

# Éléments de logique pour l'informatique

## Exercices II

Uli Fahrenberg  
`uli@lmf.cnrs.fr`

*d'après* Christine Paulin

6 décembre 2025

### Table des matières

<b>1</b>	<b>Modèle de Herbrand</b>	<b>2</b>
<b>2</b>	<b>Formes normales</b>	<b>2</b>
<b>3</b>	<b>Forme clausale</b>	<b>3</b>
<b>4</b>	<b>Représentation des formules</b>	<b>4</b>
<b>5</b>	<b>Skolémisation</b>	<b>10</b>
<b>6</b>	<b>Systèmes de déduction</b>	<b>11</b>
<b>7</b>	<b>Systèmes de déduction 2</b>	<b>12</b>

# 1 Modèle de Herbrand

**Exercice 1.1** Le domaine de Herbrand est formé des termes clos (sans variable) construits à partir de la signature. Chaque symbole de fonction est interprété par lui-même.

Pour les formules suivantes : donner le domaine de Herbrand et dire si les formules sont satisfiables.

1.  $\forall x, (P(x) \wedge Q(x) \wedge (\neg P(a) \vee \neg Q(b)))$
2.  $\forall x, ((P(x) \vee Q(x)) \wedge \neg P(a) \wedge \neg Q(b))$
3.  $\forall x, (P(x) \wedge \neg P(f(x)))$
4.  $\forall xyz, (R(x, s(x)) \wedge (R(x, y) \wedge R(y, z) \Rightarrow R(x, z)) \wedge \neg R(x, x))$

**Exercice 1.2** Pour chaque ensemble de formules suivant : donner la signature du langage, le domaine de Herbrand et dire si l'ensemble de formules est satisfiable ou non.

1.  $\{\forall x, (P(x) \vee Q(x) \vee R(x)); \neg P(a); \neg Q(b); \neg R(c)\}$
2.  $\{\forall x, P(x); \forall x, \neg Q(x); \forall x, (\neg P(f(x)) \vee Q(f(x)))\}$
3.  $\{\forall x, P(x); \forall x, \neg P(f(x)); \forall x, P(f(f(x))); \forall x, (\neg P(f(f(x))) \vee \neg P(x) \vee P(f(x)))\}$

# 2 Formes normales

**Exercice 2.1** On reprend le type Ocaml vu en cours pour représenter les formules propositionnelles

```
type connecteur = Impl | Et | Ou
type form = Var of int | Bot | Top
           | Neg of form | Bin of form * connecteur * form
```

Compléter les fonctions suivantes afin d'implémenter les fonctions `clauseb` et `fncb` qui testent respectivement si une formule est une clause et si une formule est en forme normale conjonctive.

```
let rec clauseb = function
  Var n ->
  | Bot
  | Top ->
  | Neg f ->
  | Bin(f, Or, g) ->
  | Bin(f, Et, g) ->
  | Bin(f, Impl, g) ->

let rec fncb = function
  Var n ->
  | Bot
  | Top ->
  | Neg f ->
  | Bin(f, Ou, g) ->
  | Bin(f, Et, g) ->
  | Bin(f, Impl, g) ->
```

**Exercice 2.2** *Formes normales.*

Dire si les formules suivantes sont des clauses, des formes normales conjonctives, des formes normales disjonctives ?

1.  $p \wedge \neg q$
2.  $p \vee \neg q$
3.  $\neg(p \vee q)$
4.  $p \wedge q \vee r$

**Exercice 2.3** *Formes normales à partir d'une table de vérité.*

Donner les formes normales conjonctives et disjonctives des formules  $P$  et  $Q$  dont les tables de vérité sont les suivantes :

	$a$	$b$	$c$	$P$
1	$V$	$V$	$V$	$V$
2	$V$	$V$	$F$	$F$
3	$V$	$F$	$V$	$V$
4	$V$	$F$	$F$	$F$
5	$F$	$V$	$V$	$V$
6	$F$	$V$	$F$	$F$
7	$F$	$F$	$V$	$V$
8	$F$	$F$	$F$	$V$

	$a$	$b$	$c$	$Q$
1	$V$	$V$	$V$	$V$
2	$V$	$V$	$F$	$F$
3	$V$	$F$	$V$	$V$
4	$V$	$F$	$F$	$V$
5	$F$	$V$	$V$	$F$
6	$F$	$V$	$F$	$V$
7	$F$	$F$	$V$	$F$
8	$F$	$F$	$F$	$V$

**Exercice 2.4** *Formes normales.*

1. A l'aide des lois de de Morgan et des lois de distributivité, mettre sous forme normale conjonctive et sous forme normale disjonctive les formules suivantes :
  - (a)  $(\neg p \vee \neg q \vee r) \Rightarrow (r \vee s)$
  - (b)  $p \Rightarrow ((\neg q \vee r) \Rightarrow s)$
  - (c)  $\neg(p \vee \neg q) \wedge (q \Rightarrow (p \vee r))$
  - (d)  $(p \Rightarrow q) \wedge \neg q \wedge \neg(q \Rightarrow p)$
  - (e)  $(\neg p \Rightarrow q) \wedge (q \Rightarrow p)$
2. Donner pour chacune des formules ci-dessus un modèle.
3. Utiliser les exemples précédents pour répondre aux questions suivantes
  - Est-il possible d'avoir une formule qui soit simultanément en forme normale conjonctive et disjonctive ?
  - Est-ce vrai que toutes les formes normales conjonctives d'une formule donnée ont le même nombre de clauses ?
  - Est-ce vrai si on ne permet pas de répétitions de clauses dans une forme normale conjonctive, ni de clause qui se simplifie trivialement ?

### 3 Forme clausale

**Exercice 3.1** *Propriétés théoriques des clauses*

On se donne  $n$  variables propositionnelles  $x_1, \dots, x_n$ . On pourra commencer par regarder le cas  $n = 3$ . On rappelle qu'une clause est une disjonction de littéraux avec deux cas "limites" à savoir la clause vide  $\perp$  et la clause triviale  $\top$ .

1. Combien de fonctions booléennes différentes y-a-t-il en utilisant seulement ces variables ?
2. Montrer que si une clause  $C$  contient deux littéraux qui portent sur la même variable propositionnelle alors il existe une clause  $C'$  plus courte (avec moins de littéraux) qui a la même table de vérité. On appellera dans la suite clause réduite, une telle clause.
3. Combien de clauses différentes réduites peut-on construire qui n'ont pas la même table de vérité ?
4. Soit deux clauses  $C$  et  $D$  réduites telles que  $C \models D$  et  $D \neq \top$ , comment se comparent l'ensemble des littéraux de  $C$  et ceux de  $D$  ?
5. Soit une clause  $C$  réduite qui comporte des littéraux sur les  $n$  variables. Combien de lignes de la table de la vérité ont la valeur vrai ?

6. Soit une clause  $C$  réduite qui comporte  $p$  des  $n$  variables. Combien de lignes de la table de la vérité ont la valeur vrai ?

### Exercice 3.2 Mise en forme clausale efficace

On a vu en cours que la mise en forme clausale d'une formule  $P$  (qui correspond à la forme normale conjonctive) pouvait aboutir à faire grossir exponentiellement la taille de la formule. On montre ici que si on s'intéresse juste à savoir si  $P$  est satisfiable alors on peut trouver un ensemble de clauses qui est satisfiable exactement lorsque  $P$  est satisfiable et dont la taille est linéaire en la taille de  $P$ .

Le principe de la décomposition est le suivant. On part d'une formule  $P$  qui est en forme normale de négation et qui ne contient que des conjonctions et des disjonctions. On construit un ensemble  $\text{clause}(P)$  qui contient éventuellement plus de variables.

On procède par récurrence sur la structure de la formule.

- Si  $P = P_1 \wedge P_2$  alors il suffit de décomposer  $P_1$  et  $P_2$  en clauses et de les rassembler :  
 $\text{clause}(P) = \text{clause}(P_1) \cup \text{clause}(P_2)$
- Si  $P$  est un littéral ou une disjonction de littéraux alors  $P$  est déjà une clause et  $\text{clause}(P) = \{P\}$
- Si  $P = P_1 \vee P_2$  mais que  $P$  n'est pas une disjonction de littéraux, cela signifie que  $P$  contient au moins une conjonction. La formule  $P$  est logiquement équivalente à  $l_1 \vee \dots \vee l_n \vee (R_1 \wedge Q_1) \vee \dots \vee (R_p \wedge Q_p)$ . On remplace chaque conjonction  $R_i \wedge Q_i$  par une nouvelle variable propositionnelle  $x_i$  et on ajoute des formules qui disent que  $x_i \Rightarrow (R_i \wedge Q_i)$ .

$$\begin{aligned} \text{clause}(P) = & \{l_1 \vee \dots \vee l_n \vee x_1 \vee \dots \vee x_p\} \\ & \cup \text{clause}(\neg x_1 \vee R_1) \cup \text{clause}(\neg x_1 \vee Q_1) \cup \dots \\ & \cup \text{clause}(\neg x_p \vee R_p) \cup \text{clause}(\neg x_p \vee Q_p) \end{aligned}$$

1. appliquer la méthode précédente à la formule

$$P \stackrel{\text{def}}{=} (a_1 \wedge \neg b_1) \vee (a_2 \wedge b_2) \vee (\neg a_3 \wedge (b_3 \vee (a_4 \wedge b_4))) \vee \neg a_5.$$

2. Expliquer pourquoi la procédure précédente termine
3. (optionnel) Montrer que si  $\text{clause}(P)$  est vraie pour une interprétation  $I$  alors  $P$  est aussi vraie pour cette interprétation et si  $P$  est vraie pour une interprétation  $I$  alors il existe une interprétation  $I'$  qui coïncide avec  $I$  sur les variables de  $P$  et pour laquelle  $\text{clause}(P)$  est vraie.

## 4 Représentation des formules

### Exercice 4.1 Connecteur de Sheffer

On définit le connecteur de Sheffer noté  $|$  (barre de Sheffer, ou encore NAND) par :  $p | q \stackrel{\text{def}}{=} \neg(p \wedge q)$

1. Donner la table de vérité de la formule  $(p | q)$
2. Donner la table de vérité de la formule  $((p | q) | (p | q))$
3. On veut maintenant exprimer les connecteurs usuels en utilisant la barre de Sheffer, et rien qu'elle.
  - (a) Donner la table de vérité de la formule  $(p | p)$  et en déduire que le connecteur  $\neg$  peut être défini en n'utilisant que la barre de Sheffer.
  - (b) Trouver une formule équivalente à  $p \vee q$ , qui n'utilise que la barre de Sheffer (éventuellement plusieurs fois).
  - (c) Trouver une formule équivalente à  $p \Rightarrow q$ , qui n'utilise que la barre de Sheffer (éventuellement plusieurs fois).

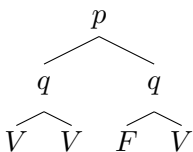
4. Définir par des équations récursives une fonction **shef** qui étant donnée une formule propositionnelle construite sur les connecteurs habituels, la transforme en une formule qui ne contient que des variables propositionnelles et la barre de Sheffer.
5. Donner le résultat de **shef**( $x \vee (y \vee z)$ ).
6. Si  $n$  est le nombre de connecteurs logiques dans la formule  $P$ , quel est l'ordre de grandeur du nombre de barres de Sheffer dans la formule **shef**( $P$ ) en fonction de  $n$ ? On pourra étudier le nombre de connecteurs dans la formule **shef**( $x_p \vee \dots \vee x_1 \vee x_0$ ) en fonction de  $p$ .
7. Soit  $C$  et  $D$  deux formules, montrer que

$$C, A, B \models D \quad \text{si et seulement si} \quad C \models (A \mid B) \vee D$$

**Exercice 4.2** *Connecteur propositionnel IF.* On considère des formules construites à partir de variables propositionnelles, des constantes  $\top$  et  $\perp$  et d'un seul connecteur logique ternaire **IF**( $P, Q, R$ ). On appelle  $\text{PROP}_{\text{IF}}$  l'ensemble des formules ainsi construites. La valeur de **IF**( $P, Q, R$ ) est égale à la valeur de  $Q$  lorsque la valeur de  $P$  est vraie et à la valeur de  $R$  sinon.

1. Donner des formules équivalentes à  $\neg P$ ,  $P \wedge Q$ ,  $P \vee Q$  et  $P \Rightarrow Q$  qui n'utilisent que le connecteur **IF**.
2. Décrire par des équations récursives une fonction **valIF** qui étant donnée une interprétation des variables propositionnelles  $I$  et une formule  $P$  de  $\text{PROP}_{\text{IF}}$  calcule la valeur de vérité de cette formule.
3. Montrer que les formules **IF**(**IF**( $P, Q, R$ ),  $S, T$ ) et **IF**( $P, \text{IF}(Q, S, T), \text{IF}(R, S, T)$ ) sont équivalentes.
4. Décrire par des équations récursives une fonction de normalisation **norm** qui étant donnée une formule  $P$  de  $\text{PROP}_{\text{IF}}$  renvoie une formule équivalente dans laquelle toute expression de la forme **IF**( $P, Q, R$ ) est telle que  $P$  est une variable propositionnelle (on dira que la formule est en forme normale).  
On commencera par introduire une fonction **IFn** qui étant données trois formules  $P, Q$  et  $R$  de  $\text{PROP}_{\text{IF}}$ , en supposant que  $Q$  et  $R$  sont déjà en forme normale, calcule une formule *en forme normale* équivalente à **IF**( $P, Q, R$ ).
5. Soit la formule propositionnelle  $A \stackrel{\text{def}}{=} ((p \Rightarrow q) \Rightarrow p) \Rightarrow p$ .
  - (a) Transformer la formule en **IF**-expression.
  - (b) Donner une forme normale équivalente.
  - (c) Représenter la formule en forme normale comme un arbre binaire dont les nœuds internes portent une étiquette de la forme **IF**( $x$ ) avec  $x$  la variable propositionnelle qui apparaît en condition.
  - (d) En utilisant la représentation précédente, montrer que  $A$  est valide.
6. En déduire une méthode pour décider si une formule de  $\text{PROP}_{\text{IF}}$  est valide.

**Exercice 4.3** *Arbre de décision binaire.* Un *arbre de décision binaire* (BDT) est un arbre binaire dont les nœuds sont étiquetés par des variables propositionnelles et les feuilles sont formées des constantes booléennes  $V$  ou  $F$ .



Les variables propositionnelles sont ordonnées et on impose de plus que les variables propositionnelles des fils sont strictement plus grandes que celle du père.

L'arbre peut se voir comme la table de vérité d'une formule propositionnelle (ou encore une fonction booléenne) : lorsque l'on passe un nœud étiqueté par la variable  $x$ , le sous arbre gauche correspond au cas où  $x$  a la valeur  $V$  et le sous-arbre droit au cas où  $x$  a la valeur  $F$ , dans l'exemple :

$p$	$q$	
$V$	$V$	$V$
$V$	$F$	$V$
$F$	$V$	$F$
$F$	$F$	$V$

On peut représenter l'arbre en utilisant un constructeur **Bool** pour les feuilles, associé à une valeur booléenne  $V$  ou  $F$  et un constructeur à trois arguments **IF**( $x, P, Q$ ) avec  $x$  une variable propositionnelle et  $P$  et  $Q$  des arbres de décision binaire.

Dans l'exemple cela donne **IF**( $p, \mathbf{IF}(q, \mathbf{Bool}(V), \mathbf{Bool}(V)), \mathbf{IF}(q, \mathbf{Bool}(F), \mathbf{Bool}(V))$ )

1. Donner les équations récursives qui définissent la valeur booléenne  $\mathbf{vald}(I, t)$  de l'arbre de décision binaire  $t$  pour une interprétation  $I$  des variables propositionnelles.

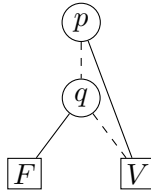
$$\begin{aligned} \mathbf{vald}(I, \mathbf{Bool}(b)) &= \\ \mathbf{vald}(I, \mathbf{IF}(x, P, Q)) &= \dots \mathbf{vald}(I, P) \dots \mathbf{vald}(I, Q) \dots \end{aligned}$$

2. Donner les équations récursives qui définissent une formule **form**( $t$ ) avec  $t$  un BDT, qui a la même table de vérité et qui de plus est en forme normale de négation et n'utilise que des littéraux, des conjonctions et des disjonctions.
3. Définir par des équations récursives une fonction **notd** qui prend en argument un BDT  $t$  qui représente une formule  $A$  et renvoie un autre BDT qui représente la formule  $\neg A$ .
4. On généralise la construction précédente au cas d'une opération binaire **op** sur les formules propositionnelles (comme la conjonction, l'implication).  
Définir par des équations récursives une fonction **opd** qui prend en argument deux BDT  $t$  et  $u$  qui représentent respectivement les formules  $A$  et  $B$  et renvoie un autre BDT qui représente la formule  $A \mathbf{op} B$ . On fera attention à respecter la contrainte sur l'ordre des variables propositionnelles dans les BDT.
5. Que peut-on dire des feuilles d'un BDT qui correspond à une formule valide ? satisfiable ? insatisfiable ?
6. Proposer un algorithme pour trouver un modèle d'une formule.

**Exercice 4.4** *Diagramme de décision binaire.* Les arbres de décision binaire ont deux défauts : des arbres différents peuvent représenter des formules équivalentes (par exemple  $\mathbf{IF}(x, P, P) \equiv P$ ) et leur taille est en général exponentielle en le nombre de variables propositionnelles.

Un *diagramme de décision binaire* (BDD) est obtenu à partir d'un arbre de décision en assurant de plus les conditions suivantes :

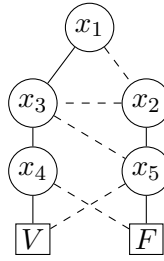
1. *réduction* : aucun nœud n'a deux fils identiques, cela correspond au fait que l'expression  $\mathbf{IF}(p, A, A)$  est logiquement équivalente à  $A$  et donc on peut remplacer un nœud qui a deux fois le même fils par ce fils ;
2. *partage* : l'arbre est transformé en graphe orienté acyclique (DAG), c'est-à-dire que deux sous-arbres identiques sont partagés. Dans un graphe, il n'y a plus de notion de sous-arbre gauche et de sous-arbre droit : on distingue par un arcs plein le cas où la variable vaut  $V$  et par un arc pointillé le cas où la variable vaut  $F$ .



3. *variables ordonnées* : comme dans les BDT, on va demander que les variables soient ordonnées : la variable d'un fils est toujours strictement supérieure à la variable du père, on parle alors de diagramme de décision binaire *ordonné* (ou OBDD).

Pour construire un OBDD à partir d'une formule propositionnelle  $P$ , il suffit de partir de la plus petite variable  $x$  qui apparaît dans la formule et de construire récursivement l'OBDD  $P_V$  correspondant à la formule  $P[x \leftarrow \top]$  pour la branche "vrai" et l'OBDD  $P_F$  correspondant à la formule  $P[x \leftarrow \perp]$  pour la branche "faux". Si les deux OBDDs sont les mêmes alors  $P_V$  est le résultat attendu, sinon on introduit le nœud  $x$  avec un arc plein vers  $P_V$  et un arc pointillé vers  $P_F$ .

1. Construire les OBDDs des formules  $(z \Leftrightarrow t)$  puis  $(x \Leftrightarrow y) \wedge (z \Leftrightarrow t)$  en prenant comme ordre  $x < y < z < t$ .
  2. En partant des feuilles, annoter chaque nœud dans les deux OBDD précédents par une formule équivalente qui n'utilise que des littéraux et les symboles  $\vee$  et  $\wedge$ .
  3. Montrer que deux formules sont équivalentes si et seulement si elles sont représentées par le même OBDD. En déduire comment, étant donné un OBDD, on peut savoir si la formule associée est valide, satisfiable ou insatisfiable.
  4. Pour représenter un OBDD, on associe un numéro à chaque sommet du graphe et on utilise une table  $G$  qui permet de stocker les variables et les arêtes. On dispose des opérations suivantes :
    - **noeud**( $G, n$ ) : teste si  $n$  est un sommet du graphe qui correspond à un nœud interne ;
    - **feuille**( $G, n$ ) : teste si  $n$  est un sommet du graphe qui correspond à une feuille ( $V$  ou  $F$ ) ;
    - pour un nœud interne : **var**( $G, n$ ) donne la variable propositionnelles associée, **vrai**( $G, n$ ) est le numéro du sommet qui correspond au cas où la variable est vraie et **faux**( $G, n$ ) est le numéro du sommet qui correspond au cas où la variable est fausse ;
    - pour une feuille : **valeur**( $G, n$ ) renvoie un booléen correspondant à la valeur  $V$  ou  $F$  de la feuille.
- (a) Exprimer à l'aide des fonctions ci-dessus, les propriétés sur la table  $G$  qui assurent que le graphe correspond bien à un OBDD : partage maximal, réduction et ordre sur les variables propositionnelles.
  - (b) Soit un graphe  $G$  correspondant à un OBDD et un sommet  $n$ , comment transformer la fonction **form** de la question 2 de l'exercice 4.3 en une fonction **form**( $G, n$ ) qui calcule la formule logique associée ? comment faire pour exploiter le partage et ne pas recalculer deux fois le même résultat ?
  - (c) Expliquer comment construire une fonction **if** qui étant donnés un graphe  $G$  correspondant à un OBDD, une variable propositionnelle  $x$  et deux sommets  $n$  et  $m$  de  $G$ , renvoie un graphe  $G'$  et un nœud  $p$ , tel que **form**( $G', p$ )  $\equiv$  **IF**( $x, \text{form}(G, n), \text{form}(G, m)$ ) et **form**( $G', k$ ) = **form**( $G, k$ ) pour tous les sommets de  $G$  ( $G'$  contient possiblement un sommet de plus que  $G$ , mais ne modifie pas les sommets de  $G$ ). On utilisera une table  $T$  qui permet de retrouver si une formule est déjà représentée dans le graphe : il suffit de donner le nom de la variable et les sommets correspondant au cas vrai et au cas faux, si le nœud apparaît dans le graphe, la table donnera le sommet correspondant.
  - (d) On veut compter le nombre de modèles d'une formule représentée par un OBDD.
    - i. Soit l'OBDD suivant sur les variables propositionnelles  $x_1, x_2, x_3, x_4, x_5$  :



annoter chaque nœud  $n$  du graphe pour donner le nombre d'interprétations concernant les variables plus grandes que celle du nœud  $n$  qui satisfont la formule associée à  $n$ .

- ii. On suppose donnée une fonction **indice** qui calcule pour chaque sommet de l'OBDD un entier : si le sommet est un nœud, l'indice est la position de la variable correspondante dans l'ordre (la plus petite variable a le numéro 1) et si le sommet est une feuille, l'indice est  $n + 1$  avec  $n$  le nombre total de variables.  
Construire une fonction **nbsat** qui étant donné un OBDD représenté par un graphe  $G$  et un sommet  $n$ , retourne le nombre de modèles de la formule en ne considérant que les variables plus grandes que celle du sommet.
- iii. Sachant que la variable racine d'un OBDD n'est pas forcément la plus petite variable, en déduire le nombre de modèles de la formule en considérant toutes les variables.

**Exercice 4.5** *Modélisation d'un sudoku.* On modélise sous forme propositionnelle un problème de Sudoku simplifié de taille 4.

1			4
	4	1	
2			3
4			

Le but du jeu est de compléter les cases avec les chiffres de 1 à 4 sans mettre deux fois le même chiffre sur une ligne, une colonne ou dans l'un des quatre cadrants.

Pour modéliser le problème, on introduit une notion de position sur le damier comme un couple  $(i, j)$  qui représente la case sur la  $i$ -ème ligne et la  $j$ -ème colonne. Pour chaque position  $p$  et chaque entier  $k$  entre 1 et 4, on introduit une variable propositionnelle pour représenter que la case à la position  $p$  contient la valeur  $k$ , on note cette variable  $x_p^k$ .

1. Combien y-a-t-il de telles variables propositionnelles ?
2. Ecrire sous une forme normale conjonctive des formules qui expriment les propriétés suivantes :
  - $A(p)$  : il y a au moins un chiffre à la position  $p$
  - $B(p)$  : il n'y a pas deux chiffres différents à la position  $p$
  - **valid1**( $p_1, p_2, p_3, p_4$ ) avec  $p_1, p_2, p_3, p_4$  4 positions distinctes : pour tout  $d \in \{1, 2, 3, 4\}$  il existe une position  $p \in \{p_1, p_2, p_3, p_4\}$  telle que  $x_p^d$  est vrai
  - $D(p, q)$  : les deux positions  $p$  et  $q$  (supposées différentes) n'ont pas la même valeur (pour tout  $d$ ,  $x_p^d$  et  $x_q^d$  ne sont pas vraies en même temps)
  - **valid2**( $p_1, p_2, p_3, p_4$ ) : les valeurs sur deux positions distinctes de  $p_1, p_2, p_3, p_4$  sont différentes (on utilisera  $D$  défini précédemment).
3. Le fait que les propriétés  $A(p)$  et  $B(p)$  sont vraies pour toutes les positions  $p$  implique que l'on peut parler de la fonction  $M$  qui associe à une position  $p$ , l'entier unique  $i$  tel que  $x_p^i = V$ .  
Si on se donne quatre positions distinctes  $p_1, p_2, p_3, p_4$  alors on s'intéresse à l'ensemble des valeurs  $X = \{M(p_1); M(p_2); M(p_3); M(p_4)\}$ .  
Montrer que les formules **valid1**( $p_1, p_2, p_3, p_4$ ) et **valid2**( $p_1, p_2, p_3, p_4$ ) sont deux manières d'exprimer que  $X$  contient exactement quatre éléments et donc sont équivalentes.
4. En utilisant les prédicats précédents, modéliser le problème de Sudoku comme un ensemble de formules en forme normale conjonctive. Combien votre modèle contient-il de clauses ?



5. Que faut-il changer au problème si on cherche à résoudre une grille initiale comme celle donnée en exemple.

#### Exercice 4.6 Propriété de graphes

L'objet de cet exercice est d'explorer quelques propriétés des graphes exprimées en logique du premier ordre.

Lorsqu'on étudie les *graphes en mathématiques ou en informatique*, on s'appuie sur la théorie des ensembles : un graphe est donné par deux ensembles, l'ensemble des sommets  $\mathcal{V}$  et l'ensemble des arêtes  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ . Pour deux sommets  $x$  et  $y$ , on a  $(x, y) \in \mathcal{E}$  si et seulement si il existe une arête qui part du sommet  $x$  et arrive au sommet  $y$ . On peut exprimer les propriétés des graphes en utilisant tous les outils disponibles en théorie des ensembles (entiers, fonctions, égalité etc.) et on peut également en théorie des ensembles quantifier sur un graphe quelconque.

La théorie des graphes en *logique du premier ordre* repose sur un langage beaucoup plus pauvre dans lequel on considère les propriétés d'un seul graphe. Pour cela on introduit une signature avec deux symboles de prédicat d'arité 2 : un symbole  $E$  qui sert à caractériser les arêtes et l'égalité  $=$ . On note  $\mathcal{G}$  l'ensemble des axiomes de la théorie de l'égalité associés à ce langage.

Une interprétation  $I$  de cette théorie est donnée par un domaine  $\mathcal{D}$  et des interprétations  $(E_I, =_I)$  des deux symboles de prédicat. Les interprétations  $E_I$  et  $=_I$  sont des relations binaires sur  $\mathcal{D}$  donc des sous-ensembles de  $\mathcal{D} \times \mathcal{D}$ . On va ici ne considérer que des modèles dits *équationnels* dans lesquels le symbole d'égalité de la logique est interprété par l'égalité (mathématique) de  $\mathcal{D}$  (si le symbole d'égalité est interprété par une relation d'équivalence plus générale, il faut se placer dans l'espace quotient  $\mathcal{D}/=_I$  pour retrouver un modèle équationnel). L'interprétation  $I$  correspond à un graphe (au sens mathématique) dont les sommets sont les éléments de  $\mathcal{D}$  et les arêtes correspondent à l'ensemble  $E_I$ . Réciproquement tout graphe  $\mathcal{V}, \mathcal{E}$  définit une interprétation de la théorie des graphes avec pour domaine  $\mathcal{V}$ . L'interprétation du symbole  $E$  est l'ensemble des arêtes  $\mathcal{E}$  et l'interprétation du symbole d'égalité est l'égalité sur  $\mathcal{V}$ .



Dans les questions qui suivent, lorsqu'il est demandé d'écrire une formule logique représentant une certaine propriété mathématique des graphes, il s'agit de l'exprimer *en utilisant uniquement les symboles de prédicats  $E$  et  $=$  de la théorie des graphes et les connecteurs et quantificateurs logiques* (les quantifications portent uniquement sur des variables qui représentent un sommet quelconque), aucune autre notation "mathématique" n'est autorisée.

1. Traduire en formule logique de la théorie des graphes les propriétés suivantes :
  - (a) De tout sommet qui a une arête qui boucle sur lui-même, il part également une arête sur un sommet différent.
  - (b) Il n'y a pas de sommet isolé (un sommet est isolé s'il n'y a aucune arête qui part ou qui arrive sur ce sommet).
  - (c) Il existe 3 sommets distincts reliés deux à deux par des arêtes (le sens de l'arête est indifférent)
2. Traduire en langue naturelle (en utilisant le vocabulaire usuel des graphes) les propriétés logiques suivantes
  - (a)  $\forall x y, E(x, y) \Rightarrow \neg E(y, x)$
  - (b)  $\forall x, \exists y z, E(x, y) \wedge E(x, z)$
  - (c)  $\exists x y, \neg E(x, y) \wedge \neg E(y, x)$
3. On s'intéresse à la propriété d'existence d'un chemin entre deux sommets du graphe. D'un point de vue mathématique, on dira qu'il existe un chemin de taille  $n$  entre deux sommets  $x$  et  $y$  s'il existe des sommets  $x_0, \dots, x_{n+1}$  avec  $x_0 = x, x_{n+1} = y$  et pour  $i = 0 \dots n$ ,  $(x_i, x_{i+1}) \in \mathcal{E}$ . Si une formule  $P$  contient une variable libre  $x$ , on pourra convenir de la noter  $P[x]$ . On pourra alors utiliser la notation  $P[t]$  pour représenter la formule  $P[x \leftarrow t]$  obtenue en remplaçant les occurrences libres de la variable  $x$  dans  $P$  par le terme  $t$ .

- (a) Reprendre la définition mathématique pour expliciter à quelle condition il existe un chemin de longueur 0 entre  $x$  et  $y$ .  
Définir une formule logique  $C_0[x, y]$  paramétrée par deux variables libres  $x$  et  $y$  qui est vraie si et seulement si  $x$  et  $y$  sont deux sommets reliés par un chemin de longueur 0.
  - (b) On suppose qu'on a construit une formule  $C_n[x, y]$  paramétrée par deux variables libres  $x$  et  $y$  qui est vraie si et seulement si  $x$  et  $y$  sont deux sommets reliés par un chemin de longueur  $n$ . Utiliser  $C_n[x, y]$  (ou des instances de cette formule) pour construire une formule  $C_{n+1}[x, y]$  paramétrée par deux variables libres  $x$  et  $y$  qui est vraie si et seulement si  $x$  et  $y$  sont deux sommets reliés par un chemin de longueur  $n + 1$ .
  - (c) Pour un entier quelconque  $n \in \mathbb{N}$ , la formule  $C_n[x, y]$  est bien une formule de la théorie des graphes.  
Expliciter cette formule dans le cas particulier de  $n = 2$  (c'est-à-dire donner la définition de  $C_2[x, y]$  en n'utilisant que les symboles  $E, =$  et les connecteurs et quantificateurs logiques).
4. On suppose qu'il existe une formule du premier ordre  $C[x, y]$  dans le langage de la théorie des graphes qui est vraie dans un modèle si et seulement si il existe un chemin (de longueur quelconque) entre les sommets correspondant à  $x$  et  $y$ . On ajoute deux constantes  $a, b$  au langage, qui représentent donc des sommets.
- (a) Montrer que tout sous-ensemble fini de l'ensemble des formules  $\mathcal{G} \cup \{\neg C_n[a, b] | n \in \mathbb{N}\} \cup \{C[a, b]\}$  est satisfiable.
  - (b) Montrer que l'ensemble des formules  $\mathcal{G} \cup \{\neg C_n[a, b] | n \in \mathbb{N}\} \cup \{C[a, b]\}$  est insatisfiable.
5. On admet dans cette question le théorème de compacité pour les formules de la logique du premier ordre. C'est-à-dire qu'un ensemble  $\mathcal{A}$  quelconque de formules est satisfiable si et seulement si tout *sous-ensemble fini* de formules de  $\mathcal{A}$  est satisfiable.  
En appliquant ce théorème à l'ensemble  $\mathcal{G} \cup \{\neg C_n[a, b] | n \in \mathbb{N}\} \cup \{C[a, b]\}$ , la question précédente montre une contradiction. Que peut-on en déduire par rapport aux hypothèses faites ?

## 5 Skolémisation

**Exercice 5.1** Mettre sous forme clausale l'ensemble de formules  $E = \{F_1, F_2, F_3\}$  où

1.  $F_1 = \forall x, (p(x) \Rightarrow \exists y, \forall x, q(x, y))$
2.  $F_2 = ((\exists x, (p(x) \Rightarrow r(x))) \vee \forall y, p(y)) \wedge \forall x, \exists y, (r(y) \Rightarrow p(x))$
3.  $F_3 = ((\forall x, (p(x) \Rightarrow \exists y, q(y))) \Rightarrow \exists z, r(z)) \Rightarrow \exists u, s(u)$

**Exercice 5.2** Soit la formule  $A \stackrel{\text{def}}{=} (\exists x, \neg P(x)) \Rightarrow ((\forall x, P(x) \vee Q(x)) \Rightarrow \forall x, Q(x))$

1. Mettre la formule  $A$  en forme normale de négation (les négations ne portent que sur les prédicats et il n'y a plus de connecteur d'implication).
2. Soit  $B$  la formule obtenue à la question précédente, dire si les affirmations suivantes sont vraies ou fausses
  - $A \models B$
  - $B \models A$
  - $A$  est satisfiable si et seulement si  $B$  l'est.
3. Eliminer dans  $B$  les quantificateurs existentiels ( $\exists$ ) en utilisant la skolémisation et mettre les quantificateurs universels ( $\forall$ ) en tête de la formule.
4. Soit  $C$  la formule obtenue à la question précédente, dire si les affirmations suivantes sont vraies ou fausses

- $B \models C$
  - $C \models B$
  - $B$  est satisfiable si et seulement si  $C$  l'est.
5. La formule  $C$  est-elle vraie dans un modèle dont le domaine a exactement un élément ? Même question pour  $A$ , justifier vos réponses.
  6. La formule  $A$  est-elle satisfiable ?
  7. Mettre la formule  $\neg A$  en forme clausale.
  8. Indiquer le domaine et la base de Herbrand pour la forme clausale ainsi obtenue.
  9. La formule  $\neg A$  est-elle satisfiable ? La formule  $A$  est-elle valide ?

**Exercice 5.3** On rappelle que la propriété de “Skolemisation” dit que pour toute formule close  $A$ , il existe une formule  $B$  de la forme  $\forall x_1 \dots x_n, C$  avec  $C$  sans quantificateur telle que les trois propriétés suivantes soient vérifiées

1.  $B \models A$ ,
2. si  $A$  est satisfiable alors  $B$  est satisfiable
3. si  $A$  est insatisfiable alors il existe des substitutions closes  $\sigma_1, \dots, \sigma_k$  telles que  $\{C[\sigma_1], \dots, C[\sigma_k]\}$  soit insatisfiable.

A partir de ce résultat, et de la correspondance entre validité d’une formule et insatisfiabilité de sa négation, montrer le résultat dual suivant

1. Pour une formule  $P$  close quelconque, montrer qu’il existe une formule  $Q$  (appelée forme de Herbrand de  $P$ ) qui s’écrit  $\exists x_1 \dots x_n, R$  avec  $R$  sans quantificateur telle que les trois propriétés suivantes sont vérifiées
  - (a)  $P \models Q$ ,
  - (b) si  $Q$  est valide alors  $P$  est valide
  - (c) si  $P$  est valide alors il existe des substitutions closes  $\sigma_1, \dots, \sigma_k$  telles que la formule  $R[\sigma_1] \vee \dots \vee R[\sigma_k]$  soit valide.
2. Soit une formule existentielle  $Q$  de la forme  $\exists x_1 \dots x_n, R$  avec  $R$  sans quantificateur. Montrer que  $R[\sigma_1] \vee \dots \vee R[\sigma_k] \Rightarrow Q$  est une formule valide.
3. Soit la formule  $P \stackrel{\text{def}}{=} (T(a) \vee T(b)) \Rightarrow \exists x, T(x)$ , trouver la forme de Herbrand associée ainsi que des substitutions closes  $\sigma_1, \dots, \sigma_k$  telles que  $R[\sigma_1] \vee \dots \vee R[\sigma_k]$  soit valide. Peut-on trouver une solution à ce problème avec une seule substitution.

## 6 Systèmes de déduction

**Exercice 6.1** On s’intéresse ici à des formules qui contiennent le symbole d’équivalence  $p \Leftrightarrow q$ . On rappelle que  $p \Leftrightarrow q$  est défini comme  $(p \Rightarrow q) \wedge (q \Rightarrow p)$ . Dans la suite  $x, y$  et  $z$  désignent trois variables propositionnelles.

1. Donner la forme normale conjonctive des formules  $(x \Leftrightarrow y)$  et  $\neg(x \Leftrightarrow y)$ .
2. Construire la table de vérité de la formule  $(x \Leftrightarrow (y \Leftrightarrow z))$ . Cette formule est-elle valide ? satisfiable ?
3. On considère maintenant des séquents avec des formules contenant le symbole d’équivalence. Soit les deux règles dans lesquelles  $\Gamma$  et  $\Delta$  sont des ensembles de formules et  $p$  et  $q$  sont des formules :

$$\frac{\Gamma, p \vdash \Delta, q \quad \Gamma, q \vdash \Delta, p}{\Gamma \vdash \Delta, p \Leftrightarrow q} \quad \frac{\Gamma, p, q \vdash \Delta \quad \Gamma \vdash \Delta, p, q}{\Gamma, p \Leftrightarrow q \vdash \Delta}$$

- (a) Montrer que ces deux règles sont correctes (si une interprétation satisfait les deux prémisses alors elle satisfait la conclusion de la règle).
- (b) Montrer qu'elles sont inversibles (si une interprétation satisfait la conclusion de la règle alors elle satisfait les deux prémisses).
- (c) Construire en utilisant ces règles un arbre de preuve pour le séquent  $\vdash x \Leftrightarrow (y \Leftrightarrow (x \Leftrightarrow y))$ .
- (d) La formule  $x \Leftrightarrow (y \Leftrightarrow (x \Leftrightarrow y))$  est-elle valide ?

**Exercice 6.2** *Symbole d'équivalence* Soit  $\mathcal{X}$  un ensemble de variables propositionnelles.

1. Montrer que toute formule propositionnelle qui utilise les constructeurs logiques  $\top, \perp, \neg, \wedge, \vee, \Rightarrow$  est équivalente à une formule qui n'utilise que les connecteurs  $\neg, \Rightarrow$ .
2. On considère le connecteur  $\Leftrightarrow$  comme un connecteur primitif.
  - (a) Proposer des formules équivalentes à  $\perp$  et  $\top$  qui n'utilisent que les connecteurs  $\Leftrightarrow$  et  $\neg$ .
  - (b) Soit une formule  $A$  qui n'utilise possiblement que deux variables propositionnelles  $x, y$  et les connecteurs  $\Leftrightarrow$  et  $\neg$ . Montrer que l'ensemble des interprétations des variables  $x, y$  qui rendent la formule  $A$  vraie contient toujours un nombre pair d'éléments.
  - (c) En déduire qu'il existe des formules de la logique du premier ordre qui n'ont pas de formulation équivalente qui utilise uniquement les connecteurs  $\Leftrightarrow$  et  $\neg$ .

## 7 Systèmes de déduction 2

**Exercice 7.1** *Séquents valides.*

Donner pour chacun des séquents suivants la formule associée. Dire si les séquents sont valides.

1.  $\vdash (p \wedge q), (\neg p \wedge \neg q)$
2.  $p, \perp \vdash$
3.  $\top \vdash$
4.  $\vdash$

Appliquer les règles du système G à ces séquents, que constatez-vous ?

**Exercice 7.2** *Preuve dans le système G.* En utilisant des arbres de dérivation dans le système G pour les formules suivantes, dire si elles sont valides ou non :

1.  $\vdash ((p \wedge q) \Rightarrow r) \Rightarrow (\neg p \vee q) \Rightarrow (p \Rightarrow r)$
2.  $\vdash ((p \wedge q) \Rightarrow r) \wedge (\neg p \vee q) \Rightarrow p$

**Exercice 7.3** *Séquents et quantificateurs.* Soient les deux séquents

1.  $\exists x, \forall y, R(x, y) \vdash \forall z, \exists t, R(t, z)$
2.  $\forall y, \exists x, R(x, y) \vdash \exists t, \forall z, R(t, z)$

Dire pour chacun s'il est valide ou non. Construire les arbres de dérivation en expliquant si le séquent n'est pas valide pourquoi la preuve ne peut être complétée.

**Exercice 7.4** On rappelle qu'une relation binaire est totale si pour tout  $x$ , il existe un  $y$  tel que  $x$  est en relation avec  $y$ .

1. Utiliser le calcul des séquents pour prouver que toute relation binaire symétrique, transitive et totale est réflexive.
2. Montrer que la propriété que la relation est totale est nécessaire (on donnera un modèle de relation symétrique et transitive qui n'est pas réflexive).