

ERI5-IFON1 - Logique des prédicats

Frédéric Herbreteau (`frederic.herbreteau@bordeaux-inp.fr`)

15 septembre 2026

Limites de la logique propositionnelle

Expressivité : capacité d'un langage à formaliser certaines propositions

Logique propositionnelle : **articulation logique** de propositions entre elles : $\wedge$, $\vee$, $\neg$, $\implies$, ...

Mais elle **ne peut pas exprimer** des **relations** entre des objets

Exemple

- ▶ **p** : Certains étudiants assistent à tous les cours
- ▶ **q** : Aucun étudiant n'assiste à un cours inintéressant

Peut-on conclure que tous les cours sont intéressants sachant $p \wedge q$?

En logique des prédicats (1/2)

Exemple

- ▶ Certains étudiants assistent à tous les cours
- ▶ Aucun étudiant n'assiste à un cours inintéressant

Analyse syntaxique :

- ▶ Variables (objets du langage) : cours, étudiants, ...
- ▶ Prédicats (propriétés et relations entre les objets) : assiste, intéressant, ...
- ▶ Quantificateurs (**contexte**) : certains, aucun, tous, ...

Exemple

En logique des prédicats :

- ▶ $\exists x.(\textit{etudiant}(x) \wedge \forall y.(\textit{cours}(y) \implies \textit{assiste}(x, y)))$
- ▶ $\neg(\exists x.\exists y.(\textit{etudiant}(x) \wedge \textit{cours}(y) \wedge \textit{assiste}(x, y) \wedge \neg\textit{interessant}(y)))$

En logique des prédicats (2/2)

Exemple

- ▶ Certains étudiants assistent à tous les cours
- ▶ Aucun étudiant n'assiste à un cours inintéressant

Peut-on en conclure que tous les cours sont intéressants ?

Oui :

1. $\exists x.(\text{etudiant}(x) \wedge \forall y.(\text{cours}(y) \implies \text{assiste}(x, y)))$
2. $\neg(\exists x.\exists y.(\text{etudiant}(x) \wedge \text{cours}(y) \wedge \text{assiste}(x, y) \wedge \neg\text{interessant}(y)))$

Syntaxe

Signature

Définition 3.1 Soit $\mathcal{F}$ un ensemble de *symboles de fonctions*. À chaque symbole $f \in \mathcal{F}$ on associe son arité $a(f) \in \mathbb{N}$, c'est à dire le nombre de paramètres de f . Si $a(f) = 0$, alors f est un *symbole de constante*. On note C l'ensemble des constantes.

Soit $\mathcal{R}$ un ensemble de *symboles de relations*. À chaque symbole $R \in \mathcal{R}$ est associée son arité $a(R)$. Si $a(R) = 0$, alors R est une *variable propositionnelle*.

Une *signature* $S = (\mathcal{F}, \mathcal{R})$ est définie par un ensemble de symboles de fonctions et un ensemble de symboles de relations. $\blacklozenge$

Exemple

- ▶ signature : $(\emptyset, \{\textit{etudiant}, \textit{cours}, \textit{assiste}, \textit{interessant}\})$
 - ▶ $\exists x.(\textit{etudiant}(x) \wedge \forall y.(\textit{cours}(y) \implies \textit{assiste}(x, y)))$
 - ▶ $\neg(\exists x.\exists y.(\textit{etudiant}(x) \wedge \textit{cours}(y) \wedge \textit{assiste}(x, y) \wedge \neg\textit{interessant}(y)))$
 - ▶ mais pas : $\exists x.\exists y.\textit{interessant}(x, y)$
- ▶ arithmétique linéaire : $(\{+, 0, 1\}, \{\leq\})$
 - ▶ $\forall x.(x \leq x + 1)$
 - ▶ $\forall x.\exists y.(x + x \leq y \wedge \neg(y \leq x + x))$
 - ▶ $\forall x.\exists y.\textit{neg}(x = y)$
 - ▶ mais pas $(x^2 \leq y)$, ni $\forall x.\exists y.(\frac{x}{2} \leq y)$

Termes

Définition 3.2 Soit X un ensemble de *symboles de variables* et S une signature. L'ensemble des *termes* est défini par :

- les symboles de constantes C et de variables X sont des termes
- si $f \in \mathcal{F}$ est une fonction d'arité $a(f) = n$ et $t_1, \dots, t_n$ sont n termes, alors $f(t_1, \dots, t_n)$ est également un terme. $\blacklozenge$

Exemples :

- ▶ arithmétique linéaire : $(\{+, 0, 1\}, \{\leq\})$
 - ▶ $0 + 1 + 1$ (c'est à dire 2)
 - ▶ $x + 1$
 - ▶ $x + y + y + 1 + 1 + 1$ (c'est à dire $x + 2y + 3$)
 - ▶ mais pas $x \times x + y$
- ▶ arithmétique (non-linéaire) : $(\{\times, +, 0, 1\}, \{\leq\})$
 - ▶ $1 + 1 + 1$ (c'est à dire 3)
 - ▶ $x + y$
 - ▶ $x \times x + y$ (c'est à dire $x^2 + y$)

Formules

Définition 3.3 Soit $S = (\mathcal{F}, \mathcal{R})$ une signature et X un ensemble de symboles de variables. L'ensemble des *formules* de la logique des prédicats défini par :

- Si $R \in \mathcal{R}$ est un symbole de relation d'arité $a(R) = n$ et $t_1, \dots, t_n$ sont des termes sur S , alors $R(t_1, \dots, t_n)$ est une formule (dite *atomique*)
- Si ϕ et ψ sont des formules, alors $\neg\phi$, (ϕ) , $\phi \implies \psi$, $\phi \iff \psi$, $\phi \wedge \psi$ et $\phi \vee \psi$ sont des formules
- Si ϕ est une formule et $x \in X$ est une variable, alors $\exists x. \phi$ et $\forall x. \phi$ sont des formules.



Priorité :

$$\begin{array}{ccccccc} \neg & & & & & & \\ \forall & & \wedge & & \vee & & \implies \\ \exists & & & & & & \iff \end{array}$$

Exemples :

- ▶ signature : $(\emptyset, \{\textit{etudiant}, \textit{cours}, \textit{assiste}, \textit{interessant}\})$
 - ▶ $\exists x. (\textit{etudiant}(x) \wedge \forall y. (\textit{cours}(y) \implies \textit{assiste}(x, y)))$
 - ▶ $\neg(\exists x. \exists y. (\textit{etudiant}(x) \wedge \textit{cours}(y) \wedge \textit{assiste}(x, y) \wedge \neg \textit{interessant}(y)))$
- ▶ arithmétique linéaire : $(\{+, 0, 1\}, \{\leq\})$
 - ▶ $\forall x. (x \leq x + 1)$
 - ▶ $\forall x. \exists y. (x + x \leq y \wedge \neg(y \leq x + x))$
 - ▶ $\forall x. \exists y. \neg(x = y)$

Exercice 1

On se donne les prédicats ci-dessous pour modéliser différentes contraintes pour la réalisation d'équipes pour un tournoi sportif.

$\text{meme_club}(x,y)$	x et y sont licenciés dans le même club sportif
$\text{equipe}(x, y)$	x et y appartiennent à la même équipe
$\text{femme}(x)$	si x est une femme

NB : l'égalité = fait implicitement partie de toute signature

Exprimer les propositions 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
5. Une équipe est constituée d'au moins trois joueurs

Exercice 2

On considère la base de données constituée des relations suivantes :

- ▶ $\text{produit}(idp, description)$
- ▶ $\text{client}(idc, nom)$
- ▶ $\text{vente}(idp, idc)$

Exprimer les requêtes suivantes en logique des prédicats.

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

Sémantique

Portée, variable libre, variable liée

Définition 3.5 La *portée* d'un quantificateur $\exists x.\phi$ ou $\forall x.\phi$ est la formule ϕ .

Toutes les occurrences de la variable x dans ϕ sont nommées *occurrences liées*. Inversement, une occurrence d'une variable x qui n'est pas sous la portée d'un quantificateur est dite *occurrence libre*.

Une formule est *close* si toutes ses occurrences de variables sont liées. ◆

Exemples :

- ▶ $\forall x.(x > 0 \implies \exists y.x \geq y + 1)$
- ▶ $\forall x.x > 0 \implies \exists y.x \geq y + 1$
- ▶ $etudiant(x) \wedge \forall y.(cours(y) \implies assiste(x, y))$
- ▶ $\neg(\exists x.\exists y.(etudiant(x) \wedge cours(y) \wedge assiste(x, y) \wedge \neg interessant(y)))$

Contexte

La formule ci-dessous n'a pas de **valeur intrinsèque** :

$$etudiant(x) \wedge \forall y.(cours(y) \implies assiste(x, y))$$

Sa valeur dépend d'un **contexte** :

- ▶ qui fixe une **valeur** pour x (ex : $x \mapsto Pierre$)
- ▶ qui donne une **interprétation** aux symboles de fonctions et de prédicats (ex : $etudiant(Pierre) = 1$, $cours(logique) = 1$,
 $assiste(Pierre, logique) = 1$, $etudiant(Frederic) = 0$, $cours(wei) = 0$, ...)
- ▶ fixé pour les **variables liées** par le quantificateur (ex : pour toute valeur de y ...)

Dans un contexte, les **seules variables sont liées** et la formule a une **valeur de vérité**

$$etudiant(Pierre) \wedge \forall y.(cours(y) \implies assiste(Pierre, y))$$

Interprétation

Définition 3.6 Une *interprétation* d'une signature $S = (\mathcal{F}, \mathcal{R})$ est un triplet $I = (\mathbb{D}, \rho, \gamma)$ où :

- $\mathbb{D}$ est un ensemble non vide qui constitue le domaine des variables
- ρ associe à chaque symbole de relation $R \in \mathcal{R}$ d'arité $a(R) = n$ un sous-ensemble R_I de $\mathbb{D}^n$
- γ associe à chaque symbole de fonction $f \in \mathcal{F}$ d'arité $a(f) = n$, une fonction $f_I : \mathbb{D}^n \rightarrow \mathbb{D}$. En particulier, à chaque constante a est associée une valeur a_I . ♦

Exemple :

- ▶ signature : $(\emptyset, \{\textit{etudiant}, \textit{cours}, \textit{assiste}, \textit{interessant}\})$
 - ▶ $\textit{etudiant} = \{\textit{Pierre}, \textit{Mona}, \dots\}$
 - ▶ $\textit{cours} = \{\textit{logique}, \textit{prog C}, \dots\}$
 - ▶ $\textit{interessant} = \{\textit{logique}, \textit{prog C}, \dots\}$
 - ▶ $\textit{assiste} = \{(\textit{Pierre}, \textit{logique}), (\textit{Mona}, \textit{prog C}), \dots\}$
- ▶ arithmétique linéaire : $(\{+, 0, 1\}, \{\leq\})$
 - ▶ interprétation implicite

Valuation, valeur d'un terme

Définition 3.7 Soit $I = (\mathbb{D}, \rho, \gamma)$ une interprétation et X un ensemble de variables. Une *valuation* $v : X \rightarrow \mathbb{D}$ associe à chaque variable de X une valeur dans le domaine $\mathbb{D}$. ♦

Définition 3.8 La *valeur* $[t]_{I,v} \in \mathbb{D}$ d'un terme t dans une interprétation I et une valuation v est définie par :

- si t est une constante $a \in C$, alors t a la valeur de la constante a dans $S : [t]_{I,v} = a_I$
- si t est une variable $x \in X$, alors t a la valeur donnée à x par la valuation $v : [t]_{I,v} = v(x)$.
- enfin, si $t = f(t_1, \dots, t_n)$ est l'application d'un symbole de fonction f à n termes $t_1, \dots, t_n$, alors la valeur de t est celle de la fonction f_I associée à f par I appliquée aux valeurs de $t_1, \dots, t_n : [t]_{I,v} = f_I([t_1]_{I,v}, \dots, [t_n]_{I,v})$. ♦

Exemples :

► $v : x \mapsto 6, y \mapsto -3$

$$\begin{aligned}[2x + y + 4]_{I,v} &= [2x]_{I,v} + [y + 4]_{I,v} \\ &= 2[x]_{I,v} + [y]_{I,v} + 4 \\ &= 2v(x) + v(y) + 4 \\ &= 13\end{aligned}$$

Valeur d'une formule

Définition 3.9 Soit I une interprétation et v une valuation. La *valeur (de vérité)* d'une formule ϕ est donnée par $[\phi]_{I,v} \in \mathbb{B}$ définie par :

- si $\phi = R(t_1, \dots, t_n)$ est l'application d'un prédicat R à n termes $t_1, \dots, t_n$ alors la valeur de ϕ vaut $[\phi]_{I,v} = 1$ si les valeurs de $t_1, \dots, t_n$ sont en relation dans I : $([t_1]_{I,v}, \dots, [t_n]_{I,v}) \in R_I$, et $[\phi]_{I,v} = 0$ sinon ;
- si $\phi = \neg\psi$, alors $[\phi]_{I,v} = \overline{[\psi]_{I,v}}$;
- si $\phi = (\psi_1 \implies \psi_2)$, alors $[\phi]_{I,v} = \overline{[\psi_1]_{I,v}} + [\psi_2]_{I,v}$;
- si $\phi = (\psi_1 \iff \psi_2)$ alors $[\phi]_{I,v} = 1$ si $[\psi_1]_{I,v} = [\psi_2]_{I,v}$ et $[\phi]_{I,v} = 0$ sinon ;
- si $\phi = (\psi_1 \wedge \psi_2)$ alors $[\phi]_{I,v} = [\psi_1]_{I,v} \cdot [\psi_2]_{I,v}$;
- si $\phi = (\psi_1 \vee \psi_2)$ alors $[\phi]_{I,v} = [\psi_1]_{I,v} + [\psi_2]_{I,v}$;
- si $\phi = (\exists x.\psi)$ alors $[\phi]_{I,v} = 1$ si et seulement s'il existe une valeur $d \in \mathbb{D}$ pour x telle que la formule ψ est vraie, c'est à dire $[\psi]_{I,w} = 1$ avec $w = v[d/x]$;
- si $\phi = (\forall x.\psi)$ alors $[\phi]_{I,v} = 1$ si et seulement si quelle que soit la valeur $d \in \mathbb{D}$ choisie pour x , la formule ψ est vraie, c'est à dire : $[\psi]_{I,w} = 1$ avec $w = v[d/x]$. ♦

Exemple : $v : y \mapsto 3$

$[\exists x.x + y = 5]_{I,v} = 1$ ssi il existe une valeur d de x tq $[d + y = 5]_{I,v} = 1$
ssi il existe une valeur d de x tq $[d + y]_{I,v} = 5$
ssi il existe une valeur d de x tq $d + [y]_{I,v} = 5$
ssi il existe une valeur d de x tq $d + v(y) = 5$
ssi il existe une valeur d de x tq $d + 3 = 5$
ssi vrai (pour $d = 2$)

Rôle des quantificateurs

Généralisation de $\wedge$ et $\vee$ à des domaines infinis

- ▶ $\forall x.P(x)$ “ $P(x)$ est vrai pour une **valeur quelconque** de x ”
- ▶ $\exists x.P(x)$ “ $P(x)$ est vrai pour une **certaine valeur** de x ”

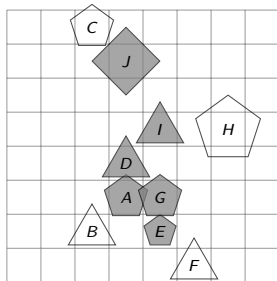
Exemple : $X = \{x_1, x_2, x_3 \dots\}$

- ▶ $\forall x.\Phi$ exprime : Φ est vraie pour x_1 **et** Φ est vraie pour x_2 **et** Φ est vraie pour x_3 **et** ...
- ▶ $\exists x.\Phi$ exprime : Φ est vraie pour x_1 **ou** Φ est vraie pour x_2 **ou** Φ est vraie pour x_3 **ou** ...

Exercise 3

- ▶ E est une **petite** figure, H et J sont de **grandes** figures, les autres figures sont de taille **moyenne**
- ▶ Comparaisons **strictes** : $\text{pluspetite}(x, y)$ se lit “ x est une figure **strictement plus petite** que y ”

Indiquer, pour chaque formule ci-dessous, si elle est vraie pour x et y interprétées sur l'ensemble des figures ci-dessous.



1. $\text{triangle}(C)$
2. $\text{pentagone}(F) \implies \text{cercle}(H)$
3. $\exists x. \neg \text{triangle}(x)$
4. $\forall x. \exists y. \text{pluspetite}(x, y)$
5. $\forall x. \exists y. \text{a_gauche}(x, y)$
6. $\exists x. \forall y. (y \neq x \implies \text{a_droite}(x, y))$
7. $\exists x. \forall y. \neg \text{pluspetite}(x, y)$
8. $\forall x. (\text{pentagone}(x) \implies \exists y. (\text{triangle}(y) \wedge \text{a_droite}(x, y)))$

Satisfiabilité et validité

Satisfiabilité, validité

Définition 3.12 Une formule ϕ est :

- *satisfaisable* s'il existe une interprétation I et une valuation v telles que $[\phi]_{I,v} = 1$
- *valide* si elle $[\phi]_{I,v} = 1$ pour toute interprétation I et toute valuation v . ♦

Exemple :

$$\exists k.(k \in \mathbb{N} \wedge n = 2k)$$

- ▶ **satisfaisable** “Existe-t-il un n tq la formule est vraie?”
→ oui, pour $n = 6$
- ▶ **non valide** “La formule est-elle vraie pour tout n ?”
→ non, pour $n = 3$

Le problème de satisfaisabilité s'énonce comme suit :

ENTRÉE : une formule ϕ de la logique des prédicats

QUESTION : existe-t-il une interprétation I et une valuation v telles que $I, v \models \phi$?

Théorème 3.1 La satisfaisabilité des formules de la logique des prédicats est indécidable. ♦

Quelques fragments décidables

On peut retrouver la **décidabilité** pour certains fragments :

Exemples :

- ▶ logique des prédicats monadiques (prédicats unaires, pas de fonctions)

$$\forall x. \exists y. (P(x) \implies Q(y))$$

- ▶ arithmétique de Presburger ($\{0, 1, +\}, \{\leq\}$)

$$\exists k. (k \in \mathbb{N} \wedge n = 2k)$$

- ▶ ...

Montrer la validité

Mq : $\forall x.P(x) \implies \exists y.P(y)$ est valide

Exercice 4

Pour chaque formule ci-dessous :

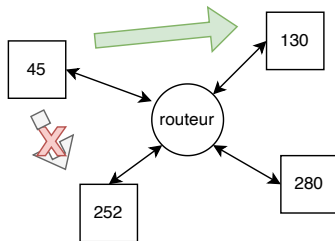
- ▶ parenthéser en respectant la priorité des opérateurs, puis indiquer les occurrences libres et liées,
- ▶ indiquer si elle satisfiable, valide ou insatisfiable.

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. $\exists x. (p(x) \vee q(x)) \iff \exists x. p(x) \vee \exists x. q(x)$

Problème 2 : cohérence des règles de filtrage d'un firewall

Contexte

Filtrer les paquets échangés
sur le réseau



► Règles **allow/block** :

r1 : allow [10,50] -> [127,255]

r2 : block [40,80] -> [250,512]

► Expressible en règles **allow** uniquement :

r1' : allow [10,39] -> [127,255]

r2' : allow [40,50] -> [127,249]

Formalisation : paquets, règles

- **Paquet** : $p = (@src, @dst)$

Exemple : $p = (45, 130)$

- **Règle** : $r = (from, to, allow)$ avec $allow \in \{0, 1\}$

Exemple : $r2' = ([40, 50], [127, 249], 1)$

$$matches_range(p, r) \triangleq (p.@src \in r.from) \wedge (p.@dst \in r.to)$$

$$allowed(p, r) \triangleq matches_range(p, r) \wedge r.allow$$

$$blocked(p, r) \triangleq matches_range(p, r) \wedge \neg r.allow$$

Formalisation : règles antagonistes

Soit R un ensemble de règles.

Exprimer : “Un paquet est autorisé par une règle de R et bloqué par une règle de R ”

$inconsistent(p) \triangleq$

Exemple :

$r1 : \text{allow } [10, 50] \rightarrow [127, 255]$

$r2 : \text{block } [40, 80] \rightarrow [250, 512]$

Formalisation : filtrage

- **Firewall** : un ensemble R de règles

Exprimer : “Un paquet passe le firewall s’il est autorisé par une règle de R et il est bloqué par aucune règle de R ”

$$\textit{filtered}(p, R) \triangleq$$

Exprimer : “Les firewalls R_1 et R_2 implémentent le même filtrage”

$$\textit{equivalent}(R_1, R_2) \triangleq$$

Implémentation

- ▶ Formalisation avec Z3

- ▶ Inspiré d'un **cas réel** de Microsoft :

[https://medium.com/@ahelwer/
checking-firewall-equivalence-with-z3-c2efe5051c8f](https://medium.com/@ahelwer/checking-firewall-equivalence-with-z3-c2efe5051c8f)

- ▶ Publication :

Nikolaj S. Bjørner, Karthick Jayaraman : Checking Cloud Contracts in Microsoft Azure. ICDCIT 2015 : 21-32

[https://www.microsoft.com/en-us/research/wp-content/uploads/
2016/02/nbjorner-icdcit2015.pdf](https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/nbjorner-icdcit2015.pdf)

- ▶ **Outil (qui a été) en production** sur Microsoft Azure :

<https://github.com/Z3Prover/FirewallChecker>

Exercices “spécification de fonctions”

Exercice 5

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 de retour :

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

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 6

Formaliser les spécifications des fonctions ci-dessous en logique des prédicats sous la forme d'un couple pré-condition (sur la valeur des paramètres) et post-conditions (sur la valeur des paramètres et des valeurs calculée par ces fonctions).

On nommera r la valeur retournée par ces fonctions.

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

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

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

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

3. La fonction `gcd` retourne le plus grand diviseur de a et b

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

4. La fonction `sort` qui trie le tableau t à n éléments

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