposition qui prend l'une des formes suivantes :*

- *un fait (littéral positif) ;*
- *la négation d'un fait (littéral négatif).*

**Déf. II.7 (Clause, clause duale)** *Une clause est une disjonction de littéraux deux à deux distincts. Une clause duale est une conjonction de littéraux deux à deux distincts.*

**Exemple II.1** *$c \vee \neg b \vee a$ est une clause. $b \wedge c \wedge \neg a$ est une clause duale.*

**Déf. II.8 (Forme conjonctive, disjonctive)** *Une forme conjonctive est une conjonction de clauses. Une forme disjonctive est une disjonction de clauses duales.*

**Exemple II.2** *$(a \vee \neg c) \wedge \neg b$ est une forme conjonctive. $c \vee (\neg a \wedge b)$ est une forme disjonctive.*

**Question II.2** *Soit la proposition $P = (a \wedge c \Rightarrow \bot) \wedge (a \Rightarrow b) \wedge (\top \Rightarrow b \vee d)$, transformer $P$ en une forme conjonctive $C$, puis en une forme disjonctive $D$ telles que $P \equiv C \equiv D$. Vous utiliserez pour cela les formules de De Morgan.*

## 2 Représentation et codage d'un ensemble de faits

La représentation et les opérations de manipulation des bases de connaissances reposent sur la structure d'ensemble. Nous allons dans une première étape étudier un codage de cette structure qui contiendra des faits. Cette structure sera ensuite adaptée aux connaissances.

## 2.1 Codage d'un fait

Un fait est représenté par le type de base `STRING`.

## 2.2 Codage d'un ensemble de faits

Nous allons utiliser un codage extrêmement simple : un ensemble est une liste contenant exactement un exemplaire de chaque élément contenu dans l'ensemble. Les opérations de manipulation d'un ensemble devront préserver cette propriété.

Un ensemble de faits est représenté par le type de base `FAITS` correspondant à une liste de faits.

Le parcours d'un ensemble sera donc effectué de la même manière que celui d'une liste.

Nous supposons prédéfinies les constantes et les fonctions suivantes dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `NIL` représente la liste vide de faits ;
- `FUNCTION LF_Ajout(f:FAIT;E:FAITS):FAITS;` renvoie une liste de faits composée d'un premier fait `f` et du reste de la liste contenu dans `E` ;
- `FUNCTION LF_Premier(E:FAITS):FAIT;` renvoie le premier fait de la liste `E`. Cette liste ne doit pas être vide ;
- `FUNCTION LF_Reste(E:FAITS):FAITS;` renvoie le reste de la liste `E` privée de son premier fait. Cette liste ne doit pas être vide ;
- `FUNCTION LF_Juxtaposition(E1:FAITS;E2:FAITS):FAITS;` renvoie la liste composée de la liste `E1` suivie de la liste `E2`.

Nous allons maintenant définir plusieurs opérations sur les ensembles de faits qui pourront être utilisées dans la suite du sujet. Ces opérations seront ensuite étendues aux ensembles de connaissances.

| Nous supposons que toutes les listes représentant des ensembles, qui sont passées comme paramètres des fonctions, ne contiennent au plus qu'une fois chaque élément. |
|---|

## 2.3 Opération sur la structure d'ensemble

Pour les calculs de complexité, nous noterons $\mid E \mid$ la taille de l'ensemble $E$, c'est-à-dire le nombre de faits qu'il contient.

### 2.3.1 Appartenance à un ensemble

Une première opération consiste à tester l'appartenance d'un fait à un ensemble.

**Question II.3** *Écrire en PASCAL une fonction* `appartenance(f:FAIT;E:FAITS):BOOLEAN;` *telle que l'appel* `appartenance(f,E)` *renvoie la valeur* `TRUE` *si l'ensemble* `E` *contient le fait* `f` *et la valeur* `FALSE` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*



**Question II.4** *Donner un exemple de valeurs des paramètres* `f` *et* `E` *de la fonction* `appartenance` *qui correspond au pire cas en nombre d'appels récursifs effectués.*

*Calculer une estimation de la complexité dans le pire cas de la fonction* `appartenance` *en fonction de la taille de l'ensemble* `E`*. Cette estimation ne prendra en compte que le nombre d'appels récursifs effectués.*

### 2.3.2 Ajout dans un ensemble

La deuxième opération est l'ajout d'un fait à un ensemble.

**Question II.5** *Écrire en PASCAL une fonction* `ajout(f:FAIT;E:FAITS):FAITS;` *telle que l'appel* `ajout(f,E)` *renvoie un ensemble contenant les mêmes faits que l'ensemble* `f` *ainsi que le fait* `E` *s'il ne figurait pas déjà dans* `E`*. L'ensemble renvoyé contiendra exactement une fois le fait* `f`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

### 2.3.3 Inclusion entre deux ensembles

La troisième opération est le test d'inclusion du contenu de deux ensembles.

**Question II.6** *Écrire en PASCAL une fonction* `inclusion(E1:FAITS;E2:FAITS):BOOLEAN;` *telle que l'appel* `inclusion(E1,E2)` *renvoie la valeur* `TRUE` *si l'ensemble* `E2` *contient au moins les mêmes faits que l'ensemble* `E1` *et la valeur* `FALSE` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

**Question II.7** *Donner un exemple de valeurs des paramètres* `E1` *et* `E2` *de la fonction* `inclusion` *qui correspond au pire cas en nombre d'appels récursifs effectués.*

*Calculer une estimation de la complexité dans le pire cas de la fonction* `inclusion` *en fonction des tailles des ensembles* `E1` *et* `E2`*. Cette estimation ne prendra en compte que le nombre d'appels récursifs effectués.*

### 2.3.4 Égalité entre deux ensembles

La quatrième opération est le test d'égalité du contenu de deux ensembles.

**Question II.8** *Écrire en PASCAL une fonction* `egalite(E1:FAITS;E2:FAITS):BOOLEAN;` *telle que l'appel* `egalite(E1,E2)` *renvoie la valeur* `TRUE` *si les deux ensembles* `E1` *et* `E2` *contiennent exactement les mêmes faits et la valeur* `FALSE` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

### 2.3.5 Soustraction de deux ensembles

La dernière opération étudiée est la construction d'un ensemble résultant de la soustraction d'un ensemble depuis un autre ensemble.

**Question II.9** *Écrire en PASCAL une fonction* `soustraction(E1:FAITS;E2:FAITS):FAITS;` *telle que l'appel* `soustraction(E1,E2)` *renvoie un ensemble contenant les faits qui sont contenus dans* `E1` *et*

*ne sont pas contenus dans* `E2`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

#### 2.3.6 Autres opérations prédéfinies

Nous supposons prédéfinies les fonctions suivantes pour les ensembles, dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `union(E1:FAITS;E2:FAITS):FAITS;` renvoie un ensemble contenant les faits contenus dans `E1` ainsi que les faits contenus dans `E2` ;
- `intersection(E1:FAITS;E2:FAITS):FAITS;` renvoie un ensemble contenant les faits contenus à la fois dans `E1` et dans `E2`.

# 3 Représentation et codage d'une connaissance

## 3.1 Représentation d'une connaissance

**Déf. II.9 (Connaissance)** *Une connaissance $\gamma$ notée $\gamma = H \vdash C$ est composée de deux ensembles finis de faits (ou variables propositionnelles) : les hypothèses $H = \{h_i\}_{i\in I}$ et les conclusions $C = \{c_j\}_{j\in J}$.*

Nous noterons les connaissances en utilisant les minuscules de l'alphabet grec.

**Déf. II.10 (Interprétation logique)** *L'interprétation logique, notée $\mathcal{I}(\gamma)$, d'une connaissance $\gamma = \{h_i\}_{i\in I} \vdash \{c_j\}_{j\in J}$ est :*

$$\mathcal{I}(\gamma) = (\top \wedge \bigwedge_{i\in I} h_i) \Rightarrow (\bot \vee \bigvee_{j\in J} c_j)$$

*Intuitivement, elle signifie que sous les hypothèses de $\{h_i\}_{i\in I}$, nous pouvons déduire une des conclusions de $\{c_j\}_{j\in J}$. $\top$ permet de traiter le cas $I = \emptyset$. $\bot$ permet de traiter le cas $J = \emptyset$. Notons que :*

- $\mathcal{I}(\emptyset \vdash \{c_j\}_{j\in J}) = \top \Rightarrow (\bot \vee \bigvee_{j\in J} c_j)$ ;
- $\mathcal{I}(\{h_i\}_{i\in I} \vdash \emptyset) = (\top \wedge \bigwedge_{i\in I} h_i) \Rightarrow \bot$ ;
- $\mathcal{I}(\emptyset \vdash \emptyset) = \top \Rightarrow \bot$.

**Exemple II.3** *Les formules logiques suivantes sont des connaissances :*

- $\{ a \} \vdash \{ b \}$ ;
- $\{ a, c \} \vdash \emptyset$ ;
- $\emptyset \vdash \{ b, d \}$.

**Question II.10** *Soit $\gamma = \{h_i\}_{i\in I} \vdash \{c_j\}_{j\in J}$ une connaissance quelconque, donner une clause $P$ telle que $\mathcal{I}(\gamma) \equiv P$.*



**Déf. II.11 (Connaissance absurde)** *Une connaissance $\gamma$ est absurde si et seulement si son interprétation $\mathcal{I}(\gamma)$ est absurde.*

**Question II.11** *Montrer que la seule connaissance absurde est $\emptyset \vdash \emptyset$.*

## 3.2 Codage d'une connaissance

Pour les variables ou les constantes dont le nom est pris dans l'alphabet grec ($\gamma$, ...), l'identifiant en PASCAL sera leur nom en toute lettre (`gamma`, ...).

Une connaissance est représentée par le type de base `CONNAISSANCE` correspondant à un couple composé de deux ensembles de faits.

Nous supposons prédéfinies les constantes et les fonctions suivantes dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `FUNCTION C_Creer(H:FAITS;C:FAITS):CONNAISSANCE;` renvoie une connaissance dont les hypothèses sont les faits de l'ensemble `H` et les conclusions sont les faits de l'ensemble `C` ;
- `FUNCTION C_Hypotheses(gamma:CONNAISSANCE):FAITS;` renvoie les hypothèses de la connaissance `gamma`,
- `FUNCTION C_Conclusions(gamma:CONNAISSANCE):FAITS;` renvoie les conclusions de la connaissance `gamma`.

**Question II.12** *Écrire en PASCAL une fonction* `comparaison(gamma1:CONNAISSANCE;gamma2:CONNAISSANCE):BOOLEAN;` *telle que l'appel* `comparaison(gamma1,gamma2)` *renvoie la valeur* `TRUE` *si les deux connaissances* `gamma1` *et* `gamma2` *contiennent exactement les mêmes hypothèses et conclusions et la valeur* `FALSE` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

# 4 Représentation et codage d'une base de connaissances

## 4.1 Représentation d'une base de connaissances

**Déf. II.12 (Base de connaissances)** *Une base de connaissances $\Omega$ est un ensemble fini de connaissances $\Omega = \{\gamma_k\}_{k\in K}$.*

Nous noterons les bases de connaissances en utilisant les majuscules de l'alphabet grec.

**Déf. II.13 (Interprétation logique)** *L'interprétation logique, notée $\mathcal{I}(\Omega)$, d'une base $\Omega = \{\gamma_k\}_{k\in K}$ est définie par :*

$$\mathcal{I}(\Omega) = \top \wedge \bigwedge_{k\in K} \mathcal{I}(\gamma_k)$$

*Elle correspond à la conjonction des interprétations logiques de chaque connaissance.*
*Notons que : $\mathcal{I}(\emptyset) = \top$.*

**Question II.13** *Soit $\Omega = \{\gamma_k\}_{k\in K}$ avec $\gamma_k = \{h_i\}_{i\in I_k} \vdash \{c_j\}_{j\in J_k}$ une base de connaissances quelconque, donner une forme conjonctive $C$ telle que $\mathcal{I}(\Omega) \equiv C$.*

**Déf. II.14 (Équivalence)** *Soient deux bases $\Omega_1$ et $\Omega_2$, $\Omega_1$ est équivalente à $\Omega_2$ (notée $\Omega_1 \equiv \Omega_2$) si et seulement si $\mathcal{I}(\Omega_1) \equiv \mathcal{I}(\Omega_2)$.*

## 4.2 Codage d'une base de connaissances

Une base de connaissances est représentée par le type de base `BASE` correspondant à une liste de connaissances.

Nous supposons prédéfinies les constantes et les fonctions suivantes dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `NIL` représente la liste vide de connaissances ;
- `FUNCTION B_Ajout(gamma:CONNAISSANCE;Omega:BASE):BASE;` renvoie une liste de connaissances composée d'une première connaissance `gamma` et du reste de la liste contenu dans `Omega` ;
- `FUNCTION B_Premier(Omega:BASE):CONNAISSANCE;` renvoie la première connaissance de la liste `Omega`. Cette liste ne doit pas être vide ;
- `FUNCTION B_Reste(Omega:BASE):BASE;` renvoie le reste de la liste `Omega` privée de sa première connaissance. Cette liste ne doit pas être vide ;
- `FUNCTION B_Juxtaposition(Omega1:BASE;Omega2:BASE):BASE;` renvoie la liste composée de la liste `Omega1` suivie de la suite `Omega2`.

**Exemple II.4** *La base composée des connaissances de l'exemple II.3 sera représentée par la valeur :*

```
B_Ajout(
  C_Creer( LF_Ajout( "a", NIL), LF_Ajout( "b", NIL ) ),
B_Ajout(
  C_Creer( LF_Ajout( "a", LF_Ajout( "c", NIL)), NIL),
B_Ajout(
  C_Creer( NIL, LF_Ajout( "b", LF_Ajout( "d", NIL))),
NIL)))
```

Une base est un ensemble de connaissances. Il est donc nécessaire d'adapter les opérations définies dans la sous-section 2.2 sur les ensembles de faits.

Nous supposons prédéfinies les fonctions suivantes pour les bases, dont le calcul se termine quelles que soient les valeurs de leurs paramètres. Elles pourront éventuellement être utilisées dans les réponses aux questions :

- `appartenanceB(gamma:CONNAISSANCE;Omega:BASE):BOOLEAN;`
- `ajoutB(gamma:CONNAISSANCE;Omega:BASE):BASE;`
- `inclusionB(Omega1:BASE;Omega2:BASE):BOOLEAN;`
- `egaliteB(Omega1:BASE;Omega2:BASE):BASE;`
- `unionB(Omega1:BASE;Omega2:BASE):BASE;`
- `intersectionB(Omega1:BASE;Omega2:BASE):BASE;`
- `soustractionB(Omega1:BASE;Omega2:BASE):BASE;`



# 5 Élimination des tautologies

Une base de connaissances peut contenir des connaissances inutiles, c'est-à-dire qui n'apportent aucune information pertinente lors d'une interrogation, par exemple les tautologies. Pour réduire la taille de la base et les coûts de l'opération d'interrogation, nous nous intéressons à l'élimination des tautologies.

**Déf. II.15 (Tautologie)** *Une connaissance $\gamma$ est une tautologie si et seulement si son interprétation $\mathcal{I}(\gamma)$ est une tautologie.*

**Question II.14** *Quelles sont les tautologies parmi les connaissances suivantes (justifier vos réponses) :*

- $\gamma_1 = \{a,b\} \vdash \{c\}$ ;
- $\gamma_2 = \{a\} \vdash \{a\}$ ;
- $\gamma_3 = \{b\} \vdash \{b,c\}$ ;
- $\gamma_4 = \{a,c\} \vdash \{c\}$ ;
- $\gamma_5 = \{b\} \vdash \emptyset$ ;
- $\gamma_6 = \emptyset \vdash \{c\}$.

**Question II.15** *Donner une relation entre les hypothèses et les conclusions d'une connaissance qui soit une condition nécessaire et suffisante pour que cette connaissance soit une tautologie. Donner une preuve de cette condition.*

Nous supposons prédéfinie la fonction `tautologie(gamma:CONNAISSANCE):BOOLEAN;` telle que l'appel `tautologie(gamma)` renvoie la valeur `TRUE` si la connaissance `gamma` est une tautologie et la valeur `FALSE` sinon. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

**Question II.16** *Soit $\Omega$ une base et $\gamma$ une tautologie, montrer que $\Omega \cup \{\gamma\} \equiv \Omega$.*

**Déf. II.16** *Soit $\Omega$ une base, nous noterons $]\,\Omega\,[$ la base $\Omega$ privée de ses tautologies, c'est-à-dire :*

$$]\,\Omega\,[ = \{\gamma \in \Omega \mid \gamma \not\equiv \top\}$$

*Nous pouvons déduire de la question II.16 que $]\,\Omega\,[ \equiv \Omega$.*

**Question II.17** *Écrire en PASCAL une fonction `elimination(Omega:BASE):BASE;` telle que l'appel `elimination(Omega)` renvoie une base de connaissances contenant les mêmes connaissances que $]\,\Omega\,[$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

# 6 Minimisation d'une base de connaissances

Une base de connaissances peut contenir des connaissances redondantes, en particulier, elle peut contenir des connaissances plus générales que d'autres. Pour réduire la taille de la base et les coûts de l'opération d'interrogation, nous nous intéressons à la minimisation de la base, c'est-à-dire à ne conserver que les connaissances les plus générales.

**Déf. II.17** *Une connaissance $\gamma_1 = H_1 \vdash C_1$ est dite plus générale qu'une connaissance $\gamma_2 = H_2 \vdash C_2$ si et seulement si $H_1 \subseteq H_2$ et $C_1 \subseteq C_2$. Cette relation sera notée $\gamma_2 \prec \gamma_1$.*

**Question II.18** *Donner les relations $\prec$ entre les connaissances suivantes :*

- $\gamma_1 = \{a,b\} \vdash \{c,d\}$ ;
- $\gamma_2 = \{a,c\} \vdash \{b,d\}$ ;
- $\gamma_3 = \{a\} \vdash \{d\}$ ;
- $\gamma_4 = \{c\} \vdash \{b,d\}$ ;
- $\gamma_5 = \emptyset \vdash \emptyset$.

**Question II.19** *Soient deux connaissances $\gamma_1$ et $\gamma_2$, montrer que les bases $\{\gamma_1,\gamma_2\}$ et $\{\gamma_1\}$ sont équivalentes si et seulement si $\gamma_2 \prec \gamma_1$. Pour cela, vous pouvez considérer une valuation $\sigma$ et envisager les différents cas possibles.*

**Déf. II.18 (Base minimale)** *Soit $\Omega$ une base de connaissance, la base minimale $\rrbracket \Omega \llbracket$ associée à $\Omega$ est définie par :*

$$\rrbracket \Omega \llbracket = \{\beta \in \Omega \mid \forall \gamma \in \Omega, \gamma \neq \beta \Rightarrow \beta \not\prec \gamma\}$$

*Nous pouvons déduire de la question II.19 que $\rrbracket \Omega \llbracket \equiv \Omega$.*

**Question II.20** *Écrire en PASCAL une fonction* `absorbable(gamma:CONNAISSANCE;Omega:BASE):BOOLEAN;` *telle que l'appel* `absorbable(gamma,Omega)` *renvoie la valeur* `TRUE` *si la base* `Omega` *contient une connaissance plus générale que* `gamma` *et la valeur* `FALSE` *sinon. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Nous supposons prédéfinie la fonction `minimisation(Omega:BASE):BASE;` telle que l'appel `minimisation(Omega)` renvoie une base de connaissances contenant les mêmes connaissances que $\rrbracket \Omega \llbracket$. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

# 7 Complétion d'une base de connaissances

La *complétion* d'une base de connaissances consiste à construire l'ensemble de toutes les connaissances qui peuvent être déduites d'une base donnée. Cette opération permet ensuite de réduire les coûts d'interrogation de la base.

**Déf. II.19 (Déduction de connaissances)** *Soient les connaissances $\gamma_1 = H_1 \vdash C_1$ et $\gamma_2 = H_2 \vdash C_2$, l'opérateur $\rhd$ de déduction de connaissances construit une base notée $\gamma_1 \rhd \gamma_2$ et définie par :*

$$\gamma_1 \rhd \gamma_2 = \{(H_1 \setminus \{f\}) \cup H_2 \vdash C_1 \cup (C_2 \setminus \{f\}) \mid f \in H_1 \cap C_2\}$$

*Notons que $\gamma_1 \rhd \gamma_2 = \emptyset$ si $H_1 \cap C_2 = \emptyset$.*

*L'ensemble des connaissances qui peuvent être déduites d'une base $\Omega$ est l'ensemble des connaissances construites par application d'un nombre quelconque de fois de l'opérateur $\rhd$ à partir des connaissances de $\Omega$.*



**Question II.21** *Appliquer l'opérateur $\rhd$ sur les connaissances suivantes (ne donner que les résultats différents de $\emptyset$) :*

- $\gamma_1 = \{a,b\} \vdash \{c,d\}$ ;
- $\gamma_2 = \{b,c\} \vdash \{e\}$ ;
- $\gamma_3 = \{e\} \vdash \emptyset$ ;
- $\gamma_4 = \emptyset \vdash \{b,c\}$.

**Question II.22** *Soient les connaissances $\gamma_1 = H_1 \vdash C_1$ et $\gamma_2 = H_2 \vdash C_2$, montrer que si $\mid H_1 \cap C_2 \mid > 1$ alors $\gamma_1 \rhd \gamma_2$ ne contient que des tautologies. Que peut-on en conclure ?*

La composition d'une connaissance $\gamma$ et d'une base $\Omega$ consiste à appliquer l'opérateur de déduction $\rhd$ sur $\gamma$ et sur chaque connaissance de $\Omega$.

Nous souhaitons écrire une fonction `composition` qui prend en paramètre une connaissance $\gamma$ et une base $\Omega$ et qui construit une nouvelle base contenant les connaissances résultant de la composition de `gamma` avec chaque connaissance de la base `Omega`. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.

**Question II.23** *Décrire cette fonction* `composition` *et expliquer son algorithme selon le format présenté en page 15.*

**Question II.24** *Écrire en PASCAL cette fonction* `composition(gamma:CONNAISSANCE;Omega:BASE):BASE;` *telle que l'appel* `composition(gamma,Omega)` *renvoie une base contenant les connaissances résultant de la composition de* `gamma` *avec les connaissances de la base* `Omega`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Nous supposons prédéfinie la fonction `deduction(Omega:BASE):BASE;` telle que l'appel `deduction(Omega)` renvoie une base de connaissances qui contient les connaissances de la base `Omega` ainsi que le résultat des compositions deux à deux des connaissances de `Omega`. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

**Question II.25** *Soient deux connaissances $\gamma_1 = H_1 \vdash C_2$ et $\gamma_2 = H_2 \vdash C_2$ telles que $H_1 \cap C_2 = \{f\}$, montrer que les bases $\{\gamma_1,\gamma_2\} \cup \gamma_1 \rhd \gamma_2$ et $\{\gamma_1,\gamma_2\}$ sont équivalentes.*

**Question II.26** *Soit une base de connaissances quelconque $\Omega$, soit la suite $\{\Omega_i\}$ définie par :*

$$\begin{cases} \Omega_0 = \Omega \\ \Omega_{i+1} = \Omega_i \cup \bigcup\limits_{\gamma_i,\gamma_j \in {\Omega_i}^2} \gamma_i \rhd \gamma_j \end{cases}$$

*Nous admettrons que le résultat de la question II.25 peut être étendu à : $\forall i \in \mathbb{N}, \Omega_{i+1} \equiv \Omega_i$.*

1. *Montrer que la suite $\{\Omega_i\}$ est croissante ;*
2. *Montrer que : $\exists k \in \mathbb{N}, \Omega_{k+1} = \Omega_k$ (soit $l = \min\{k \in \mathbb{N} \mid \Omega_{k+1} = \Omega_k\}$, nous noterons alors $\overline{\Omega} = \Omega_l$ et nous appelerons $\overline{\Omega}$ la complétion de la base $\Omega$) ;*
3. *Montrer que $\overline{\Omega} \equiv \Omega$.*

**Question II.27** *Calculer la complétion $\overline{\Omega}$ de la base $\Omega$ contenant les connaissances suivantes :*

- $\gamma_1 = \{a,b\} \vdash \{c,d\}$ ;
- $\gamma_2 = \{b,c\} \vdash \{e\}$ ;
- $\gamma_3 = \{e\} \vdash \emptyset$.

L'élimination des tautologies et la minimisation de la base obtenue permettent ensuite de réduire la taille de la base et les coûts d'interrogation.

**Question II.28** *Éliminer les tautologies et minimiser la réponse que vous avez proposée pour la question précédente.*

Nous souhaitons écrire une fonction `completion` qui prend en paramètre une base $\Omega$ et qui construit une nouvelle base contenant les mêmes connaissances que $\overline{\Omega}$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.

**Question II.29** *Décrire cette fonction* `completion` *et expliquer son algorithme selon le format présenté en page 15.*

**Question II.30** *Écrire en PASCAL cette fonction* `completion(Omega:BASE):BASE;` *telle que l'appel* `completion(Omega)` *renvoie une base de connaissances contenant les mêmes connaissances que $\overline{\Omega}$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

# 8 Interrogation d'une base

Les connaissances contenues dans la base établissent un lien entre des faits « *hypothèses* » et des faits « *conclusions* ». Intuitivement, l'interrogation de la base consiste à :

- compléter la base (voir section 7), minimiser la base complétée (voir section 6) et en éliminer les tautologies (voir section 5) ;
- fournir une liste de faits « vrais » et une liste de faits « *faux* » ;
- intégrer ces faits dans la base complétée ;
- rendre la base obtenue minimale (voir section 6) pour obtenir la réponse à la requête.

Définissons ceci formellement :

**Déf. II.20 (Interrogation d'une base)** *Soit une base $\Omega$, la réponse à la requête composée des faits vrais $V$ et des faits faux $F$ est la base $\frac{\Omega}{V,F}$ définie par :*

$$\frac{\Omega}{V,F} = ]\!] \ \{ \ (H \setminus V) \vdash (C \setminus F) \ \mid \ H \vdash C \in \ ]\!] \overline{\Omega} [\![ \ [ \ \} \ [\!]$$

**Question II.31** *Soit la base de connaissances proposée à la question II.27, construire la réponse à la requête $V = \{b\}$,$F = \{d\}$.*

**Question II.32** *Montrer que* $]\,(\frac{\Omega}{V,F})\,[ \equiv \frac{\Omega}{V,F}$.



**Question II.33** *Montrer que* $\overline{(\frac{\Omega}{V,F})} \equiv \frac{\Omega}{V,F}$.

**Question II.34** *Écrire en PASCAL une fonction* `interrogation(V:FAITS;F:FAITS;Omega:BASE):BASE;` *telle que l'appel* `interrogation(V,F,Omega)` *renvoie une base de connaissances contenant les mêmes connaissances que $\frac{\Omega}{V,F}$. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

# Fin de l'énoncé