

Introduction au calcul relationnel

Jef Wijsen

Université de Mons (UMONS)

8 octobre 2024

Motivation

$$\text{sort}(ABUS) = \{Nom, Cru, Annee\}$$

$$\text{sort}(CRU) = \{Cru, Millesime, Qualite\}$$

$$\{z \mid \exists x \exists y (ABUS('An', x, y) \wedge CRU(x, y, z))\}$$

Logique des prédicats (syntaxe)

Alphabet :

- des noms de relation (R, S, T) avec leurs arités
- des variables (x, y, z)

Formules :

- Si R est un nom de relation d'arité n et x_1, x_2, \dots, x_n sont des variables, alors $R(x_1, x_2, \dots, x_n)$ est une formule dans laquelle les variables x_1, \dots, x_n sont **libres**.
- Si x est une variable libre d'une formule φ , alors $\exists x\varphi$ et $\forall x\varphi$ sont des formules dans lesquelles x n'est plus libre.
- Si φ est une formule, alors $\neg\varphi$ est une formule avec les mêmes variables libres que φ .
- Si φ_1 et φ_2 sont des formules, alors $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$ et $\varphi_1 \leftrightarrow \varphi_2$ sont des formules ayant comme variables libres chaque variable qui est libre dans φ_1 ou φ_2 .

Des parenthèses sont utilisées au besoin. Par simplicité, dans un premier temps, nous considérons seulement des formules sans égalité ($=$) et sans constantes.

Formule avec indication des variables libres

Soit R une relation d'arité 2 et S une relation d'arité 1.

$$\exists y \left(\underbrace{R(x, y)}_{x, y} \wedge \forall z \left(\underbrace{\left(\underbrace{R(x, z)}_{x, z} \rightarrow \underbrace{S(z)}_z \right)}_{x, z} \right) \right)$$

$\underbrace{\hspace{15em}}_{x, y}$

$\underbrace{\hspace{15em}}_x$

Logique des prédicats : sémantique "classique"

Une **structure** relationnelle est un couple $(\mathcal{D}, \mathcal{I})$ où \mathcal{D} est un **domaine fini** (mais voir slide 20 !) et non vide et l'**interprétation** \mathcal{I} associe une relation à chaque nom de relation. Ces relations n'utilisent que les valeurs de \mathcal{D} . Une formule sans variables libres est soit **vraie**, soit **fausse par rapport à cette structure**.

Exemple 1 (Vérité d'une formule par rapport à une structure)

$\mathcal{D} = \{a, e, i, o, u, b, c, d\}$	\mathcal{I}				
	R	A	B	S	A
		a	b		a
		e	b		e
		a	c		o
	i	c			

$$\exists y ((\exists x R(x, y)) \wedge \forall z (R(z, y) \rightarrow S(z)))$$

Si l'on choisit $y = b$, alors $\exists x R(x, "b")$ est vrai et $\forall z (R(z, "b") \rightarrow S(z))$ est vrai. En conclusion, la formule est vraie par rapport à cette structure.

Example 2

$$\mathcal{D} = \{a, e, o, b\} \quad \begin{array}{c|cc} R & A & B \\ \hline & a & b \\ & e & b \end{array} \quad \begin{array}{c|c} S & A \\ \hline & a \\ & o \end{array}$$

$$\exists y ((\exists x R(x, y)) \wedge \forall z (R(z, y) \rightarrow S(z)))$$

La formule est fausse par rapport à cette nouvelle structure.

Observation : La vérité peut dépendre du domaine

Exemple 3

$$\mathcal{D} = \{a, e, i\} \quad \begin{array}{c|c} S & A \\ \hline & a \\ & e \end{array}$$
$$\exists x (\neg S(x))$$

La formule est vraie par rapport à cette structure.

Exemple 4

$$\mathcal{D} = \{a, e\} \quad \begin{array}{c|c} S & A \\ \hline & a \\ & e \end{array}$$
$$\exists x (\neg S(x))$$

La formule est fausse par rapport à cette nouvelle structure.

Observation : Tautologies et contradictions

Tautologie (ou plutôt “formule valide”)

$\exists x (S(x) \vee \neg S(x))$ est vrai par rapport à chaque structure.

Contradiction

$\exists x (S(x) \wedge \neg S(x))$ est faux par rapport à chaque structure.

- Observation : la négation d'une tautologie est fausse par rapport à chaque structure (et est donc une contradiction).
- Dans la littérature, il est d'usage de réserver le terme **tautologie** à la logique propositionnelle (par exemple, $P \vee \neg P$), tandis que le terme **formule valide** est utilisé pour désigner une formule en logique des prédicats que est vraie par rapport à chaque structure.
- Existe-t-il un algorithme pour le problème suivant ?

INPUT : Une formule φ en logique propositionnelle.

QUESTION : Est-ce que φ est une tautologie ?

Terminologie

Quelques synonymes :

- logique propositionnelle = calcul propositionnel = logique des propositions = calcul des propositions = logique booléenne
- logique des prédicats = calcul des prédicats = logique du premier ordre = logique de premier ordre
- Le terme **calcul relationnel** est utilisé pour le calcul des prédicats dans le contexte des bases de données relationnelles, avec des spécificités expliquées au slide 13.

Souvent (mais pas dans ce cours), la syntaxe de la logique du premier ordre inclut des noms de fonctions, interprétés par des fonctions. Par exemple :

$$\forall x \forall y (S(x, y) \rightarrow R(x, y, f(x, y))) \wedge \forall z (R(x, y, z) \rightarrow z = f(x, y)).$$

Logique propositionnelle $\stackrel{?}{\subseteq}$ logique des prédicats

Une variable propositionnelle peut être simulée par un nom de relation P d'arité 0.

Notez que $P() \wedge \neg P()$ respecte bien la syntaxe du slide 3.

Comment P d'arité 0 peut être interprété ?

- Un nom de relation d'arité n peut être interprété, entre autres, par $\{\}$ (la relation vide d'arité n) et $\{(c_1, \dots, c_n)\}$.
- Donc, en prenant $n = 0$, on conclut que P peut être interprété par $\{\}$ (la relation vide d'arité 0) et $\{()\}$.
 - ▶ $P()$ est **faux** par rapport à une interprétation \mathcal{I} si $\mathcal{I}(P) = \{\}$; et
 - ▶ $P()$ est **vrai** par rapport à une interprétation \mathcal{I} si $\mathcal{I}(P) = \{()\}$.

Digression : Fermeture transitive

La **fermeture transitive**
d'une relation binaire est
la plus petite relation
transitive qui l'inclut.

Par exemple, la fermeture transitive

	A	B		A	B
	a	b	est	a	b
de	b	c		b	c
	c	b		c	b
				a	c
				b	b
			c	c	

Theorem 5

La fermeture transitive d'une relation binaire n'est pas exprimable en logique des prédicats (du premier ordre).

Pour chaque $\ell \in \{1, 2, 3, \dots\}$, la formule suivante ne calcule pas la fermeture transitive :

$$\left\{ \begin{array}{l} R(x, z) \vee \\ \exists y_1 (R(x, y_1) \wedge R(y_1, z)) \vee \\ \exists y_1 \exists y_2 (R(x, y_1) \wedge R(y_1, y_2) \wedge R(y_2, z)) \vee \\ \exists y_1 \exists y_2 \exists y_3 (R(x, y_1) \wedge R(y_1, y_2) \wedge R(y_2, y_3) \wedge R(y_3, z)) \vee \\ \vdots \\ \exists y_1 \exists y_2 \cdots \exists y_\ell (R(x, y_1) \wedge R(y_1, y_2) \wedge \cdots \wedge R(y_{\ell-1}, y_\ell) \wedge R(y_\ell, z)) \end{array} \right\}$$

Logique du deuxième ordre

Logique du deuxième ordre = logique du premier ordre + $Y(x_1, \dots, x_n)$ avec Y une variable (du deuxième ordre) pouvant être quantifiée par $\exists Y$ et $\forall Y$. Une variable de deuxième ordre d'arité n est interprétée par une relation d'arité n .

Par exemple, Y est une variable d'arité 2 dans la formule suivante qui calcule la fermeture transitive d'une relation R :

$$\left\{ x, z \mid \exists Y \left(Y(x, z) \wedge \forall u \forall w \left(Y(u, w) \leftrightarrow \left(\begin{array}{c} R(u, w) \\ \vee \\ \exists v (R(u, v) \wedge Y(v, w)) \end{array} \right) \right) \right) \right\}$$

- $Y(x, z)$ exprime : *“il existe un chemin de x à z ”*.
- La sous-formule exprime qu'il existe un chemin de u à w si et seulement s'il y a un arc de u à w , ou un arc de u à v suivi d'un chemin de v à w .

Logique des prédicats : sémantique en bases de données

Une [instance de] base de données relationnelle \mathcal{I} associe une relation à chaque nom de relation.

Le domaine reste inconnu !!! Mais il est su que les valeurs dans les relations appartiennent au domaine.

Une formule est *d^omain-dependent* (et considérée comme étant *erronée*) si sa vérité peut dépendre du domaine.

Exemple 6

$$\mathcal{D} \supseteq \{a, e, i, o, b, c\}$$

R	A	B
	a	b
	e	b
	a	c
	i	c

S	A
	a
	e
	o

$$\exists y ((\exists x R(x, y)) \wedge \forall z (R(z, y) \rightarrow S(z)))$$

Malgré que \mathcal{D} soit [largement] inconnu, on peut conclure que la formule est vraie par rapport à cette structure.

Exemple 7

$$\mathcal{D} \supseteq \{a, e\} \quad \begin{array}{c|c} S & A \\ \hline & a \\ & e \end{array}$$

$$\exists x (\neg S(x))$$

Cette formule est *d \odot main-dependent* (et donc *erronée* en théorie des bases de données), parce que sa vérité dépend de \mathcal{D} . En effet, si $\mathcal{D} = \{a, e\}$, la formule est fausse ; si $\mathcal{D} = \{a, e, i\}$, la formule est vraie.

Noter :

- Si le domaine reste inconnu, il n'est pas toujours possible de déterminer la valeur de vérité (vrai ou faux) d'une formule qui est *d \odot main-dependent*.
"Erronné" n'est pas une troisième valeur de vérité !
- Une formule qui n'est *pas d \odot main-dependent* est appelée *d \odot main-independent* (et *correct*).

Au moins une espèce de fleur est disponible en toute couleur...

<i>Fleur</i>	<i>F</i>	<i>C</i>	<i>Couleur</i>	<i>C</i>
	rose	rose		rose
	rose	jaune		jaune
	tulipe	jaune		brun

$$\exists x \left(\underbrace{(\exists y' Fleur(x, y'))}_{x \text{ est une fleur}} \wedge \forall y \left(\underbrace{(\exists x' Fleur(x', y))}_{y \text{ est une couleur de fleur}} \rightarrow Fleur(x, y) \right) \right) \text{ est vrai}$$

$$\exists x \left(\underbrace{(\exists y' Fleur(x, y'))}_{x \text{ est une fleur}} \wedge \forall y \left(\underbrace{Couleur(y)}_{y \text{ est une couleur}} \rightarrow Fleur(x, y) \right) \right) \text{ est faux}$$

$$\exists x \left(\underbrace{(\exists y' Fleur(x, y'))}_{x \text{ est une fleur}} \wedge \forall y (Fleur(x, y)) \right) \text{ est } d\text{omain-dependent}$$

Simplification (à mon goût personnel)

$$\exists x \left(\underbrace{(\exists y' Fleur(x, y'))}_{x \text{ est une fleur}} \wedge \forall y \left(\underbrace{(\exists x' Fleur(x', y))}_{y \text{ est une couleur de fleur}} \rightarrow Fleur(x, y) \right) \right)$$

est équivalent à

$$\exists x \left(\underbrace{(\exists y' Fleur(x, y'))}_{x \text{ est une fleur}} \wedge \forall x' \forall y (Fleur(x', y) \rightarrow Fleur(x, y)) \right)$$

$$\begin{aligned} \forall y ((\exists x' (F(x', y))) \rightarrow F(x, y)) &\equiv \forall y (\neg(\exists x' (F(x', y))) \vee F(x, y)) \\ &\equiv \forall y ((\forall x' (\neg F(x', y))) \vee F(x, y)) \\ &\equiv \forall y (\forall x' (\neg F(x', y) \vee F(x, y))) \\ &\equiv \forall x' (\forall y (\neg F(x', y) \vee F(x, y))) \\ &\equiv \forall x' (\forall y (F(x', y) \rightarrow F(x, y))) \end{aligned}$$

Formules équivalentes

Soit φ une formule sans variables libres.

La notation $(\mathcal{D}, \mathcal{I}) \models \varphi$ est une manière succincte de dire “ φ est vrai par rapport à la structure relationnelle $(\mathcal{D}, \mathcal{I})$ ”.

Noter : si φ est *domain-independent*, on peut se contenter d'écrire $\mathcal{I} \models \varphi$.

Soient φ_1 et φ_2 deux formules, sans variables libres. Ces deux formules sont équivalentes (notation : $\varphi_1 \equiv \varphi_2$) si pour toute structure relationnelle $(\mathcal{D}, \mathcal{I})$, nous avons $(\mathcal{D}, \mathcal{I}) \models \varphi_1 \leftrightarrow \varphi_2$.

Theorem (Trakhtenbrot 1950)

There exists no algorithm for the following problem :

INPUT : Two formulas φ_1, φ_2 without free variables.

QUESTION : Are φ_1 and φ_2 equivalent ?

Trakhtenbrot et Turing (pour votre culture générale)

- Le théorème de Trakhtenbrot suppose que tout domaine soit fini. Noter : avec un nombre fini de valeurs, il est impossible de construire une relation infinie.
- En 1936, Turing avait prouvé un tel résultat pour des relations qui peuvent être **finies ou infinies**.
- Aucun de ces deux théorèmes implique l'autre !!!
- En fait, il existent des formules φ et ψ sans variables libres telles que :
 - ▶ par rapport à chaque relation **finie**, φ et ψ ont la même valeur de vérité ; mais
 - ▶ il existe une relation **infinie** par rapport à laquelle φ est vrai et ψ est faux.

Motivation pour la théorie des modèles finis

Soit $sort(R) = \{A, B\}$.

PRIMARY KEY(A) (1)

UNIQUE(B) (2)

FOREIGN KEY(B) REFS R (3)

$\exists x ((\exists y R(x, y)) \wedge \neg \exists z R(z, x))$ (4)

R	A	B
	0	1
	1	2
	2	3
	\vdots	\vdots

un ensemble infini de tuples

PK(A) = $\forall x \forall y \forall z (R(x, y) \wedge R(x, z) \rightarrow y = z)$

UNIQUE(B) = $\forall x \forall y \forall z (R(y, x) \wedge R(z, x) \rightarrow y = z)$

FK(B) REFS R = $\forall x \forall y (R(x, y) \rightarrow \exists z R(y, z))$

Vérifiez : Aucune relation finie respecte toutes ces quatre contraintes !

Donc, contrairement au théorème de Turing, le théorème de Trakhtenbrot considère que les formules

(1) \wedge (2) \wedge (3) \wedge (4) et $\exists x \exists y (R(x, y) \wedge \neg R(x, y))$

sont équivalentes (il s'agit de deux contradictions).

La notion de **domaine** en théorie des bases de données

En théorie des bases de données,

- on part d'un domaine, dénoté par **dom**, qui est infini ;
- on requiert que toutes les relations soient finies ;
- on se limite aux formules qui sont *d😊main-independent* et, par conséquence, il est obsolète de connaître **dom**. Même l'observation "**dom** est infini" devient sans importance.

Ce choix correspond à la pratique : ce serait contre-intuitif que la réponse à une requête dépende des valeurs qui ne sont pas présentes dans les tables.

Non-existence d'un algorithme pour tester si une formule est *d☺main-independent*

Prouvez :

$\exists x (\neg S(x)) \vee (\varphi_1 \leftrightarrow \varphi_2)$ est *d☺main-independent* ssi $\varphi_1 \equiv \varphi_2$.

Preuve :

- Si $\varphi_1 \equiv \varphi_2$, cette formule est vraie par rapport à toute structure relationnelle \implies *d☺main-independent* ;
- si $\varphi_1 \not\equiv \varphi_2$, il existe une structure $(\mathcal{D}, \mathcal{I})$ telle que $(\mathcal{D}, \mathcal{I}) \not\models \varphi_1 \leftrightarrow \varphi_2$. Par rapport à cette structure, la formule est vraie si $\exists x (\neg S(x))$ est vraie \implies *d☺main-dependent*.

Variables libres

- La variable x a une occurrence libre dans la formule $\exists y (Fleur(x, y))$.
- La vérité d'une formule par rapport à une structure est seulement définie pour des formules sans variables libres.
- Une formule avec variable libre x définit une fonction (ou requête) : par rapport à une structure $(\mathcal{D}, \mathcal{I})$, renvoyer toute valeur $c \in \mathcal{D}$ telle que la formule est vraie si l'on remplace x par c .
- $\exists y (Fleur(x, y)) \rightsquigarrow$ renvoyer les espèces de fleurs ;
 $Fleur(x, y) \rightsquigarrow$ renvoyer les fleurs avec leurs couleurs.
- On utilisera la notation suivante :

$$\{x \mid \exists y (Fleur(x, y))\}$$
$$\{x, y \mid Fleur(x, y)\}$$

Requêtes peuvent être *d*☹️*main-dependent* (et donc incorrectes)

$$\{x \mid \neg \exists y (Fleur(x, y))\}$$

$$\{x \mid \exists y (Fleur(x, y)) \vee \exists u (Couleur(u))\}$$

Examples

<i>VINS</i>	<u><i>Cru</i></u>	<u><i>Millesime</i></u>	<u><i>Qualite</i></u>
	Chablis	1992	excellent
	Chablis	1993	bon
	Rothschild	1993	bon
	Rothschild	1994	imbuvable

<i>ABUS</i>	<u><i>Nom</i></u>	<u><i>Cru</i></u>	<u><i>Annee</i></u>
	Ed	Chablis	1992
	Ed	Chablis	1993
	An	Chablis	1993
	An	Rothschild	1993

Qui n'a jamais bu un vin excellent ?

$$\{x \mid \underbrace{\exists u \exists m (ABUS(x, u, m))}_{\pi_{\{Nom\}}(ABUS)} \wedge \underbrace{\neg \exists u \exists m (ABUS(x, u, m) \wedge VINS(u, m, 'excellent'))}_{\pi_{\{Nom\}}(E_1 \bowtie E_2)}\}$$

Comparaison avec l'algèbre SPJRUD. Soit

$$E_1 = \rho_{Annee \rightarrow Milleime}(ABUS)$$

$$E_2 = \sigma_{Qualite="excellent"}(VINS)$$

$$E_3 = \pi_{\{Nom\}}(E_1 \bowtie E_2)$$

E_3 renvoie les personnes ayant déjà bu un vin excellent. La requête demandée est donc :

$$\pi_{\{Nom\}}(ABUS) - E_3$$

Quels crus ont varié en qualité ?

$$\left\{ u \mid \exists q \exists v \left(\underbrace{\exists m (VINS(u, m, q)) \wedge \exists m (VINS(u, m, v))}_{E_2(u, q, v)} \right) \wedge \neg (q = v) \right\}$$

Soit

$$E_1 = \pi_{\{Cru, Qualite\}}(VINS)$$

$$E_2 = E_1 \bowtie \rho_{Qualite \rightarrow Valeur}(E_1)$$

E_2 renvoie $\{Cru : c, Qualite : q, Valeur : v\}$ ssi c est un cru qui a été de qualité q et de qualité v (il est possible que $q = v$). Soit

$$E_3 = E_2 - \sigma_{Qualite=Valeur}(E_2)$$

E_3 renvoie $\{Cru : c, Qualite : q, Valeur : v\}$ ssi c est un cru qui a été de qualité q et de qualité v avec $q \neq v$. La requête demandée est donc :

$$\pi_{\{Cru\}}(E_3)$$

Qui a bu tous les crus ?

$$\left\{ x \mid \exists u \exists m (ABUS(x, u, m)) \wedge \forall u \forall m \forall q (VINS(u, m, q) \rightarrow \exists m (ABUS(x, u, m))) \right\}$$

$$\left\{ x \mid \exists u \exists m (ABUS(x, u, m)) \wedge \neg \exists u ((\exists m \exists q VINS(u, m, q)) \wedge \neg \exists m (ABUS(x, u, m))) \right\}$$

Soit

$$E_1 = \pi_{\{Nom\}}(ABUS) \bowtie \pi_{\{Cru\}}(VINS)$$

$$E_2 = \pi_{\{Nom, Cru\}}(ABUS)$$

$$E_3 = E_1 - E_2$$

E_3 renvoie $\{Nom : n, Cru : c\}$ ssi c est un cru qui n'a pas été bu par la personne n . La requête demandée est donc :

$$\pi_{\{Nom\}}(ABUS) - \pi_{\{Nom\}}(E_3)$$

Arbre de syntaxe de “Qui a bu tous les crus ?”

