Le calcul des prédicats
Motivations

Le calcul des prédicats ou logique du premier ordre

- Le calcul propositionnel est très limité (opérations booléennes).
- On veut des constructions plus expressives, comme par exemple, pour tout x, si x est premier est supérieur à 2, alors x est impair. Formellement, $\forall x.\ ((premier(x) \wedge x > 2) \rightarrow impair(x))$.
- Variables, quantificateurs, fonctions, prédicats.
- Si on quantifie sur les variables $\Rightarrow$ premier ordre.
- Si on quantifie sur les fonctions/prédicats $\Rightarrow$ ordre supérieur.

Applications

- Bases de données
- Programmation logique
- Système de types pour les langages de programmation..
- Preuves de (propriétés de) programmes
- Circuits et architecture
- Intelligence artificielle (systèmes experts).

Bases de données - exemple

On veut modéliser une base de données:

- Ayant comme domaine un ensemble de personnes `Alice`, `Bob`, `Charles`, `Denis`, `Eric`, `Fanny`, `Georges`, `Hélène`.
- Autorisant deux relations binaires entre ces personnes, `parent` et `fratrie`.

Ces relations doivent satisfaire plusieurs contraintes, par exemple "*deux personnes sont dans la même fratrie ssi elles ont un parent commun*."
Les contraintes vont s'exprimer avec des **formules du premier ordre**:

$$\forall x.\ \forall y.\ (\texttt{fratrie}(x, y) \leftrightarrow \exists z\ \texttt{parent}(z, x) \wedge \texttt{parent}(z, y))$$

Programmation logique - exemple

On veut formaliser une relation ternaire de concatenation entre listes:

```
append([ ], L, L).
append([X|L1], L2, [X|L3]) :- append(L1, L2, L3).
```

Puis, on peut executer les requetes suivantes:

- Quel est le résultat de la concatenation de $[a, b]$ et $[d, e]$?
  $[a, b, d, e]$.
- Quelle est la liste à concatener à gauche de $[d, e]$ pour obtenir $[a, b, d, e]$? $[a, b]$.
- Quelles sont **toutes** les listes dont la concatenation donne $[a, b, d, e]$?
  [] et $[a, b, d, e]$.
  $[a]$ et $[b, d, e]$.
  $[a, b]$ et $[d, e]$.
  $[a, b, d]$ et $[e]$.
  $[a, b, d, e]$ et [].

Systèmes de Types pour les langages de programmation

Considérons le langage Ocaml. Soit A le type polymorphe $\forall\alpha.(\alpha \to \alpha)$. Les règles de typage sont données par des règles *logiques*. Puis on peut typer un programme comme suit:

$$\dfrac{\dfrac{\dfrac{}{y : \alpha \vdash y : \alpha}}{\vdash \lambda y.y : \alpha \to \alpha} \qquad \dfrac{\dfrac{\forall\alpha.(\alpha \to \alpha) \geq int \to int}{f : A \vdash f : int \to int} \qquad \dfrac{}{f : A \vdash 1 : int}}{f : \forall\alpha.(\alpha \to \alpha) \vdash f\ 1 : int}}{\vdash \mathtt{let}\ f = \lambda y.y\ \mathtt{in}\ f\ 1 : int}$$

Le calcul des prédicats

Considérons un programme séquentiel P qui calcule x divisé par y:

```
r:=x; /* le reste de la division*/
q:=0; /* le résultat de la division*/
while r >= y do
r := r - y;
q := q + 1;
done
```

- On se donne des règles de raisonnement pour toutes les constructions du langage.
- Puis on peut prouver des propriétés comme "pour tout $x \geq 0$ et tout $y > 0$ le programme P produit comme résultat q et r telles que $q * y + r = x$ et $0 \leq r < y$."

Le calcul des prédicats

Le calcul des prédicats

- Syntaxe
- Formalisation du langage naturel
- Sémantique
- Systèmes de preuves syntaxiques
  1. Deduction naturelle
  2. Calcul de Gentzen
  3. Résolution
     - ★ Théorie de l'unification
     - ★ Règles de résolution
     - ★ Propriétés de la résolution

Syntaxe: alphabet

- Les connecteurs $\rightarrow$, $\neg$, $\wedge$, $\vee$
- Les quantificateurs $\exists$, $\forall$
- Un ensemble dénombrable $\mathcal{X}$ de variables $x, y, z, \ldots$
- Une signature Σ contenant :
  - Un ensemble dénombrable de symboles de fonction $\Sigma_F = \{f, g, h, \ldots\}$, chacun ayant une arité.
  - Un ensemble dénombrable de symboles de prédicat $\Sigma_P = \{p, q, r, \ldots\}$, chacun ayant une arité.

On écrit f/n (ou p/n) pour dire que le symbole de fonction f (ou de prédicat p) est d'arité n.

Les termes

Soit une signature $\Sigma = \Sigma_F \cup \Sigma_P$. L'ensemble des termes par rapport à un ensemble de variables $\mathcal{X}$ et à une signature Σ est noté $\mathcal{T}_{\Sigma,\mathcal{X}}$. Il est le plus petit ensemble engendré par les règles suivantes:

$$\frac{x \in \mathcal{X}}{x \in \mathcal{T}_{\Sigma,\mathcal{X}}} \qquad \frac{t_1, \ldots, t_n \in \mathcal{T}_{\Sigma,\mathcal{X}} \quad f/n \in \Sigma_F}{f(t_1, \ldots, t_n) \in \mathcal{T}_{\Sigma,\mathcal{X}}}$$

- Les cas de base de la définition inductive: les variables et les symboles de fonction d'arité 0 (*i.e.* les constantes).
- On note $Var(t)$ l'ensemble des variables du terme t. Un terme est clos s'il ne contient aucune variable.

Exemple : Soit $\mathcal{X} = \{x, y, z\}$ et $\Sigma_F = \{a/0, b/0, f/1, g/2\}$. Voici quelques termes dans $\mathcal{T}_{\Sigma,\mathcal{X}}$: $a, x, f(b), g(f(y), f(b))$. Les termes a et $f(b)$ sont clos.

Les sous-termes

La définition inductive de $\mathcal{T}_{\Sigma,\mathcal{X}}$ induit une notion d'ensemble des sous-termes d'un terme t, notée $\mathcal{ST}(t)$, et donnée par la définition suivante:

- Si $t = x$, alors $\mathcal{ST}(x) = \{x\}$.
- Si $t = f(t_1, \ldots, t_n)$, alors $\mathcal{ST}(t) = \{t\} \bigcup_{i=1\ldots n} \mathcal{ST}(t_i)$.

On remarque que dans le cas particulier d'un symbole 0-aire $a \in \Sigma_F$, on obtient $\mathcal{ST}(a) = \{a\}$.

Exemple : Pour le terme $t = g(f(y), f(b))$ du transparent précédent, on a $\mathcal{ST}(t) = \{g(f(y), f(b)), f(y), f(b), y, b\}$.

Les atomes

Soit une signature $\Sigma = \Sigma_F \cup \Sigma_P$. L'ensemble des atomes sur un ensemble de variables $\mathcal{X}$ et une signature Σ est noté $\mathcal{A}_{\Sigma,\mathcal{X}}$. Un atome dans $\mathcal{A}_{\Sigma,\mathcal{X}}$ est un objet de la forme $p(t_1, \ldots, t_n)$, où p est un symbole de prédicat d'arité n dans Σ_P et $t_1, \ldots, t_n$ sont des termes dans $\mathcal{T}_{\Sigma,\mathcal{X}}$.

Exemple : Si $\Sigma_F = \{0/0, s/1\}$ et $\Sigma_P = \{inf/2\}$, alors 0 et $s(s(s(x)))$ sont des termes, 0 et $s(s(s(s(0))))$ sont des termes clos et $inf(0, s(s(s(x))))$ est un atome.

Les formules

L'ensemble des formules sur un ensemble de variables $\mathcal{X}$ et une signature Σ est noté $\mathcal{F}_{\Sigma,\mathcal{X}}$. Il est le plus petit ensemble engendré par les règles suivantes:

$$\frac{p(t_1, \ldots, t_n) \in \mathcal{A}_{\Sigma,\mathcal{X}}}{p(t_1, \ldots, t_n) \in \mathcal{F}_{\Sigma,\mathcal{X}}}\ \text{(cas de base)} \qquad \frac{A \in \mathcal{F}_{\Sigma,\mathcal{X}}}{\neg A \in \mathcal{F}_{\Sigma,\mathcal{X}}}$$

$$\frac{A, B \in \mathcal{F}_{\Sigma,\mathcal{X}}}{A \to B \in \mathcal{F}_{\Sigma,\mathcal{X}}} \qquad \frac{A, B \in \mathcal{F}_{\Sigma,\mathcal{X}}}{A \vee B \in \mathcal{F}_{\Sigma,\mathcal{X}}} \qquad \frac{A, B \in \mathcal{F}_{\Sigma,\mathcal{X}}}{A \wedge B \in \mathcal{F}_{\Sigma,\mathcal{X}}}$$

$$\frac{A \in \mathcal{F}_{\Sigma,\mathcal{X}} \quad x \in \mathcal{X}}{\forall x.\ A \in \mathcal{F}_{\Sigma,\mathcal{X}}} \qquad \frac{A \in \mathcal{F}_{\Sigma,\mathcal{X}} \quad x \in \mathcal{X}}{\exists x.\ A \in \mathcal{F}_{\Sigma,\mathcal{X}}}$$

Remarque :

- Nous omettons les parenthèses quand cela n'entraîne pas d'ambiguïtés. Nous écrivons aussi $\forall x_1 \ldots x_n.A$ pour $\forall x_1.\forall x_2.\cdots\forall x_n.A$ et $\exists x_1 \ldots x_n.A$ pour $\exists x_1.\exists x_2.\cdots\exists x_n.A$

Exemples

Soit $\Sigma_F = \{0/0, s/1\}$ et $\Sigma_P = \{enfant/1, mere/2, inf/2\}$. Est-ce que l'objet syntaxique suivant est une formule?

- $\forall x.\ (enfant(x) \to \exists y.\ mere(y, x))$
- $\forall x.\ inf(0, s(x))$
- $s(x) \wedge s(y)$
- $\forall x.\ mere(enfant(x), y)$
- $mere(x, y, z)$
- $\exists x.\ inf(0, s(s((x)))$
- $\exists x.\ inf(s(s(x), 0)$
- $s(s(0))$

Un cas particulier

Le calcul propositionnel peut se voir comme un calcul des prédicats sur une signature Σ t.q.

- l'ensemble Σ_F est vide,
- l'ensemble Σ_P contient uniquement des prédicats 0-aires,
- les quantificateurs ne sont jamais utilisés.

Les sous-formules

La définition inductive de $\mathcal{F}_{\Sigma,\mathcal{X}}$ induit une notion d'ensemble des sous-formules d'une formule A, notée $\mathcal{SF}(A)$, et donnée par la définition suivante:

- Si A est un atome, $\mathcal{SF}(A) = \{A\}$.
- Si $A = \neg B$, $\mathcal{SF}(A) = \{A\} \cup \mathcal{SF}(B)$.
- Si $A = B \# C$, $\mathcal{SF}(A) = \{A\} \cup \mathcal{SF}(B) \cup \mathcal{SF}(C)$.
- Si $A = \forall x.B$, $\mathcal{SF}(A) = \{A\} \cup \mathcal{SF}(B)$.
- Si $A = \exists x.B$, $\mathcal{SF}(A) = \{A\} \cup \mathcal{SF}(B)$.

Exemple : Pour la formule $A = \forall x.\ (enfant(x) \to \exists y.\ mere(y,x))$ du du transparent précédent, on a
$\mathcal{SF}(A) = \{\forall x.\ (enfant(x) \to \exists y.\ mere(y,x)), enfant(x) \to \exists y.\ mere(y,x), enfant(x), \exists y.\ mere(y,x), mere(y,x)\}$.

Variables libres et liées

Les variables libres (VI) et liées (VE) d'une formule sont définies comme suit :

- Si A est un atome $p(t_1,\ldots,t_n)$, $VE(A) = \emptyset$ et $VI(A)$ contient toutes les variables de ses termes, *i.e.* $VI(A) = \bigcup_{1\leq i\leq n} Var(t_i)$.
- Si $A = \neg B$, $VI(A) = VI(B)$ et $VE(A) = VE(B)$.
- Si $A = B \# C$, $VI(A) = VI(B) \cup VI(C)$ et $VE(A) = VE(B) \cup VE(C)$.
- Si $A = \forall x.\ B$ ou $A = \exists x.\ B$, $VI(A) = VI(B) \setminus \{x\}$ et $VE(A) = VE(B) \cup \{x\}$.

Exemple : Si $A = \forall x.\ q(x, f(x,y))$ on a $VI(A) = \{y\}$ et $VE(A) = \{x\}$.
Si $B = r(x) \vee \forall x.\ q(x, f(x,y))$ on a $VI(B) = \{x, y\}$ et $VE(B) = \{x\}$.

Renommage

Remarque : On suppose que l'on peut toujours renommer les variables liées d'une formule afin de l'écrire sous une forme rectifiée :

- toutes les variables liées d'une formule sont distinctes. On ne peut pas avoir par exemple $\forall x.\ \exists x.\ A$.
- les variables libres et liées d'une formule A portent des noms distincts, i.e. $VI(A) \cap VE(A) = \emptyset$. On ne peut plus écrire la formule $r(x) \vee \forall x.\ q(x, f(x,y))$.

Exemples de renommage

- Renommages valides de la formule $\forall x.\exists y.p(x,y)$: $\forall z.\exists y.p(z,y)$, $\forall x.\exists z.p(x,z)$, $\forall z.\exists w.p(z,w)$, $\forall y.\exists x.p(y,x)$ ...
- Renommages valides de la formule $(\forall x.p(x)) \vee p(x)$: $(\forall z.p(z)) \vee p(x)$, ..., mais pas $(\forall x.p(x)) \vee p(z)$, ni $(\forall z.p(z)) \vee p(z)$,
- Renommages valides de la formule $\forall x.\exists x.p(x)$: $\forall y.\exists x.p(x)$, $\forall x.\exists y.p(y)$, ..., mais pas $\forall y.\exists x.p(y)$

Mais comment formaliser cette notion de renommage?

Il faut définir une notion de substitution sur les formules.

Les substitutions

Définition :

- Une substitution est une fonction $\sigma : \mathcal{X} \to \mathcal{T}_{\Sigma,\mathcal{X}}$. On note $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ si $\sigma(x_i) = t_i$ et $x_i \neq t_i$ $(1 \leq i \leq n)$. Le domaine de σ est $\mathtt{dom}(\sigma) := \{x \mid \sigma(x) \neq x\}$ et l'ensemble de variables de σ est $Var(\sigma) := \bigcup Var(t_i)$.
- Un renommage est une substitution de la forme $\{x_1 \leftarrow y_1, \ldots, x_n \leftarrow y_n\}$, où $y_i \neq y_j$ $(i \neq j)$, *i.e.* chaque variable est remplacée par une variable distincte.
- L'application d'une substitution σ à un terme est l'extension de σ aux termes donnée par $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.
- Soient σ et τ deux substitutions. La composition de σ avec τ est donnée par $\sigma \circ \tau(x) = \sigma(\tau(x))$.
- Si σ est une substitution, alors la restriction de σ à $x := t$ est une nouvelle substitution $\sigma[x := t]$ donnée par $\sigma[x := t](y) = \sigma(y)$ si $y \neq x$ et $\sigma[x := t](x) = t$ sinon.

Exemples: voir tableau.

Substitution d'une formule

Soit Σ une signature et $\mathcal{X}$ un ensemble de variables.

Définition : Soit $\sigma = \{x_1 \leftarrow u_1, \ldots, x_k \leftarrow u_k\}$ $(k \geq 0)$ une substitution. L'application d'une substitution σ à une formule A est l'opération qui consiste à remplacer toute occurrence libre de x_i dans A par u_i. Par induction sur A :

- $\sigma(r(t_1, \ldots, t_n)) = r(\sigma(t_1), \ldots, \sigma(t_n))$
- $\sigma(\neg B) = \neg\sigma(B)$ et $\sigma(B\#C) = \sigma(B)\#\sigma(C)$
- $A = \forall x.\ B$, où l'on suppose (grâce à un renommage) $x \notin (Var(\sigma) \cup \mathtt{dom}(\sigma))$. Alors $\sigma(\forall x.\ B) = \forall x.\ \sigma(B)$.
- Pareil pour $\sigma(\exists x.\ B)$

Exemples de substitutions

Voir Tableau.

Mais comment formaliser la notion de renommage?

Etant donnée une formule, comment calculer un renommage?
Appliquer plusieurs fois (clôture transitive) la définition inductive non-déterministe suivante:

- $\mathtt{renom}(p(t_1, \ldots, t_n)) = p(t_1, \ldots, t_n)$.
- $\mathtt{renom}(\neg B) = \neg\mathtt{renom}(B)$.
- $\mathtt{renom}(B_1\#B_2) = \mathtt{renom}(B_1)\#\mathtt{renom}(B_2)$.
- $\mathtt{renom}(\forall x.B) = \forall x.\mathtt{renom}(B)$
- $\mathtt{renom}(\forall x.B) = \forall y.\mathtt{renom}(\{x \leftarrow y\}B)$, où y est une variable fraîche.
- $\mathtt{renom}(\exists x.B) = \exists x.\mathtt{renom}(B)$
- $\mathtt{renom}(\exists x.B) = \exists y.\mathtt{renom}(\{x \leftarrow y\}B)$, où y est une variable fraîche.

Deux exemples simples

- $$\begin{array}{l}\texttt{renom}(\forall x.\exists y.p(x,y)) = \\ \forall x.\texttt{renom}(\exists y.p(x,y)) = \\ \forall x.\exists z.\texttt{renom}(\{y \leftarrow z\}p(x,y)) = \\ \forall x.\exists z.\texttt{renom}(p(x,z)) = \\ \forall x.\exists z.p(x,z)\end{array}$$
- $$\begin{array}{l}\texttt{renom}(\forall x.\exists y.p(x,y)) = \\ \forall z.\texttt{renom}(\{x \leftarrow z\}\exists y.p(x,y)) = \\ \forall z.\texttt{renom}(\exists y.p(z,y)) = \\ \forall z.\exists w.\texttt{renom}(\{y \leftarrow w\}p(z,y)) = \\ \forall z.\exists w.\texttt{renom}(p(z,w)) = \\ \forall z.\exists w.p(z,w)\end{array}$$

Un exemple plus compliqué

Comment calculer $\forall y.\exists x.p(y,x)$ à partir de $\forall x.\exists y.p(x,y)$?
On calcule deux fois un renommage:

$$\begin{array}{l}\texttt{renom}(\forall x.\exists y.p(x,y)) = \forall x'.\exists y'.p(x',y') \\ \texttt{renom}(\forall x'.\exists y'.p(x',y')) = \forall y.\exists x.p(y,x)\end{array}$$

Puis, on compose:

$$\texttt{renom}(\texttt{renom}(\forall x.\exists y.p(x,y))) = \forall y.\exists x.p(y,x)$$

À partir de maintenant on peut assumer qu'une formule est rectifiée si cela est nécessaire. On verra dans la suite (ou en TD) que cette supposition est correcte.

Formalisation du langage naturel

- Tous les humains sont méchants.
$$\forall x.\ (H(x) \rightarrow M(x))$$
- Que veut dire
$$\forall x.\ (H(x) \wedge M(x)) \qquad ?$$
- Seulement les humains sont méchants.
$$\forall x.\ (M(x) \rightarrow H(x))$$
- Il existe un humain méchant.
$$\exists x.\ (H(x) \wedge M(x))$$

Formalisation du langage naturel

- Il n'existe pas d'humain méchant.

$$\neg\exists x.\ (H(x) \wedge M(x))$$

- Il existe un humain qui aime tous les chats.
  $\exists x.\ (H(x) \wedge Aimetousleschats(x))$
  $Aimetousleschats(x) \equiv \forall y.\ (Chat(y) \rightarrow Aime(x,y))$
- Chaque humain sait qui le déteste.

$$\forall x.\ (H(x) \rightarrow Connaitdeteste(x))$$

$$Connaitdeteste(x) \equiv \forall y.\ (D(y,x) \rightarrow C(x,y))$$

Formalisation du langage naturel

- Chaque personne aime quelqu'un et personne n'aime tout le monde ou bien quelqu'un aime tout le monde et quelqu'un n'aime personne.

$$(A_1 \wedge B_1) \vee (A_2 \wedge B_2)$$
$$aimetoutlemonde(x) \equiv \forall y.\ Aime(x,y)$$
$$aimepersonne(x) \equiv \forall y.\ \neg Aime(x,y)$$

$$A_1 \equiv \forall x.\ (H(x) \rightarrow \exists y.\ Aime(x,y))$$
$$B_1 \equiv \neg\exists x.\ (H(x) \wedge aimetoutlemonde(x))$$
$$A_2 \equiv \exists x.\ (H(x) \wedge aimetoutlemonde(x))$$
$$B_2 \equiv \exists x.\ (H(x) \wedge aimepersonne(x))$$