IF107 Logique et preuve - TD "Logique des prédicats"
Département "Informatique", ENSEIRB-MATMECA 27 octobre 2025

Voir la page https://thor.enseirb-matmeca.fr/sites/if107-main/ pour toute information complémentaire sur cet enseignement.

1 Logique des prédicats

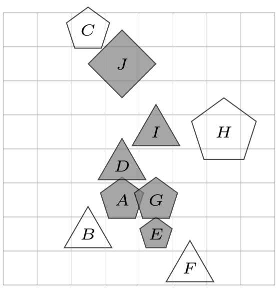
Exercice 1

1. Écrire une formule équivalente à $\exists x.\, p(x)$ sans quantificateur pour x à valeur dans $\{a, b, c\}$. Faire de même avec la formule $\forall. x\, p(x)$.
2. On considère maintenant que x est à valeurs dans un ensemble infini. Quel est le rôle des quantificateurs ?

Exercice 2

Soit l'ensemble de figures représentées sur la grille ci-dessous. Le pentagone E est une *petite* figure. Les figures H et J sont de *grandes* figures. Toutes les autres sont des figures de taille *moyenne*. Indiquez pour chaque formule ci-dessous si elle est vraie, sachant que :

— Le domaine des variables x et y est l'ensemble des figures ci-dessous ;
— les prédicats plus_petite, a_gauche, a_droite sont strictes. Par exemple, plus_petite(x, y) se lit "x est une figure *strictement* plus petite que y".



1. $\mathsf{triangle}(C)$
2. $\mathsf{pentagone}(F) \implies \mathsf{cercle}(H)$
3. $\exists x.\, \neg\, \mathsf{triangle}(x)$
4. $\forall x.\, \exists y.\, \mathsf{plus\_petite}(x, y)$
5. $\forall x.\, \exists y.\, \mathsf{a\_gauche}(x, y)$
6. $\exists x.\, \forall y.\, (y \neq x \implies \mathsf{a\_droite}(x, y))$
7. $\exists x.\, \forall y.\, \neg\, \mathsf{plus\_petite}(x, y)$
8. $\forall x.\, (\mathsf{pentagone}(x) \implies \exists y.\, (\mathsf{triangle}(y) \wedge \mathsf{a\_droite}(x, y)))$

Exercice 3

Indiquer la valeur de vérité des propriétés suivantes, et écrire leur négation.

1. $\forall x \in \mathbb{R}, (x > 3 \wedge x \leq -2)$
2. $\exists x, y, w \in \mathbb{R}, (x > y + z \wedge x - y \neq y - z \wedge -1 \leq z \leq 2)$
3. $\forall a \in \mathbb{N}, \exists b, c \in \mathbb{N}, (a = bc)$
4. $\forall a, b \in \mathbb{N}, \exists c \in \mathbb{N}, (a = bc)$
5. $\exists a \in \mathbb{N}, \forall b \in \mathbb{N}, \exists c \in \mathbb{N}.(a = bc)$
6. $\forall a, b \in \mathbb{R}, (a < b \implies \exists q \in \mathbb{Q}, a < q < b)$

Exercice 4

Écrire la négation des propositions suivantes :

1. Toutes les voitures de course sont rouges

2. Il existe un arbre dont les feuilles sont vertes
3. Pour tout $\epsilon > 0$, il existe $q \in \mathbb{Q}$ tel que $0 < q < \epsilon$
4. Dans toutes les filières d'enseignement, il existe un élève qui aime tous les cours

Exercice 5

1. Donner un modèle simple (ensemble E et relation pour les prédicats P et Q) tel que les propositions ci-dessous sont toutes les deux vraies :

$$\exists x \in E, \neg P(x) \tag{1}$$

$$\exists x \in E, (P(x) \wedge \neg Q(x)) \tag{2}$$

2. Pour tout modèle de E, P, et Q vérifiant (1) et (2), donner en justifiant, la valeur de vérité des deux propositions suivantes :
   a) $(\forall x \in E, P(x)) \implies (\forall x \in E, Q(x))$
   b) $\forall x \in E, (P(x) \implies Q(x))$

Exercice 6

Indiquer les occurrences libres et les occurrences liées de variables dans les formules ci-dessous. Indiquer si elles sont satisfiables, valides ou insatisfiables. Lorsqu'une formule est insatisfiable ou valide, le démontrer à l'aide de la technique des tableaux vue en cours (rappel : la formule doit être close et en forme normale négative).

1. $\forall x.\neg p(x) \vee \exists x.p(x)$
2. $p(x) \implies \forall y.p(y)$
3. $\exists x.p(x) \implies p(z)$
4. $\forall x.\forall y.(p(x) \iff \neg p(y))$
5. $\forall x.\exists y.p(x, y) \implies \exists y.\forall x.p(x, y)$
6. $\forall x.(p(x) \wedge q(x)) \iff \forall x.\, p(x) \wedge \forall x.q(x)$
7. $\forall x.p(x) \vee \forall x.q(x) \implies \forall x.(p(x) \vee q(x))$
8. $\forall x.(p(x) \vee q(x)) \implies \forall x.p(x) \vee \forall x\, q(x)$
9. $\exists x.(p(x) \wedge q(x)) \implies \exists x.p(x) \wedge \exists x.q(x)$
10. $\exists x.p(x) \wedge \exists x.q(x) \implies \exists x.(p(x) \wedge q(x))$
11. $\exists x.(p(x) \vee q(x)) \iff \exists x.p(x) \vee \exists x.q(x)$

2 Logique et bases de données relationnelles

Exercice 7

On souhaite modéliser différentes contraintes pour la réalisation d'équipes pour un tournoi sportif. On se donne les prédicats suivants :

`meme_club`(x,y)	x et y sont licenciés dans le même club sportif
`equipe`(x, y)	x et y appartienent à la même équipe
`femme`(x)	si x est une femme
$x = y$	si x et y sont la même personne
$x \neq y$	si x et y sont deux personnes distinctes

Exprimer les contraintes suivantes en logique des prédicats :

1. La relation `equipe` est (a) symétrique, (b) transitive et (c) réflexive
2. Tout joueur est l'équipier d'au moins un autre joueur
3. Les équipes sont constituées de joueurs du même club
4. Toute équipe est mixte (on considère deux genres : "femme" et "non femme")
5. Une équipe est constituée d'au moins trois joueurs

Exercice 8
On considère la base de données constituée des relations suivantes :
— produit$(idp, description)$
— client(idc, nom)
— vente(idp, idc)

Exprimer les requêtes suivantes en logique des prédicats en utilisant les relations ci-dessus.

1. nom_client(nom) qui est satisfaite si nom est un nom de client
2. ventes_shampooing(nom) qui est satisfaite si nom est le nom d'un client qui a acheté du shampooing
3. invendu$(description)$ qui est satisfaite si $description$ est un produit invendu

3 Logique et spécifications

Exercice 9
On considère une fonction :

```
int find(int * t, unsigned int n, int x)
```

qui cherche `x` dans le tableau `t` de taille `n`. La fonction `find` retourne la position i telle que `t`$[i] =$ `x` si `x` appartient à `t` et -1 sinon.

1. On formalise la spécification ci-dessus par la formule suivante où r est la valeur retournée par `find` :

   $$(r = -1) \vee (0 \leq r < n \wedge t[r] = x)$$

   Expliquer pourquoi cette formule n'exprime pas la même chose que la spécification informelle ci-dessus (indication : donner une implémentation de la fonction `find` qui satisfait la formule mais qui ne correspond pas à la spécification informelle, ou inversement).
2. Proposer une spécification formelle correcte pour `find`.

Exercice 10
On considère des tableaux t, t_1, et t_2 d'entiers à $N \in \mathbb{N}$ éléments. On note $t[i]$ l'élément d'indice i dans t avec $0 \leq i < N$. Écrire les propositions suivantes en logique des prédicats.

1. le tableau t représente la fonction $x \mapsto x^2$ pour x entier tel que $0 \leq x < N$
2. le tableau t est trié par ordre croissant
3. les valeurs du tableau t sont majorées par M

4. M est le maximum des valeurs du tableau t
5. la valeur x apparaît pour la première fois en position p dans le tableau t (on supposera $0 \leq p < N$)
6. les tableaux t_1 et t_2 contiennent les mêmes éléments et dans le même ordre
7. les tableaux t_1 et t_2 contiennent les mêmes éléments (on supposera que les éléments ne se répètent pas)
8. tout élément de t_1 est plus grand qu'un élément de t_2
9. tout élément de t_1 est la somme de deux éléments distincts de t_2

Exercice 11
Formaliser les spécifications des fonctions ci-dessous en logique des prédicats sous la forme :
— d'une *pré-condition* portant sur les paramètres, qui indique les valeurs de ceux-ci pour laquelle la fonction est définie (PRE) ;
— et d'une *post-condition* portant sur la valeur des paramètres et sur la valeur `r` retournée par la fonction (POST).

1. La fonction `max` qui retourne le maximum de `x` et de `y`

```
int max(int x, int y);
```

2. La fonction `fact` retourne la factorielle de `n`

```
int fact(unsigned int n);
```

3. La fonction `abs` retourne la valeur absolue de `x`

```
int abs(int x);
```

4. La fonction `gcd` retourne le plus grand commun diviseur à `a` et `b`

```
int gcd(unsigned int a, unsigned int b);
```

5. ★ La fonction `xchange` échange les valeurs de `*x` et `*y`.

```
void xchange(int * x, int * y);
```

6. ★ La fonction `sort` qui trie le tableau `t` à `n` éléments (ordre croissant)

```
void sort(int * t, unsigned int n);
```

7. ★ La fonction `array_eq` qui indique si les tableaux `t1` et `t2` à `n` éléments sont identiques (convention usuelle : 0 pour faux et tout autre valeur pour vrai)

```
int array_eq(int const * t1, int const * t2, unsigned int n);
```