Yannis Haralambous

Département Informatique & UMR CNRS 6285 Lab-STICC
D03-118A, tél. 1427
yannis.haralambous@telecom-bretagne.eu

INF 424
Logique

8 avril 2016

Table des matières

Une vidéo de cours d'une durée de 4h30 a été préparée pour vous aider à reviser.
Vous la trouverez sur Moodle.

Mode d'emploi

La logique, ou plutôt *les* logiques puisqu'on en traitera plus d'une, sont un vaste sujet et le temps que vous allez passer sur ce module est limité. Voici quelques conseils pour en profiter au mieux :

1. l'objectif pédagogique est que vous soyez capable de modéliser une situation en utilisant le formalisme logique, et ensuite, le cas échéant, d'y répondre en utilisant des algorithmes vus en cours ou d'autres algorithmes que vous jugerez utiles ;
2. il y a une multitude de définitions, le but n'est pas de les apprendre par cœur mais de comprendre ce qu'elles signifient, et comment elles se placent dans le système de concepts et de relations de la logique ;
3. on n'apprend vraiment une notion que quand on l'applique à la résolution d'un problème auquel on est confronté. Donc, un conseil : essayez de bien comprendre chaque notion introduite et d'imaginer son utilité en l'appliquant à un problème ;
4. vous trouverez sur Moodle une *vidéo* du cours. Elle date de 2013, donc certaines parties peuvent ne pas avoir été actualisées (et le chapitre sur les logiques de description n'y est pas traité), mais néanmoins elle vous aidera à la révision du cours ;
5. si vous avez des questions, n'hésitez pas à les poser, et surtout sur le forum Moodle : cela permettra aux promotions futures d'en profiter ;
6. si vous voulez consulter d'autres ouvrages sur la logique, je vous conseille le : Mordechai Ben-Ari, *Mathematical Logic for Computer Science*, 3e édition (2012), sa cote à la bibliothèque brestoise de Télécom Bretagne est `1.21 BEN` (la bibliothèque ne propose que la 2e édition). Également très intéressant, l'ouvrage *Outils logiques pour l'intelligence artificielle* de Jean-Paul Delahaye (connu pour ses excellentes vulgarisations), vous le trouverez à la bibliothèque sous la cote `2.7 DELA` ;
7. à la fin de ce polycopié vous trouverez un index (resp. un index des notations), qui vous permettra de retrouver rapidement les notions (resp. les notations) dans le cours. Il n'y a pas de glossaire parce qu'il n'est pas toujours pertinent de donner une définition en dehors de son contexte ;
8. le code source de ce cours se trouve sur Overleaf, à l'adresse `https://www.overleaf.com/read/ttcncvjghtzg`. Ceux qui seraient curieux de voir le code LaTeX utilisé peuvent s'y rendre. N'hésitez pas d'utiliser ce système — libre et performant — pour vos rapports et autres documents.

Trêve de conseils, il ne me reste plus qu'à vous souhaiter bon courage
et que la force de la logique soit avec vous !

Introduction

Humans, it seems, know things ; and what they know helps them do things.[1]

Those unused to formal theories should try not to overreact to the symbolic nature of the formalism.[2]

Quand on proclama que la Bibliothèque comprenait tous les livres, la première réaction fur un bonheur extravagant... À l'espoir éperdu succéda, comme il est naturel, une dépression excessive.[3]

L'intelligence humaine opère à travers des **processus de raisonnement** basés sur des *représentations (internes) de la connaissance*.

En intelligence artificielle on crée des **bases de connaissances**. Celles-ci contiennent des **énoncés** exprimés dans des **langages de représentation des connaissances**.

Il y a des énoncés de départ (appelés **axiomes**) et des processus de dérivation de nouveaux énoncés à partir des énoncés existants : l'**inférence**.

On peut interroger la base de connaissances : pour obtenir la réponse à une requête, on se sert également du processus d'inférence.

À la base de la représentation des connaissances, on trouve la logique.

Il y a plusieurs types de logique : propositionnelle, du 1er ordre, du 2e ordre, modale, floue, temporelle, etc.

Toutes les logiques se partagent les notions suivantes :

1. Une **syntaxe** du langage : ainsi, par exemple, $x + y = 4$ est un énoncé bien formé, $x4y+ =$ ne l'est pas.
2. Un **système déductif**, c'est-à-dire un ensemble d'énoncés qualifiés d'**axiomes** et une **méthode de preuve** qui permet d'en déduire d'autres énoncés (appelés **théorèmes**) à partir de ceux-ci, en utilisant des **règles d'inférence**.
3. Une **théorie des modèles**, c'est-à-dire une notion d'**interprétation** (ou de « monde possible ») des énoncés (intuitivement : une manière de leur donner du sens dans un domaine choisi) qui les dote d'une valeur de vérité. Une interprétation qui rend un énoncé vrai est un **modèle** de celui-ci (ainsi, $x + y = 4$ est vraie dans un monde où $x = y = 2$ et fausse dans un monde où $x = 1, y = -1$).
4. On dit qu'un énoncé β est **conséquence** d'un énoncé α ($\alpha \models \beta$) quand tout modèle de α est aussi modèle de β. Exemple : $(x + y = 2) \models (x + y > 0)$.

On montre, pour les logiques propositionnelle et du 1er ordre, que la preuve par déduction (dans le cadre d'un système déductif donné) et la conséquence (au sens de la théorie des modèles) sont *équivalentes* : autrement dit, β est conséquence de α si et seulement si on peut déduire β à partir de α par un nombre fini d'inférences.

1. S. Russell & P. Norvig, *Artificial Intelligence : A Modern Approach*, 3e édition, Prentice Hall, 2009.
2. R. Cann, *Formal Semantics : An Introduction*, Cambridge Textbooks in Linguistics, 1993.
3. Jorge Luís Borges, cité par J.-P. Delahaye, *Outils logiques pour l'intelligence artificielle*, Eyrolles, 1988.

Chapitre 1

Systèmes formels

La notion de système formel

Cette notion constitue le lien avec la première partie du module INF 424.

Un **système formel** est la donnée :

1. d'un **alphabet** Σ fini ou dénombrable,
2. d'un sous-ensemble récursif F de l'ensemble Σ^* des suites finies d'éléments de Σ, on appelle F l'ensemble des **énoncés bien formés**,
3. d'un **système déductif**, c'est-à-dire
   (a) d'un sous-ensemble récursif A de F, appelé ensemble des **axiomes**,
   (b) d'un ensemble fini R de prédicats décidables définis sur F appelés **règles d'inférence**.

Explications

Les *énoncés bien formés* F sont les combinaisons de symboles de Σ qui respectent une certaine syntaxe. À noter que l'ensemble des énoncés bien formés peut être défini par une grammaire formelle. ATTENTION : ne pas confondre les règles de production de la grammaire formelle et les règles d'inférence du système formel !

Les **règles d'inférence** permettent de déduire de nouveaux énoncés bien formés à partir d'énoncés existant.

Les **axiomes** sont des énoncés que l'on a choisi de ne pas déduire d'autres énoncés, ils servent d'énoncés de départ à partir desquels on déduit tous les autres.

Les **propriétés de récursivité** de F et de A, et de décidabilité de R signifient *grosso modo* :
— il existe un programme P_1 qui, pour la donnée de $f \in \Sigma^*$ indique, au bout d'un temps fini, si $f \in F$ ou si $f \notin F$,
— il existe un programme P_2 qui, pour la donnée de $f \in F$ indique, au bout d'un temps fini, si $f \in A$ ou si $f \notin A$,
— pour chaque $r \in R$ il existe un programme Q_r qui, pour la donnée de $f_1, \ldots, f_{n-1}$ et de g, indique, au bout d'un temps fini, si on a bien $f_1, f_2, \ldots, f_n \vdash g$ ou non.

Un élément r de R est une application de F^n (pour $n \geq 1$) dans l'ensemble {vrai, faux}. Au lieu d'écrire $r(f_1, f_2, \ldots, f_{n-1}, g)$ on écrira

$$\frac{f_1, f_2, \ldots, f_{n-1}}{g} \quad r$$

ou alors $f_1, f_2, \ldots, f_{n-1} \underset{r}{\vdash} g$, et on lira « g peut être déduit de $f_1, f_2, \ldots, f_{n-1}$ par la règle d'inférence r ».

Une **preuve** dans un système déductif est une suite d'ensembles de formules telle que chaque élément est soit un axiome, soit peut être déduit par les éléments des ensembles précédents en appliquant les règles d'inférence. Si $\{A\}$ est le dernier ensemble de la suite, on dira que A est un **théorème**, et on écrira $\vdash A$.

Chapitre 2

Logique du 1er ordre

2.1 Système formel

La logique du 1er ordre est un **système formel** dont l'**alphabet** est constitué :

- de **prédicats** n-aires $P, Q, R, \ldots$,
- de **fonctions** n-aires $f, g, h, \ldots$,
- de **variables** $X, Y, Z, \ldots$,
- de **connecteurs** $\neg$ (négation) et $\rightarrow$ (implication),
- du **quantificateur universel** $\forall$ (« pour tout »),
- du signe de l'**égalité** $=$,
- et des **parenthèses** ( ).

Le cas particulier du prédicat 0-aire est appelé une **proposition**, et le cas de la fonction 0-aire est appelé une **constante**.

Parmi ces objets/symboles, certains ont une sémantique (= un sens) pré-établie, on les appelle des **symboles logiques** : $\neg, \rightarrow, \forall$, d'autres n'ont pas de sens en soi, et ne l'acquièrent que lors d'une **interprétation** (que l'on verra plus loin, en p. 6), on les appelle des **symboles non-logiques**.

Un **terme** est une constante, une variable ou une fonction d'autres termes. Un **terme clos** est un terme sans variables.

Un **énoncé** (ou **formule**) est une suite valide de symboles (valide au sens de la grammaire BNF donnée ci-dessous en § 2.1.1).

Il faut bien comprendre que tant qu'on reste au niveau du système déductif (axiomes et règles d'inférence) et qu'on ne donne pas d'interprétation, on n'aura *que des symboles obéissant à certaines règles syntaxiques* (cf. ci-dessous).

C'est la **théorie des modèles** qui nous donnera la possibilité d'*interpréter* ces symboles dans un domaine donné, aptes à être utilisés en mathématiques ou à modéliser la réalité.

2.1.1 Grammaire BNF de la logique du 1er ordre

Voici le BNF de la syntaxe de la logique du 1er ordre :

Énoncé	→	ÉnoncéAtomique | ÉnoncéComplexe
ÉnoncéAtomique	→	Proposition | Prédicat(Terme+)
		| Terme = Terme
ÉnoncéComplexe	→	(Énoncé)
		| ¬Énoncé
		| Énoncé → Énoncé
		| Quantificateur Variable, Énoncé
Terme	→	Fonction(Terme+) | Constante | Variable
Quantificateur	→	∀
Constante	→	A | B | Michel...
Variable	→	X | Y | Z...
Prédicat	→	aime() | existe()...
Fonction	→	père() | carré() | sinus()...

Précédence des opérateurs : $\forall$, $\neg$, $=$, $\rightarrow$.

2.1.2 Système déductif

Rappel : un **système déductif** est un ensemble d'**axiomes** et un ensemble de **règles d'inférence**.

Axiomes

Voici les axiomes de la logique du 1er ordre :

1. $(P \rightarrow (Q \rightarrow P))$ (répétition),
2. $(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$ (distribution conditionnelle),
3. $(\neg P \rightarrow \neg Q) \rightarrow (Q \rightarrow P)$ (**contraposition**),
4. $\forall x\, P(x) \rightarrow P(t)$, où t est un terme clos (élimination universelle),
5. $\forall x\, (P \rightarrow Q) \rightarrow (\forall x\, P \rightarrow \forall x\, Q)$ (distribution universelle).

Règles d'inférence

Rappelons la notation : si, à travers la règle d'inférence r, à partir de plusieurs énoncés $\alpha_1, \ldots, \alpha_n$ on peut inférer (= déduire, prouver) un énoncé β, alors on écrit

$$\frac{\alpha_1, \ldots, \alpha_n}{\beta}\, r.$$

Deux règles d'inférence sont indispensables, la première est celle du **modus ponens** :

$$\frac{\alpha \rightarrow \beta \qquad \alpha}{\beta}\ \textit{modus ponens}$$

et la deuxième est celle de la **généralisation** :

$$\frac{A}{\forall x\, A}\ \text{généralisation.}$$

Exemple d'application des règles d'inférence

Montrons que $\forall x(P(x) \to P(x))$:

(1) $P(a) \to ((P(a) \to P(a)) \to P(a))$ Ax. 1
(2) $P(a) \to ((P(a) \to P(a)) \to P(a)) \to ((P(a) \to (P(a) \to P(a)) \to (P(a) \to P(a))))$ Ax. 2
(3) $(P(a) \to (P(a) \to P(a))) \to (P(a) \to P(a))$ (1)+(2) *mod. pon.*
(4) $(P(a) \to (P(a) \to P(a)))$ Ax. 1
(5) $P(a) \to P(a)$ (3)+(4) *mod. pon.*
(6) $\forall x(P(x) \to P(x))$ (5) général.

Ce n'est certes pas un résultat très spectaculaire, mais cela montre qu'il aurait été inutile de l'inclure parmi les axiomes.

Et maintenant, que fait-on ?

Le **système déductif** nous fournit un formalisme. Nous « jouons » avec les symboles : nous formons des énoncés bien formés, nous déclarons que certains parmi eux sont des axiomes et nous obtenons d'autres énoncés par inférences successives. Mais à quoi cela sert-il ?

Pour faire le lien avec les mathématiques ou le monde réel, il faut donner un sens aux symboles, et en déduire des valeurs de vérité pour les énoncés dans des domaines d'application donnés. Cela se fait à travers la **théorie des modèles**.

Mais avant d'attaquer cette dernière, définissons quelques connecteurs supplémentaires dans le but de rapprocher le formalisme des énoncés de la langue naturelle.

2.1.3 Connecteurs et un quantificateur supplémentaires

Pour faciliter la lecture des énoncés, ainsi que le passage des situations du monde réel aux énoncés logiques, on introduit trois connecteurs supplémentaires : le $\wedge$ (« et », également appelé **conjonction**), le $\vee$ (« ou », également appelé **disjonction**) et le $\leftrightarrow$ (« si et seulement si », **double implication**), ainsi que le quantificateur existentiel $\exists$ (« il existe »).

Ils sont définis à partir des symboles existants de la manière suivante :

- $A \vee B = \neg B \to A$,
- $A \wedge B = \neg(\neg A \vee \neg B) = \neg(B \to \neg A)$,
- $A \leftrightarrow B = (A \to B) \wedge (B \to A) = \neg((B \to A) \to \neg(A \to B))$,
- $\exists x\, P = \neg\forall x\, \neg P$.

Voici la grammaire BNF revue pour tenir compte des connecteurs que nous venons de rajouter :

Énoncé	→	ÉnoncéAtomique | ÉnoncéComplexe
ÉnoncéAtomique	→	Proposition | Prédicat(Terme+)
		| Terme = Terme
ÉnoncéComplexe	→	(Énoncé)
		| ¬Énoncé
		| Énoncé ∧ Énoncé | Énoncé ∨ Énoncé
		| Énoncé → Énoncé
		| Énoncé ↔ Énoncé
		| Quantificateur Variable, Énoncé
Terme	→	Fonction(Terme+) | Constante | Variable
Quantificateur	→	∀ | ∃
Constante	→	A | X | Michel...
Variable	→	a | x | s...
Prédicat	→	aime | existe...
Fonction	→	Père | Carré | Sinus...

Précédence des connecteurs : $\forall/\exists$, $\neg$, $=$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$.

2.1.4 Variables libres et liées

Une variable est *libre* quand elle n'est pas quantifiée. Une variable quantifiée est appelée **liée**.

Attention : dans certains énoncés mal écrits, on peut trouver le même symbole pour une variable liée et une variable libre, par exemple dans $P(x,y) \wedge \forall x\, Q(x,y)$. Dans cet énoncé, x est-elle libre ou liée ?

Pour éviter ce genre de problème il convient de renommer les variables liées en utilisant des symboles qui ne figurent pas parmi les variables libres. Cet énoncé devient donc $P(x,y) \wedge \forall z\, Q(z,y)$, dans lequel x et y sont libres et z est liée.

Un énoncé sans variables libres est appelé **énoncé clos** ou **sentence**.

2.2 Théorie des modèles

2.2.1 Structure, signature, interprétation

En théorie des modèles, les prédicats ont une **valeur de vérité** (ils peuvent être **vrais** ou **faux**) et les variables, fonctions et constantes prennent leurs valeurs dans un ensemble, appelé **domaine** ou **univers**. Autrement dit, les formules qui jusqu'à maintenant n'étaient que des assemblages de symboles prennent un sens, elles sont interprétées dans un contexte donné.

Une **structure** est un triplet $\mathcal{A} = (\Delta, \sigma, \mathbf{I})$ où Δ est un **domaine** (ou **univers**), σ est une **signature** et $\mathbf{I}$ est une **fonction d'interprétation**.

Un **domaine** Δ est un ensemble quelconque (non vide) dans lequel les variables vont prendre leurs valeurs ; il servira également à définir les images par la fonction d'interprétation des constantes, fonctions et prédicats.

Une **signature** σ est un ensemble de symboles de fonction et de prédicat, ainsi qu'une fonction qui envoie ces symboles dans $\mathbb{N}_0$ appelée **arité**. Les fonctions d'arité 0 sont appelées des **constantes**.

Une **fonction d'interprétation** est une fonction $\mathbf{I}$ qui envoie :

1. les constantes de σ dans Δ, on notera $\mathbf{I}(c) = c^{\mathcal{A}} \in \Delta$;
2. toute fonction n-aire ($n \geq 1$) f de σ vers une fonction $\mathbf{I}(f) = f^{\mathcal{A}} \colon \Delta^n \to \Delta$;
3. toute prédicat n-aire ($n \geq 1$) P de σ vers une fonction $\mathbf{I}(P) = P^{\mathcal{A}} \colon \Delta^n \to \{\bot, \top\}$, où $\bot$ signifie «faux» et $\top$ signifie «vrai».

Ayant défini $\mathbf{I}$ sur les éléments de σ on peut l'étendre aux termes : on obtient des applications $t^{\mathbf{I}} \colon \Delta^j \to \Delta$ pour chaque **terme** ayant j variables.

De même, on peut l'étendre aux énoncés. On obtient donc une application $\alpha^{\mathbf{I}} \colon \Delta^k \to \{\top, \bot\}$ pour α en supposant que α a k variables libres.

Autrement dit, si l'énoncé est **clos**, c'est-à-dire si il n'a aucune variable libre, alors toute interprétation lui associe une valeur de vérité (puisque $k = 0$).

Définition 2.1. *Un **modèle** (= monde possible) est une **structure** qui rend un énoncé vrai.*

Parenthèse : vision ensembliste de l'interprétation

Si $\mathcal{A} = (\Delta, \sigma, \mathbf{I})$ est une structure, alors on peut associer à tout prédicat unaire P, le sous-ensemble $P^{\mathcal{A}}$ de Δ des $e \in \Delta$ tels que $P(e)$ soit vrai.

Dans ce cas, si on dénote par $\top$ le prédicat toujours vrai, l'ensemble des $e \in \Delta$ pour lesquels $\top$ est vraie est tout Δ. De même, si $\bot$ est le prédicat toujours faux, c'est-à-dire l'ensemble des $e \in \Delta$ pour lesquels $\bot$ est vraie, cet ensemble est $\emptyset$.

Si P et Q sont des prédicats, alors $(P \wedge Q)^{\mathcal{A}} = P^{\mathcal{A}} \cap Q^{\mathcal{A}}$ (pourquoi ?) et $(P \vee Q)^{\mathcal{A}} = P^{\mathcal{A}} \cup Q^{\mathcal{A}}$.
Et $(\neg P)^{\mathcal{A}} = \Delta \setminus P^{\mathcal{A}}$.

De même $(P \to Q)^{\mathcal{A}}$ est $(\neg P)^{\mathcal{A}} \cup Q^{\mathcal{A}} = (\Delta \setminus P^{\mathcal{A}}) \cup Q^{\mathcal{A}}$ et $(P \leftrightarrow Q)^{\mathcal{A}} = ((\Delta \setminus P^{\mathcal{A}}) \cup Q^{\mathcal{A}}) \cap ((\Delta \setminus Q^{\mathcal{A}}) \cup P^{\mathcal{A}}) = (\Delta \setminus (P^{\mathcal{A}} \cup Q^{\mathcal{A}})) \cup (P^{\mathcal{A}} \cap Q^{\mathcal{A}})$ (dessiner les patates !).

Conclusion : l'interprétation d'une formule peut être considérée comme le résultat d'une série d'opérations ensemblistes sur des sous-ensembles du domaine. Si ce résultat est vide, la formule est insatisfaisable, s'il est Δ c'est une tautologie.

2.2.2 Cas particulier : tables de vérité

On appelle **logique propositionnelle** le cas où on n'a ni prédicat (autre que les propositions), ni variable, ni fonction, ni quantificateur. En logique propositionnelle, on peut calculer la valeur vraie ou fausse d'un énoncé en appliquant les **tables de vérité** suivantes aux connecteurs :

A	$\neg A$
faux	vrai
vrai	faux

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$
faux	faux	faux	faux	vrai	vrai
faux	vrai	faux	vrai	vrai	faux
vrai	faux	faux	vrai	faux	faux
vrai	vrai	vrai	vrai	vrai	vrai

(Ces tables nous montrent que l'on pourrait s'amuser à définir 2^4 opérateurs binaires, mais traditionnellement on se limite aux 4 ci-dessus.)

Exemple : quelle est la valeur de $(P \wedge Q) \to (P \vee Q)$ selon celles de P et de Q ?

P	Q	$P \wedge Q$	$P \vee Q$	$(P \wedge Q) \to (P \vee Q)$
faux	faux	faux	faux	vrai
faux	vrai	faux	vrai	vrai
vrai	faux	faux	vrai	vrai
vrai	vrai	vrai	vrai	vrai

(On dira que cette formule est une **tautologie**.)

On voit que dans ce cas bien particulier (celui de la logique propositionnelle) une interprétation n'est rien d'autre qu'une combinaison de valeurs des propositions (et donc une ligne de la table de vérité).

Définition 2.2. *Un énoncé β est **conséquence** de α quand tout modèle de α est aussi modèle de β. On écrit alors $\alpha \models \beta$ (à ne pas confondre avec $\alpha \vdash \beta$ qui signifie que β peut être déduit de α par une suite d'inférences).*

Théorème 2.3. *Soient α et β deux formules. On a $\alpha \models \beta$ ssi $\alpha \to \beta$ est une tautologie (c'est-à-dire : l'implication est vraie pour toute interprétation).*

Démonstration. On va montrer que $\alpha \not\models \beta$ ssi $\alpha \to \beta$ n'est pas une tautologie. Ce dernier signifie qu'il existe $\mathcal{A}$ tel que $\mathcal{A} \models \neg(\alpha \to \beta)$, autrement dit que $\mathcal{A} \models (\alpha \wedge \neg\beta)$, autrement dit que $\mathcal{A} \models \alpha$ et $\mathcal{A} \not\models \beta$ ce qui est exactement dire que β n'est pas conséquence de α. □

2.2.3 Liens avec la langue naturelle

Dès lors qu'elle a une valeur de vérité, une phrase peut interpréter une formule (ou inversement : une formule peut formaliser une phrase).

Exemples : « Michel est élève de Télécom Bretagne », « je m'ennuie », « les Bretons n'aiment pas l'écotaxe ».

Contre-exemples : « 700 millions de Chinois, et moi, et moi, et moi », « courage, fuyons ! », « travailler plus pour gagner plus ».

Attention au comportement du connecteur $\to$:

« (Les coqs pondent des œufs) $\to$ (Télécom Bretagne est une grande école) » est vraie,

« (Les ânes volent) $\to$ (Télécom Bretagne est une discothèque) » est vraie aussi.

Remarquons à propos des quantificateurs que :
$\forall$ est le **quantificateur universel**, on l'utilise souvent suivi d'une implication :

$$\text{tout homme est mortel}$$
$$\forall x\,(\text{homme}(x) \rightarrow \text{mortel}(x)).$$

Attention à ne pas confondre avec $\forall x\,(\text{homme}(x) \wedge \text{mortel}(x))$, qui a une toute autre signification (laquelle ?).

$\exists$ est le **quantificateur existentiel**, on l'utilise souvent suivi d'une conjonction :

$$\text{Il existe un élève de Télécom dont le prénom est Xavier}$$
$$\exists x\,(\text{élèveTélécom}(x) \wedge \text{prénom}(x, \text{Xavier})).$$

À ne pas confondre avec $\exists x\,(\text{élèveTélécom}(x) \rightarrow \text{prénom}(x, \text{Xavier}))$ (pourquoi ?).

Exercice : comment interpréter $\forall x \exists y\,\text{aime}(x, y)$ et $\exists x \forall y\,\text{aime}(x, y)$?

2.2.4 L'égalité

L'opérateur $=$ dénote une relation spéciale, celle de l'**égalité** entre deux termes : $x = y$ signifie que x et y doivent être interprétés par la même valeur du domaine, quelque soit l'interprétation : $x^{\mathbf{I}} = y^{\mathbf{I}} = e \in \Delta$.

De même, sa négation $\neq$ signifie que deux termes ne pourront jamais être interprétés par les mêmes valeurs du domaine.

Pour dire, par exemple, que a a au moins deux frères, on écrira

$$\exists x\,\exists y\,(\text{frère}(x, a) \wedge \text{frère}(y, a) \wedge x \neq y)$$

dans le domaine des êtres humains (ou des nœuds d'un graphe, ou...).

2.2.5 Équivalences logiques

2.2.6 QCDC : Question cruciale du cours

On se pose la question cruciale de ce cours :

Question cruciale de ce cours 1. *Étant donnés une* ***base de connaissances*** *KB (c'est-à-dire la conjonction d'un ensemble d'énoncés vrais) et un énoncé α, est-ce qu'on a* $\text{KB} \models \alpha$ *?*

autrement dit : est-ce que tout modèle de KB est un modèle de α ? ou encore : est-ce que α est vrai dans tous les mondes possibles dans lesquels KB est vraie ? (et en particulier, le nôtre, si KB est une base de connaissances du monde réel).

Dans le cas de la logique propositionnelle, les modèles ne sont que des combinaisons de valeurs des propositions ; un algorithme (de « vérification de modèles ») peut alors procéder par la force brute et vérifier cette conséquence pour toute combinaison de valeurs de vérité des propositions de α. Il sera de complexité exponentielle $O(2^n)$ où n est le nombre de propositions.

Ceci ne s'applique plus quand on a des prédicats, fonctions et variables sur des domaines quelconques, potentiellement très grands ou infinis. Dans la suite on va découvrir d'autres méthodes pour répondre à la QCDC (= question cruciale du cours).

2.2.7 Exemple d'application de la logique du 1$^{\text{er}}$ ordre

L'exemple classique est celui des **nombres naturels**.

Comment définir $(\mathbb{N}, +)$? Il suffit de disposer d'un prédicat unaire NombreNaturel(), d'une constante 0 et d'un symbole de fonction $S()$ (= **successeur**) :

$\text{NombreNaturel}(0)$
$\forall n\, \text{NombreNaturel}(n) \rightarrow \text{NombreNaturel}(S(n))$
$\forall m \forall n, m \neq n \rightarrow S(m) \neq S(n)$ (sinon ?)
$\forall n\, 0 \neq S(n)$ (pourquoi ?)

Ces axiomes sont appelés **axiomes de Peano**.

L'*addition* se définit alors de la manière suivante :

$\forall m\, \text{NombreNaturel}(m) \rightarrow \text{Addition}(0, m) = m$
$\forall m \forall n, \text{NombreNaturel}(m) \wedge \text{NombreNaturel}(n) \rightarrow \text{Addition}(S(m), n) = S(\text{Addition}(m, n))$

Et à partir des nombres entiers on obtient les rationnels, les réels, les complexes. À partir de l'addition, la multiplication, la division, l'exponentielle, etc.

Une machine à qui on apprend les assertions ci-dessus est donc capable de calculer, elle a « connaissance » de l'**arithmétique**.

2.3 Satisfaisabilité

Un énoncé est une **tautologie** si toutes ses interprétations sont des modèles. Exemple : $A \vee \neg A$. On note $\models A$.

Un énoncé est **satisfaisable** [1] s'il possède au moins un modèle.

Un énoncé est **insatisfaisable** s'il ne possède aucun modèle. Exemple : $\alpha \wedge \neg\alpha$. On dit aussi que c'est une **contradiction**.

La notion d'insatisfaisabilité est très importante parce que la QCDC « KB $\models \alpha$ » est équivalente au problème de montrer que « KB $\wedge \neg\alpha$ est insatisfaisable ».

Un ensemble d'énoncés fermé pour la relation de conséquence est appelé une **théorie**, et les énoncés, des **théorèmes**.

2.3.1 Équivalence entre déduction et conséquence

Un théorème fondamental de la logique du 1er ordre dit que les deux approches : celle de la déduction $\alpha \vdash \beta$ (= on déduit β de α en faisant des inférences) et celle de la conséquence $\alpha \models \beta$ (= β est conséquence de α si tout modèle de α est modèle de β), sont équivalentes. Autrement dit, pour montrer que KB $\models \alpha$, qui est la **QCDC**, il suffit de déduire α de KB par une suite d'inférences.

Reste à trouver un système déductif qui soit facilement implémentable. Pour cela, nous allons étudier un certain nombre de méthodes pour normaliser et simplifier la formule, et ensuite nous allons appliquer la **méthode de résolution**.

2.4 Techniques de normalisation/simplification de formule

Le programme de cette section est le suivant :

1. Nous allons d'abord nous amuser à *substituer* des variables par d'autres variables ou par des termes.
2. La substitution va justement servir à *unifier* des formules ou des parties de la même formule, pour la rendre aussi simple que possible.
3. Puis nous allons « pousser » tous les quantificateurs vers le début de la formule, on obtient aussi la **forme normale prénexe** PNF.

1. Attention : « satisfiable » est un anglicisme (« faisable » ≠ « fiable »). À éviter, même si google-hits(satisfaisabilité) = 5 120 ≪ 10 700 = google-hits(satisfiabilité).

4. La partie sans quantificateurs peut être ré-écrite sous forme de conjonction et disjonctions ou, inversement, de disjonctions de conjonctions. Dans le premier cas on parlera de **forme normale conjonctive** CNF, et dans le deuxième de **forme normale disjonctive** DNF.
5. En combinant les formes normales prénexe et conjonctive, on obtient la **forme normale prénexe conjonctive** CPNF.
6. Enfin, on va se débarrasser des quantificateurs existentiels on effectuant une **skolémisation**.

À noter que les étapes 3–5 gardent l'équivalence des formules, ce n'est pas le cas de la 6.

On dit qu'on a **équivalence logique** entre deux énoncés s'ils possèdent exactement les mêmes modèles. Voici une liste d'équivalences qui nous seront bien utiles pour faire des calculs :

$\alpha \wedge \beta$	$\equiv$	$\beta \wedge \alpha$	**commutativité**
$\alpha \vee \beta$	$\equiv$	$\beta \vee \alpha$	commutativité
$\alpha \wedge (\beta \wedge \gamma)$	$\equiv$	$(\alpha \wedge \beta) \wedge \gamma$	**associativité**
$\alpha \vee (\beta \vee \gamma)$	$\equiv$	$(\alpha \vee \beta) \vee \gamma$	associativité
$\neg\neg\alpha$	$\equiv$	α	**double négation**
$\neg(\alpha \wedge \beta)$	$\equiv$	$\neg\alpha \vee \neg\beta$	**De Morgan**
$\neg(\alpha \vee \beta)$	$\equiv$	$\neg\alpha \wedge \neg\beta$	De Morgan
$\alpha \wedge (\beta \vee \gamma)$	$\equiv$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$	**distributivité**
$\alpha \vee (\beta \wedge \gamma)$	$\equiv$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$	distributivité
$\forall x\, \neg P$	$\equiv$	$\neg\exists x\, P$	
$\neg\forall x\, P$	$\equiv$	$\exists x\, \neg P$	
$\forall x\, P$	$\equiv$	$\neg\exists x\, \neg P$	
$\exists x\, P$	$\equiv$	$\neg\forall x\, \neg P$	

2.4.1 Substitution

On note $\textsc{Subst}(\{x/t\}, \alpha)$ la **substitution** $\{x/t\}$, où l'on remplace la variable x par le terme t, appliquée à α. Par exemple $\textsc{Subst}(\{x/t\}, P(x) \wedge P(t))$ est $P(t)$. Une substitution remplace toujours une variable par un terme (= une variable, une constante, une fonction de termes). Attention : une variable ne doit pas être remplacée par un terme qui la contient, sinon c'est la boucle infinie...

On a les deux règles d'inférence suivantes qui montrent tout l'intérêt de la substitution :

Proposition 2.4 (Règle d'instantiation universelle)**.**

$$\frac{\forall v\, \alpha}{\textsc{Subst}(\{v/g\}, \alpha)}$$

où g est un terme.

Plus intuitivement : si l'énoncé α est vrai pour tout le monde, alors elle est vraie pour un terme en particulier.

Proposition 2.5 (Règle d'instantiation existentielle)**.**

$$\frac{\exists v\, \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

où k est une constante qui n'apparaît nulle part ailleurs dans KB.

Plus intuitivement : si on nous dit qu'une valeur de v existe, alors autant lui donner un nom, même si on ne connaît pas explicitement sa valeur. Ce nom doit être nouveau !

Exemple : $\exists x \in \mathbb{R}$ tel que $\frac{d(x^y)}{dy} = x^y$, on peut lui donner un nom (par exemple e) mais il ne faut pas que ce soit un nom déjà pris (par exemple π, i, etc.). On appelle ce genre de constante, une **constante de Skolem**.

Le processus de **substitution** est très important. Imaginons que l'on ait des énoncés atomiques $p_1, \ldots, p_n$ tels que $\bigwedge_i p_i \to q$, et que l'on ait des énoncés $p'_1, \ldots, p'_n$ provenant des p_i après une substitution θ : $p'_i = \textsc{Subst}(\theta, p_i)$ (le même θ pour tous les i). Alors la règle du **modus ponens généralisé** nous dit que :

$$\frac{p'_1, \ldots, p'_n \qquad p_1 \wedge \cdots \wedge p_n \to q}{\textsc{Subst}(\theta, q)} \textit{ modus ponens} \text{ généralisé.}$$

Exemple :

$$\frac{\text{humain(Socrate)} \qquad \forall x\,(\text{humain}(x) \to \text{mortel}(x))}{\text{mortel(Socrate)}}\ m.\ p.\ \text{gén.}$$

Ici $\theta = \{x/\text{Socrate}\}$, $p_1 = \text{humain}(x)$, $p'_1 = \text{humain(Socrate)}$, $q = \text{mortel}(x)$ et donc $\textsc{Subst}(\theta, q) = \text{mortel}(\textbf{Socrate})$.

2.4.2 Unification

La substitution est utile seulement quand elle est effectuée dans le but de rendre les énoncés « similaires » et de permettre ainsi l'application du modus ponens généralisé. On appelle ce processus **unification** et la substitution obtenue, un **unificateur** :

$$\textsc{Unificateur}(p, q) = \theta \text{ tel que } \textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q).$$

Exemple :

$$\textsc{Unificateur}(\text{humain}(x), \text{humain(Socrate)}) = \{x/\text{Socrate}\}.$$

Exemple

$\mathcal{L}_0 = (R(f(g(x)), a, x), R(f(g(a)), a, b), R(f(y), a, z))$.
y une variable, $g(a)$ est un terme ne contenant pas y, on fait $\text{sub}_1 = \{y/g(a)\}$:
$\mathcal{L}_1 = (R(f(g(x)), a, x), R(f(g(a)), a, b), R(f(g(a)), a, z))$.
x une variable, a est un terme ne contenant pas x, on fait $\text{sub}_2 = \{x/a\}$:
$\mathcal{L}_2 = (R(f(g(a)), a, a), R(f(g(a)), a, b), R(f(g(a)), a, z))$.
a et b sont des termes, on ne peut pas les substituer.
Conclusion : $\mathcal{L}_0$ n'est pas unifiable.

2.4.3 Forme normale prénexe

On dit que l'énoncé φ est en **forme normale prénexe** (**PNF**) s'il est de la forme

$$Q_1 x_1 \cdots Q_n x_n \psi$$

où Q_i sont des quantificateurs et ψ est un énoncé sans quantificateurs.

Exemples : $\forall x \exists y\,(f(x) = y)$ est en PNF. $\neg\forall x \exists y\, P(x, y, z)$ ne l'est pas (pourquoi ?), ni $\exists x \forall y\, \neg P(x, y, z) \wedge \forall x \exists y\, Q(x, y, z)$.

Théorème 2.6. *Pour tout énoncé de logique du 1er ordre, il existe un énoncé en PNF qui lui est équivalent.*

L'algorithme pour l'obtenir consiste à renommer les variables liées et à déplacer les quantificateurs de variables de manière à ce qu'ils englobent des parties de l'énoncé dans lesquelles ces variables n'apparaissent pas.

Exemple d'application de l'algorithme PNF :

$$\begin{aligned}\varphi &\equiv \exists x \forall y\, \neg P(x, y, z) \wedge \forall x \exists y\, Q(x, y, z)\\ &\equiv \exists x \forall y\, \neg P(x, y, z) \wedge \forall u \exists v\, Q(u, v, z)\\ &\equiv \exists x \forall y \forall u \exists v\, (\neg P(x, y, z) \wedge Q(u, v, z)).\end{aligned}$$

2.4.4 CNF

On dit qu'un énoncé est en **CNF** (**forme normale conjonctive**) s'il s'écrit sous forme d'une conjonction de disjonctions ($(F_1 \vee F_2) \wedge (G_1 \vee \neg G_2)$, etc.).

Proposition 2.7. *Tout énoncé (sans quantificateurs) est équivalent à un énoncé en CNF.*

Voici l'algorithme pour l'obtenir :

1. remplacer les $\leftrightarrow$ par des conjonctions de $\rightarrow$;
2. remplacer les $\alpha \rightarrow \beta$ par des $\neg\alpha \vee \beta$;
3. faire entrer les négations dans les parenthèses pour obtenir des littéraux (autrement dit : appliquer les lois de **De Morgan**) ;
4. utiliser la propriété distributive.

Exemple :

Énoncé de départ : $B \leftrightarrow (F \vee G)$.
Élimination du $\leftrightarrow$: $B \rightarrow (F \vee G) \wedge (F \vee G) \rightarrow B$.
Élimination du $\rightarrow$: $(\neg B \vee F \vee G) \wedge (\neg(F \vee G) \vee B)$.
Faire entrer les $\neg$: $(\neg B \vee F \vee G) \wedge ((\neg F \wedge \neg G) \vee B)$.
Appliquer la **distributivité** : $(\neg B \vee F \vee G) \wedge (\neg F \vee B) \wedge (\neg G \vee B)$.

2.4.5 CPNF

Quand un énoncé est sous forme prénexe et sa partie sans quantificateurs sous CNF, on dira qu'il est sous **CPNF** (**forme normale prénexe conjonctive**).

Théorème 2.8. *Pour tout énoncé de logique du 1^{er} ordre, il existe un énoncé en CPNF qui lui est équivalent.*

2.4.6 Skolémisation

Thoralf Skolem (1887–1963) était un mathématicien norvégien.

Définition 1. *Un énoncé est sous **forme normale de Skolem** (**SNF**) s'il est en CPNF avec uniquement des quantificateurs universels.*

Il existe un algorithme qui fournit pour tout énoncé φ, un énoncé φ_S sous SNF que l'on appelle sa **skolémisation** :

1. à partir de φ, obtenir $\varphi' = Q_1 \cdots Q_m \, \varphi(x_1, \ldots, x_m)$ en CPNF ;
2. si tous les Q_i sont $\forall$, φ' est en SNF ;
3. sinon, à partir de φ' obtenir $s(\varphi')$ qui a un $\exists$ en moins que φ' :
   – si le $\exists$ est en première position ($\exists x_1$), on supprime $\exists x_1$ et on remplace x_1 dans l'énoncé par une constante c qui n'apparaît pas dans φ' ;
   – si le $\exists$ est en i-ème position, on supprime $\exists x_i$ et on remplace x_i par une fonction $(i-1)$-aire $f(x_1, \ldots, x_{i-1})$ qui n'apparaît pas dans φ' ;
4. en répétant cette étape k fois (pour k quant. exist.) on obtient $\overbrace{s \circ \cdots \circ s}^{k \text{ fois}}(\varphi')$, qui est en SNF.

Théorème 2.9. *Soit φ un énoncé et φ^S son skolémisé. Alors φ est satisfaisable ssi φ^S est satisfaisable.*

Exemple de skolémisation

- En français : « Tous ceux qui aiment tous les animaux sont aimés par qqun ».
- En logique : $\forall x\,((\forall y\,\text{Animal}(y) \to \text{Aime}(x,y)) \to (\exists y\,\text{Aime}(y,x)))$;
- on élimine la 2e implication : $\forall x\,(\neg((\forall y\,\text{Animal}(y) \to \text{Aime}(x,y))) \vee (\exists y\,\text{Aime}(y,x)))$;
- on élimine la 1re implication : $\forall x\,(\neg(\forall y\,(\neg(\text{Animal}(y)) \vee \text{Aime}(x,y))) \vee (\exists y\,\text{Aime}(y,x)))$;
- on gère les négations : $\forall x\,(\exists y\,\neg(\neg\text{Animal}(y) \vee \text{Aime}(x,y))) \vee (\exists y\,\text{Aime}(y,x)))$;
- et encore : $\forall x\,(\exists y\,(\text{Animal}(y) \wedge \neg\text{Aime}(x,y))) \vee (\exists y\,\text{Aime}(y,x)))$;
- on skolemise : $\forall x\,(\text{Animal}(f(x)) \wedge \neg\text{Aime}(x,f(x))) \vee \text{Aime}(g(x),x)$ où f et g sont des fonctions de Skolem ;
- on distribue la disjonction : $\forall x\,(\text{Animal}(f(x)) \vee \text{Aime}(g(x),x)) \wedge (\neg\text{Aime}(x,f(x)) \vee \text{Aime}(g(x),x))$.

Cet énoncé est plus opaque que celui du début, mais aussi plus maniable pour la machine.

(Ici $f(x)$ est un «animal potentiellement non-aimé par x» et $g(x)$ est «quelqu'un qui peut aimer x».)

2.5 Résolution

La **résolution** est une règle d'inférence facilement implémentable. Elle fonctionne sur des formules skolemisées.

Soient deux disjonctions telles que la première contient un prédicat F et la deuxième la négation $\neg F$ de celui-ci. Alors la résolution consiste à « éliminer » F et $\neg F$:

$$\frac{F \vee G \vee \neg H \qquad K \vee \neg F \vee L}{G \vee \neg H \vee K \vee L}\ \text{résolution.}$$

La nouvelle disjonction obtenue est appelée **résolvant** des deux premières.

En appliquant de manière répétée la règle de résolution, on obtient de plus en plus de résolvants. Le processus s'arrête en un temps fini puisque le nombre total de prédicats diminue de deux à chaque étape.

Proposition 2.10. *Si l'**ensemble vide** $\emptyset$ figure parmi les résolvants obtenus, alors l'énoncé de départ est une contradiction.*

Pour vérifier la **QCDC** ($\text{KB} \models \alpha$), on prend $\text{KB} \wedge \neg\alpha$, on le ré-écrit en SNF et on applique la règle de résolution à toutes les paires de disjonctions. À celles-ci viennent s'ajouter les résolvants et on continue à effectuer des résolutions jusqu'à ce qu'il n'y reste plus aucune disjonction non traitée. Le résultat obtenu va dépendre des choix de paires à résoudre, alors on recommence en faisant d'autres choix, etc. La formule $\text{KB} \wedge \neg\alpha$ est insatisfaisable si et au seulement si au cours de ces opérations on a, à un moment donné, obtenu un résolvant vide.

(Ce qui veut dire que si on veut montrer l'**insatisfaisabilité** on peut s'arrêter dès l'arrivée du $\emptyset$, par contre si on veut montrer la satisfaisabilité on doit traiter *toutes* les paires de clauses, clauses et résolvants, et résolvants entre eux.)

À noter que souvent pour faire une élimination on devra d'abord unifier les prédicats. Ainsi, par exemple, si la base de connaissances KB est

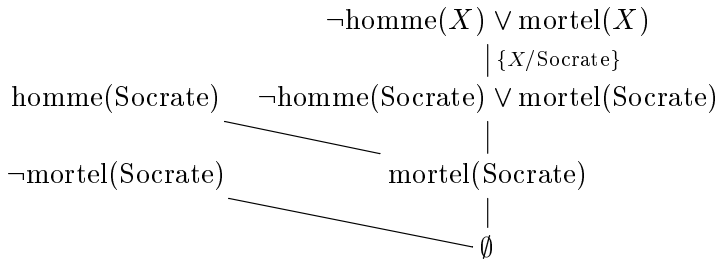
$$\begin{cases} \text{homme(Socrate)} \\ \forall X\ \text{homme}(X) \to \text{mortel}(X) \end{cases}$$

et α est « mortel(Socrate) », alors la résolution revient à montrer que la formule $\text{KB} \wedge \neg\alpha$, c'est-à-dire

$$\text{homme(Socrate)} \wedge (\neg\text{homme}(X) \vee \text{mortel}(X)) \wedge \neg\text{mortel(Socrate)}$$

est insatisfaisable. Mais on voit tout de suite qu'on ne peut pas éliminer homme(Socrate) et $\neg$homme(X), ni mortel(X) et $\neg$mortel(Socrate).

C'est là que l'*unification* entre en jeu. On aura donc la résolution suivante :



2.5.1 Exemple de résolution

Soit la base de connaissances KB suivante :

$$\begin{cases} \text{pa}(\text{Robert}, \text{Kevin}) \\ \text{pa}(\text{Robert}, \text{Samantha}) \\ \text{pa}(\text{Samantha}, \text{Thomas}) \\ \text{pa}(\text{Doris}, \text{Jimmy}) \\ \text{pa}(\text{Boris}, \text{Jimmy}) \\ \text{pa}(\text{Boris}, \text{Elisabeth}) \\ \text{pa}(\text{Jimmy}, \text{Thomas}) \\ \text{pa}(\text{Samantha}, \text{Samuel}) \\ \text{pa}(\text{Jimmy}, \text{Samuel}) \\ \text{pa}(\text{Zelda}, \text{Max}) \\ \text{pa}(\text{Samuel}, \text{Max}) \\ \forall X, Y, Z\, \text{pa}(X, Z) \wedge \text{pa}(Z, Y) \rightarrow \text{gp}(X, Y) \end{cases}$$

où $\text{pa}(X, Y)$ signifie que X est parent de Y, et $\text{gp}(X, Y)$ que X est grand-parent de Y.

Soit α la formule suivante :

$$\exists X\, \text{gp}(\text{Robert}, X) \wedge \text{pa}(X, \text{Max}).$$

On pose la **QCDC** : $\text{KB} \models \alpha$? On sait quelle est équivalente à montrer que $\text{KB} \wedge \neg\alpha$ est insatisfaisable.

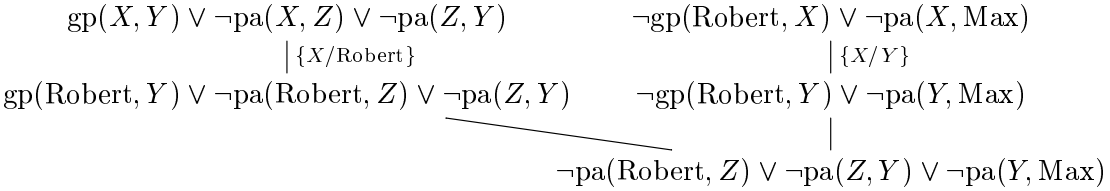
On prend donc $\text{KB} \wedge \neg\alpha$ et on lui applique une résolution.

À noter que comme $\neg\alpha$ est

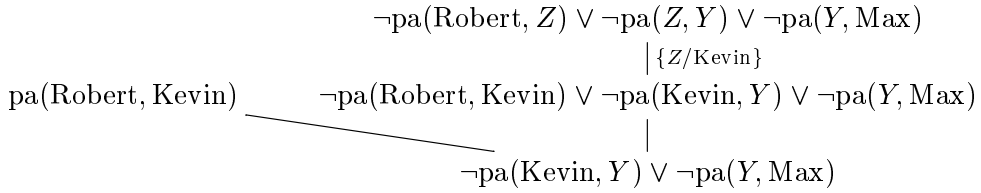
$$\forall X\, \neg\text{gp}(\text{Robert}, X) \vee \neg\text{pa}(X, \text{Max}),$$

on a des quantificateurs universels partout, donc $\text{KB} \wedge \neg\alpha$ est déjà skolemisée, on peut commencer à éliminer des prédicats dans les paires de disjonctions.

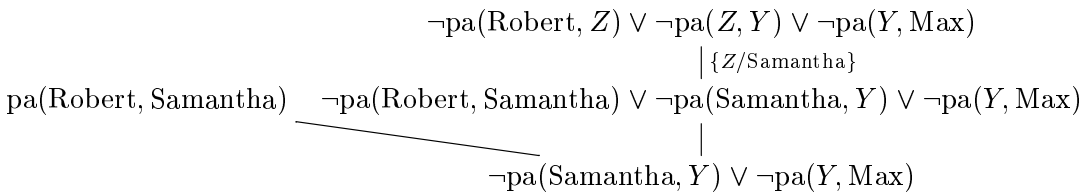
Allons-y :



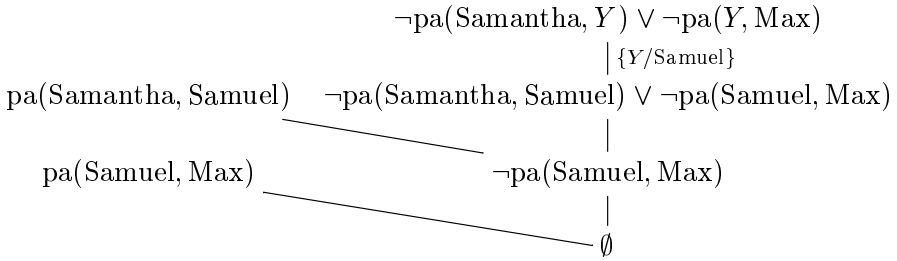
Ensuite il faut éliminer $\neg\text{pa}(\text{Robert}, Z)$. Un premier choix serait d'éliminer avec $\text{pa}(\text{Robert}, \text{Kevin})$. Voici la résolution :



Donc, pour avancer, la prochaine étape serait l'élimination de $\neg\text{pa}(\text{Kevin}, Y)$. Sauf que dans KB, il n'y a aucune formule avec la constante « Kevin » en première position. On s'aperçoit donc que l'on fait fausse route. Qu'à cela ne tienne ! On revient un pas en arrière et on essaie d'éliminer $\neg\text{pa}(\text{Robert}, Z)$ par le prédicat $\text{pa}(\text{Robert}, \text{Samantha})$. On a donc :



Cette fois-ci on a plus de possibilités pour éliminer Samantha : il y a Thomas et Samuel. Comme cela a été le cas avec Kevin, Thomas va également s'avérer être un mauvais choix, et c'est avec Samuel que l'on ira jusqu'au bout :



et donc $\text{KB} \wedge \neg\alpha$ est insatisfaisable, et donc α est conséquence de KB, la réponse à la **CQDC** est affirmative.

Notons que les substitutions effectuées lors de cette résolution nous fournissent aussi le renseignement à la question auxiliaire (mais tout aussi importante) : «finalement, qui est le X tel que $X\,\text{gp}(\text{Robert}, X) \wedge \text{pa}(X, \text{Max})$?»

2.5.2 Monty Python : brûlez la sorcière !

Dans leur film *Monty Python : Sacré Graal*, les Monty Python utilisent une logique un peu particulière pour prouver qu'une — au demeurant, très charmante — jeune femme (jouée par Connie Booth) est une sorcière. Nous allons montrer, en nous servant de la méthode de résolution, que leur raisonnement est fallacieux.

— (1) Que fait-on avec les sorcières ? On les brûle !
— (2) Que brûle-t-on d'autre ? Le bois !
— (3) Que fait le bois ? Il flotte.
— (4) Qui d'autre flotte ? Un canard.
— (5) Non énoncé, mais sous-entendu : tout ce qui a le même poids qu'un objet qui flotte, flotte aussi
— (E) Expérimentalement, on montre que la jeune femme a le même poids que le canard
— (α) Conclusion : elle est une sorcière !

Qui peut être formalisé par :

FIGURE 2.1 – Un extrait du film *Monty Python : Sacré Graal*

— (1) $\forall X, \mathrm{Wi}(X) \to \mathrm{B}(X)$
— (2) $\forall X, \mathrm{Wo}(X) \to \mathrm{B}(X)$
— (3) $\forall X, \mathrm{Wo}(X) \to \mathrm{F}(X)$
— (4) $\mathrm{F}(D)$
— (5) $\forall X, Y, (\mathrm{F}(X) \wedge (w(X) = w(Y))) \to \mathrm{F}(Y)$
— (E) $w(\mathrm{JF}) = w(\mathrm{D})$
— (α) $\mathrm{Wi}(\mathrm{JF})$

Autrement dit :
— (1) $\forall X, \neg\mathrm{Wi}(X) \vee \mathrm{B}(X)$
— (2) $\forall X, \neg\mathrm{Wo}(X) \vee \mathrm{B}(X)$
— (3) $\forall X, \neg\mathrm{Wo}(X) \vee \mathrm{F}(X)$
— (4) $\mathrm{F}(\mathrm{D})$
— (5) $\forall X, Y, \neg\mathrm{F}(X) \vee (w(X) \neq w(Y)) \vee \mathrm{F}(Y)$
— (E) $w(\mathrm{JF}) = w(\mathrm{D})$
— ($\neg\alpha$) $\neg\mathrm{Wi}(\mathrm{JF})$

Comme (1) et (2) contiennent $\mathrm{B}(X)$, et qu'il n'y a aucun $\neg\mathrm{B}(X)$ dans aucune formule, elles sont inutiles, on les enlève. Idem pour (3) qui contient $\neg\mathrm{Wo}(X)$ et ($\neg\alpha$) qui contient $\neg\mathrm{Wi}(W)$. Ne restent donc plus que (4), (5) et (E). Voici les seules combinaisons possibles :
— (4) et (5) : $\forall Y, (w(\mathrm{D}) \neq w(Y)) \vee \mathrm{F}(Y)$ (R1)
— (E) et (5) : $\neg\mathrm{F}(\mathrm{JF}) \vee \mathrm{F}(\mathrm{D})$ (R2)
— (R1) et (R2) : $(w(\mathrm{D}) \neq w(\mathrm{JF})) \vee \mathrm{F}(\mathrm{D})$ (R3)

et donc on ne trouve pas le parmi les résolvants, donc $\mathrm{KB} \wedge \neg\alpha$ est satisfaisable, et $\mathrm{Wi}(\mathrm{JF})$ n'est pas conséquence de la base de connaissances, on ne peut donc — hélas — pas en conclure que la fille était une sorcière.

Le *modus bogus*]

Notons, sur un registre humoristique, que Jesse Hoey, de l'université de Waterloo au Canada, montre que si on introduit une nouvelle règle d'inférence, le **modus bogus**, qui s'énonce comme suit : «si on a Y et $X \to Y$, alors on a X», alors le raisonnement des Monty Python est correct.

Voici comment il procède :

— Il montre d'abord le théorème $\forall X, \mathrm{Wo}(X) \to \mathrm{Wi}(X)$ (= tout ce qui est de bois, est une sorcière). En effet, supposons $\mathrm{Wo}(X)$ pour un X donné. Alors par un **modus ponens** avec (2), on obtient $\mathrm{B}(X)$ et par un *modus bogus* avec (1), on obtient $\mathrm{Wi}(X)$.
— Ensuite il montre la proposition 1 : F(JF) (= la jeune fille flotte). En effet, par (4) et (E) on obtient la conjonction $\mathrm{F(D)} \wedge (w(\mathrm{JF}) = w(\mathrm{D}))$. Celle-ci, par un *modus ponens* avec (5) nous donne F(JF).
— Ensuite il montre la proposition 2 : Wo(JF) (= la jeune fille est de bois). Par en combinant la proposition 1 et le (3) par un *modus bogus*, on obtient Wo(JF).
— Enfin, en combinant la proposition 2 et le théorème, par un *modus ponens*, on a bien Wi(JF), donc c'est bel et bien une sorcière, malgré son apparence angélique.

AVERTISSEMENT : le *modus bogus* relève de l'humour, et son utilisation dans quelque contexte que ce soit (que ce soit dans sa vie professionnelle d'ingénieur, ou lors d'examen du module, ou ailleurs) sera considérée comme une banale erreur, et sanctionnée comme telle !

2.6 Exercices

Exercice 1 Exercice 0 (Proverbes)

Écrire en utilisant le formalisme de la logique du 1er ordre les proverbes :
— L'amour est aveugle.
— Chose promise, chose due.
— Faute avouée est à demi pardonnée.
— Il n'est point de sot métier.
— Il n'y a que la vérité qui blesse.
— Connais-toi toi-même.
— Fais ce que je dis, pas ce que je fais.
— Je ne sais qu'une chose, c'est que je ne sais rien.

Exercice 2 Exercice 0′ (Théorèmes)

Écrire en utilisant le formalisme de la logique du 1er ordre les théorèmes suivants. Bien noter qu'est-ce qui est fonction, prédicat, variable et constante.
— Théorème de Thalès. — Soit un triangle ABC et deux points D et E situés sur AB et AC respectivement. Alors $\frac{AD}{AB} = \frac{AE}{AC} = \frac{DE}{BC}$.
— Théorème de Cauchy. — Soit U un ouvert simplement connexe de $\mathbb{C}$, $f\colon U \to \mathbb{C}$ une fonction continue sur U et possédant une dérivée complexe sauf éventuellement en un nombre fini de points et γ un lacet de longueur fini dans U, alors $\int_\gamma f(z)\,dz = 0$.
— Théorème des nombres premiers. — Si $\pi(x)$ est le nombre de nombres premiers inférieurs ou égaux à x, alors $\lim_{x\to\infty} \pi(x)\frac{\log(x)}{x} = 1$. (Écrire d'abord une version avec une fonction pi(x) pour $\pi(x)$, ensuite une version où pi() est décrite par une fonction cardinal() et des prédicats nombre-premier() et inf-ou-égal().)

Exercice 3 Exercice 1

Lesquelles parmi les affirmations suivantes sont correctes ?

1. faux $\models$ vrai
2. vrai $\models$ faux
3. $(A \wedge B) \models (A \leftrightarrow B)$
4. $A \leftrightarrow B \models A \vee B$
5. $A \leftrightarrow B \models \neg A \vee B$
6. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B \vee C) \wedge (B \wedge C \wedge D \to E)$

7. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B) \wedge (\neg D \vee E)$
8. $(A \vee B) \wedge \neg(A \rightarrow B)$ est satisfaisable
9. $(A \wedge B) \rightarrow C \models (A \rightarrow C) \vee (B \rightarrow C)$
10. $(C \vee (\neg A \wedge \neg B)) \equiv ((A \rightarrow C) \wedge (B \rightarrow C))$
11. $(A \leftrightarrow B) \wedge (\neg A \vee B)$ est satisfaisable

Exercice 4 Exercice 2

Décider lesquels parmi les énoncés suivants sont des tautologies, des contradictions ou ni l'un ni l'autre.

1. Fumée $\rightarrow$ fumée
2. Fumée $\rightarrow$ feu
3. (Fumée $\rightarrow$ feu) $\rightarrow$ ($\neg$fumée $\rightarrow$ $\neg$feu)
4. Fumée $\vee$ feu $\vee$ $\neg$feu
5. ((Fumée $\wedge$ chaleur) $\rightarrow$ feu) $\leftrightarrow$ ((fumée $\rightarrow$ feu) $\vee$ (chaleur $\rightarrow$ feu))
6. Grand $\vee$ stupide $\vee$ (grand $\rightarrow$ stupide)
7. (Grand $\wedge$ stupide) $\vee$ $\neg$stupide

Exercice 5 Exercice 3

Écrire sous forme d'énoncé de logique propositionnelle et vérifier :

1. Quand la temperature est constante et la pression est constante, il ne pleut pas. Donc si la temperature est constante et qu'il pleut, alors la pression a nécessairement varié.
2. Quand les gens ont faim, ils mangent. Quand Jean mange, il porte son meilleur constume. Aujourd'hui Jean n'a pas faim, donc il ne porte pas son meilleur costume. (Utiliser une résolution.)

Exercice 6 Exercice 4

Soit $P(x)$ le prédicat «a été à Prague», et D le domaine de vos camarades. Exprimer chacune des quantifications suivantes en français :

1. $\exists x, P(x)$
2. $\forall x, P(x)$
3. $\neg\exists x, P(x)$
4. $\neg\forall x, P(x)$
5. $\exists x, \neg P(x)$
6. $\forall x, \neg P(x)$
7. $\neg\exists x, \neg P(x)$
8. $\neg\forall x, \neg P(x)$

Lesquelles signifient la même chose ?

Exercice 7 Exercice 5

Donner la valeur de vérité de chacun des énoncés suivants, en considérant que le domaine est $\mathbb{R}$:

1. $\exists x, (x^2 = 2)$
2. $\exists x, (x^2 = -1)$
3. $\forall x, (x^2 + 2 \geq 1)$
4. $\forall x, (x^2 \neq x)$.

Exercice 8 Exercice 6

Soit le vocabulaire suivant :
— $O(p, o)$ prédicat, la personne p occupe le poste p
— $C(p_1, p_2)$ prédicat, la personne p_1 est client de la personne p_2
— $B(p_1, p_2)$ prédicat, la personne p_1 est un supérieur hiérarchique (B comme «boss») de la personne p_2
— D, Ch, Av, Ac constantes, des métiers : docteur, chirurgien, avocat, acteur
— E, J constantes, des personnes : Émilie, Joël.

Utiliser ces symboles pour écrire les énoncés suivants en logique du 1er ordre :

1. Émilie est soit chirurgienne, soit avocate
2. Joël est acteur, mais il a aussi un autre job
3. Tous les chirurgiens sont docteurs
4. Émilie a un boss qui est avocat
5. Il existe un avocat dont tous les clients sont des docteurs
6. Tout chirurgien a un avocat

Exercice 9 Exercice 7

Soient les nombres naturels $\mathbb{N}$ avec le prédicat $<$, les fonctions $+$ et $\times$, et les constantes 0 et 1. Écrire en logique du 1er ordre :

1. la propriété de parité (Pair(x))
2. la propriété d'être nombre premier (Premier(x))
3. la conjecture de Goldbach [2].

Exercice 10 Exercice 8

En utilisant les prédicats 1-aires ADN, Personne, Père, Mère et le prédicat 3-aire DérivéDe, écrire en logique du 1er ordre la phrase : l'ADN d'une personne est unique et dérivé de ceux de ses parents.

Exercice 11 Exercice 9 (Sur l'alternance des quantificateurs)

Soient les quatre énoncés :

1. $\forall x, \forall y, \exists z, (\text{aime}(x, y) \wedge ((z \neq y) \rightarrow \neg\text{aime}(y, z)))$
2. $\exists x, \forall y, \forall z, (\text{aime}(x, y) \wedge ((z \neq y) \rightarrow \neg\text{aime}(y, z)))$
3. $\exists x, \exists y, \forall z, (\text{aime}(x, y) \wedge ((z \neq y) \rightarrow \neg\text{aime}(y, z)))$
4. $\forall x, \exists y, \forall z, (\text{aime}(x, y) \wedge ((z \neq y) \rightarrow \neg\text{aime}(y, z)))$.

Déterminer la valeur de vérité de ces énoncés pour les quatre domaines suivant :

1. un monde avec une seule personne, qui s'aime
2. un monde avec Yorick et Zelda, où Yorick aime Zelda, Zelda s'aime, et c'est tout
3. un monde avec pas mal de gens, y compris FH, VT et SR. Tout le monde s'aime soi-même, tout le monde aime FH, tout le monde aime VT sauf SR, tout le monde aime SR sauf FH et VT. Tracer un graphe des amours.

Exercice 12 Exercice 10 (Dragons et touristes)

Aucun dragon fort ne peut ne pas souffler le feu.
Un dragon rusé a toujours des cornes.
Aucun dragon faible n'a des cornes.
Les touristes ne chassent que les dragons ne soufflant pas de feu.
Un dragon rusé doit-il craindre les touristes ?

2. Tout nombre pair est somme de deux nombres premiers : $2 = 1 + 1$, $4 = 2 + 2$, $6 = 1 + 5$, $8 = 7 + 1$, $10 = 7 + 3$, $12 = 7 + 5$, $14 = 7 + 7$, ...

Exercice 13 Exercice 11 (Affaire criminelle)

Tous ceux qui aiment tous les animaux sont aimés par qqun
Quiconque tue un animal n'est aimé par personne
Jack aime tous les animaux
C'est soit Jack soit Curiosité qui a tué le chat appelé Luna
Montrer que c'est Curiosité qui a tué le chat.

Exercice 14 Exercice 12

Comment montrer en utilisant la méthode de résolution qu'un énoncé α est une tautologie ? Une contradiction ?

Exercice 15 Exercice 13

Unifier, si possible, les paires d'énoncés suivantes :

1. $P(A, A, B)$, $P(x, y, z)$
2. $Q(y, G(A, B))$, $Q(G(x, x), y)$
3. PlusVieux(Père$(y), y)$, PlusVieux(Père(x), Jerry)
4. Connaît(Père$(y), y)$, Connaît(x, x).

Exercice 16 Exercice 14 (Casse-tête démoniaque)

Vrai ou faux ? Expliquer

1. $\exists x, x =$ Démon est une tautologie.
2. Tout énoncé existentiellement quantifié est vrai dans tout modèle contenant exactement un objet.
3. $\forall x, y\ x = y$ est satisfaisable.

Exercice 17 Exercice 15 (Questions pièges)

Soit une KB ne contenant que le seul énoncé $\exists x$, AussiHautQue$(x,$ Everest). Lesquels parmi les énoncés suivants peuvent être dérivés de KB en appliquant une instantiation existentielle ?

1. AussiHautQue(Everest, Everest).
2. AussiHautQue(Kilimandjaro, Everest).
3. AussiHautQue(Kilimandjaro, Everest) $\wedge$ AussiHautQue(MontsD'Arrhée, Everest).

Chapitre 3

Logiques de description

3.1 Les bases

Les **logiques de description** sont des variantes de la logique du 1er ordre qui forment des compromis sur deux tableaux :

1. elles ajoutent des nouvelles notations pour améliorer l'expressivité du langage (par ex. la possibilité de dire qu'il existe exactement n éléments ayant telle propriété) ;
2. elles limitent les possibilités de la logique du 1er ordre pour la rendre décidable.

On a trois types d'objets :

1. des **individus** (ce qui correspond aux constantes de la logique du 1er ordre),
2. des **concepts** ou **classes** (des prédicats unaires, dont l'interprétation ensembliste correspond à des ensembles d'individus),
3. des **rôle** ou **propriétés** (des prédicats binaires sur les individus) ;

On définit trois types de formules :

1. celles de l'**ABox**, «A» comme «assertion», qui décrivent des concepts et des rôles (prédicats unaires et binaires sur des constantes) ;
2. celles de la **TBox**, «T» comme «terminologie», qui décrivent des relations entre concepts ;
3. celles de la **RBox**, «R» comme «relation», qui décrivent une hiérarchie des rôles, des compositions de rôles et des relations entre rôles, comme l'exclusion mutuelle, la réflexivité, la symétrie, la transitivité, etc.

3.1.1 ABox

Une **assertion de concept** est un prédicat unaire du type ÉlèveTélécom(xavier).
Une **assertion de rôle** est un prédicat binaire Père(andré,mathilde).
Dans les logiques de description on ne fait pas l'**hypothèse de l'unicité des noms** : deux individus de même nom peuvent avoir le même référent, on écrira cloclo $\approx$ claudeFrançois, et samsonFrançois $\not\approx$ claudeFrançois.

3.1.2 TBox

Puisqu'on peut interpréter les **concepts** (prédicats unaires) comme des ensembles, on peut aussi utiliser des relations de théorie d'ensembles :
Mère $\sqsubset$ Parent, qui équivaut à $\forall X\,\text{Mère}(X) \rightarrow \text{Parent}(X)$;
Personne $\equiv$ Humain, qui équivaut à $\forall X\,\text{Personne}(X) \leftrightarrow \text{Humain}(X)$.

3.1.3 RBox

La **hiérarchie des concepts** induit une **hiérarchie des rôles** :

ParentDe $\sqsubset$ AncêtreDe, qui équivaut à $\forall X, Y$ ParentDe$(X, Y) \rightarrow$ AncêtreDe(X, Y).

Une relation binaire peut se combiner avec une autre : sachant que le frère d'un père est un oncle, on peut écrire

FrèreDe $\circ$ ParentDe $\sqsubset$ OncleDe, qui équivaut à $\forall X, Y, Z$ FrèreDe$(X, Y) \wedge$ParentDe$(Y, Z) \rightarrow$ OncleDe(X, Z).

Enfin, on peut donner des caractéristiques générales des concepts : *Disjoint*(ParentDe, EnfantDe), ce qui équivaut à $\forall X, Y$ ParentDe$(X, Y) \rightarrow \neg$Parent(Y, X).

3.2 Constructeurs et restrictions

3.2.1 Constructeurs de concepts

Toujours en mimant la théorie des ensembles, on peut écrire Mère $\equiv$ Femme $\sqcap$ Parent, qui équivaut à $\forall X$ Mère$(X) \leftrightarrow$ Femme$(X) \wedge$ Parent(X),

ou Parent $\equiv$ Père $\sqcup$ Mère, qui équivaut à $\forall X$ Parent$(X) \leftrightarrow$ Mère$(X) \wedge$ Père(X),

la négation correspond au complémentaire d'un ensemble : $\neg$Célibataire est équivalent à Marié.

L'ensemble complet correspond à un prédicat qui est toujours vrai, on l'écrit $\top$, pour exprimer une partition on écrira $\top \sqsubset$ Homme $\sqcup$ Femme.

L'ensemble vide correspond à un prédicat qui est toujours faux, on l'écrit $\bot$, pour exprimer une exclusion mutuelle on écrira Homme $\sqcap$ Femme $\sqsubset \bot$.

3.2.2 Restrictions de rôle

Si jusqu'à maintenant les notations semblent claires et intuitives, dans ce qui va suivre elles vont se compliquer, attention à bien comprendre de quoi il s'agit !

Restriction existentielle

Imaginons qu'on a un rôle Parent(X, Y) et que l'on cherche à caractériser les X qui sont parents. Il s'agit donc de dire «je cherche les X pour lesquels $\exists Y$ tel que Parent(X, Y)», on écrira $\exists$Parent.$\top$; le $\top$ signifie qu'on prend les X pour lesquels il existe un Y, sans les filtrer davantage.

Si je cherchais ceux qui sont des parents d'au moins une fille, j'écrirais $\exists$Parent.Fille.

Restriction universelle

De même, ceux qui n'ont que des filles : $\forall$Parent.Fille.

Mais que signifie alors $\forall$Parent.$\top$?

Pour le savoir, prenons les définitions formelles des sémantiques de ces notations.

Soit R un rôle, et $R^{\mathbf{I}}$ son interprétation. Rappelons que dans une interprétation ensembliste, $R^{\mathbf{I}}$ devient un ensemble de paires d'éléments $R^{\mathbf{I}} = \{(x^{\mathbf{I}}, y^{\mathbf{I}}), \ldots\}$ du domaine Δ. On dira que $y^{\mathbf{I}}$ est un R-**successeur** de $x^{\mathbf{I}}$ si la paire $(x^{\mathbf{I}}, y^{\mathbf{I}})$ appartient à $R^{\mathbf{I}}$.

Soit C un concept (et donc $C^{\mathbf{I}}$ est un sous-ensemble de Δ). Alors l'interprétation de $\exists R.C$ est $\{x^{\mathbf{I}} \mid$ quelques successeurs de $x^{\mathbf{I}}$ sont dans $C^{\mathbf{I}}\}$.

Et celle de $\forall R.C$ est $\{x^{\mathbf{I}} \mid$ tous les successeurs de $x^{\mathbf{I}}$ sont dans $C^{\mathbf{I}}\}$.

Donc, quelle est l'interprétation de $\forall$Parent.$\top$?

Réponse : «tous ses successeurs sont dans Δ» demeure vraie aussi.

C'est le domaine tout entier. Effectivement, si $x^{\mathbf{I}}$ a des successeurs alors ils sont forcément dans l'interprétation de $\top$ qui est le domaine Δ. Et si $x^{\mathbf{I}}$ n'a pas de successeur alors l'assertion

Restrictions au-plus et au-moins

De même que l'on peut demander au moins un successeur ou tous les successeurs, on peut aussi spécifier «au moins n successeurs» : $\geq nR.C$, ainsi qu'«au plus n successeurs» : $\leq nR.C$.

Réflexivité

Une autre notation (attention : les apparences sont trompeuses, cette notation ressemble à la restriction existentielle, sans l'être !) permet de décrire l'ensemble des éléments pour lesquels un rôle R est réflexif : $\exists R.Self$.

Description extensionnelle

Une manière de décrire un concept est en donnant explicitement ses membres : au lieu de Parent(jacques,julie), on peut aussi écrire {jacques} $\sqsubset \exists$Parent.{julie}.

Rôle inverse

On note R^- le **rôle inverse** de R, par exemple Parent$^-$ est équivalent à Enfant puisque $\forall X, Y$ Parent$(X, Y) \leftrightarrow$ Enfant(Y, X).

Rôle universel

Enfin, on note U le **rôle universel**, c'est-à-dire celui qui associe chaque élément à chaque autre élément (y compris lui-même).

Clôture transitive

Enfin, on note R^+ la **clôture transitive** de R, c'est-à-dire le plus petit rôle transitif contenant R.

3.3 Ontologies DL $\mathcal{SROIQ}$

3.3.1 Ontologies DL

Une **ontologie** est la formalisation d'un domaine de connaissances dans un langage de représentation de connaissances.

Une **ontologie DL** est une ontologie représentée dans le langage d'une logique de description.

3.3.2 Expressions et axiomes $\mathcal{SROIQ}$

$\mathcal{SROIQ}$ est une logique de description spécifique que nous allons décrire dans la suite.

Si N_C est un concept quelconque, N_R un rôle quelconque et N_I un individu quelconque, alors on définit de manière itérative dans $\mathcal{SROIQ}$:

1. une **expression de concept C** par $\mathbf{C} ::= N_C \mid (\mathbf{C} \sqcap \mathbf{C}) \mid \mathbf{C} \sqcup \mathbf{C}) \mid \neg\mathbf{C} \mid \top \mid \bot \mid \exists\,\mathbf{R}.\mathbf{C} \mid \forall\,\mathbf{R}.\mathbf{C} \mid \geq n\,\mathbf{R}.\mathbf{C} \mid \leq n\,\mathbf{R}.\mathbf{C} \mid \exists\,\mathbf{R}.Self \mid \{N_I\}$,
2. une **expression de rôle R** par $\mathbf{R} ::= U \mid N_R \mid N_R^-$.

Les axiomes d'une ontologie $\mathcal{SROIQ}$ sont des types suivants :

1. ABox : $\mathbf{C}(N_I)$, $\mathbf{R}(N_I, N_I)$, $N_I \approx N_I$, $N_I \not\approx N_I$,
2. TBox : $\mathbf{C} \sqsubset \mathbf{C}$, $\mathbf{C} \equiv \mathbf{C}$,
3. RBox : $\mathbf{R} \sqsubset \mathbf{R}$, $\mathbf{R} \equiv \mathbf{R}$, $\mathbf{R} \circ \mathbf{R} \sqsubset \mathbf{R}$, $Disjoint(\mathbf{R}, \mathbf{R})$.

3.3.3 Propriétés de $\mathcal{SROIQ}$

Voici les propriétés de la logique de description $\mathcal{SROIQ}$:

1. $\mathcal{S}$ (également appelé $\mathcal{ALCR}^+$) : ($\mathcal{AL}$) présence de noms de concepts et de rôles, de $\top$, du constructeur $\sqcap$ (conjonction), du quantificateur universel, ($\mathcal{C}$) de la négation de concept, ($\mathcal{R}^+$) de la clôture transitive ;
2. $\mathcal{R}$: inclusion de rôles, réflexivité, irréflexivité, exclusion de rôles ;
3. $\mathcal{O}$: possibilité de décrire un concept extensionnellement ($\{N_{I,1}, \ldots, N_{I,n}\}$) ;
4. $\mathcal{I}$: possibilité d'avoir des rôles inverses (R^-) ;
5. $\mathcal{Q}$: possibilité d'avoir des restrictions pleinement quantifiées ($\leq n$, $\geq n$).

D'autres logiques de description sont éventuellement dotées des propriétés suivantes :

6. $\mathcal{H}$: hiérarchie des rôles $R_1 \sqsubset R_2$, $\mathcal{H}$ est plus faible que $\mathcal{R}$;
7. $\mathcal{N}$: restrictions de cardinalité plus faibles que $\mathcal{Q}$.

La norme **OWL** 2 du consortium **WWW** correspond à la logique $\mathcal{SROIQ}$, alors que la norme plus faible OWL-DL correspond à $\mathcal{SHOIN}$. Le logiciel **Protégé** d'édition d'ontologies est basé sur cette dernière.

Chapitre 4

Sémantique et vérification des langages de programmation

4.1 Qu'est-ce qu'un énoncé de langage de programmation ?

Dans la première partie du module, on vérifie la correction syntaxique d'un programme à l'aide de la grammaire formelle du langage de programmation dans lequel il est écrit. Nous allons maintenant nous intéresser à la correction sémantique d'un programme. Mais comment décrire la sémantique d'un programme ?

Un programme écrit dans un langage de programmation peut être considéré comme une **fonction** qui transforme l'état de mémoire en un autre état. Exemple : si les **variable**s $(\mathtt{x}, \mathtt{y})$ ont les valeurs $(8, 7)$ à l'état (initial) s, après l'exécution de $\mathtt{x} := 2 * \mathtt{y} + 1$, on aura un état (final) s' avec des valeurs $(15, 7)$.

Si un programme utilise n variables $(\mathtt{x}_1, \ldots, \mathtt{x}_n)$, un **état** s est un n-uplet de valeurs $(x_1, \ldots, x_n)$.

4.2 Le prédicat caractéristique

Soit U l'ensemble de tous les n-uplets dans un domaine donné, et $U' \subseteq U$. Le **prédicat caractéristique** $P_{U'}$ de U', est défini par

$$U' = \{(x_1, \ldots, x_n) \in U \mid P_{U'}(x_1, \ldots, x_n)\}.$$

Exemple : le code $\mathtt{x} := 2 * \mathtt{y} + 1$ transforme les états de $\{(x, y) \mid \text{vrai}\}$ en des états de $\{(x, y) \mid x = 2y + 1\}$. Et ceux de $\{(x, y) \mid (y \leq 3)\}$ en des états de $\{(x, y) \mid (x \leq 7) \wedge (y \leq 3)\}$. On dira que $\mathtt{x} := 2 * \mathtt{y} + 1$ transforme $(y \leq 3)$ en $(x \leq 7) \wedge (y \leq 3)$.

La **sémantique d'un langage de programmation** est la manière dont chaque énoncé transforme un état initial en un état final.

Une **assertion** est un triplet $\{p\}$ $\mathtt{S}$ $\{q\}$ où $\mathtt{S}$ est un programme, et p et q des formules de la logique du 1$^{\text{er}}$ ordre appelées **précondition** et **postcondition**.

4.3 Assertions, pré- et postconditions

On dit qu'un programme $\mathtt{S}$ *est* ***partiellement correct*** par rapport à p et q, si on a la condition suivante : *quand* $\mathtt{S}$ *se termine après avoir commencé dans un état satisfaisant* p*, alors on se trouve dans un état satisfaisant* q*.*

De même, on dit qu'un programme $\mathtt{S}$ *est* ***totalement correct*** par rapport à p et q, si on a la condition suivante : *quand* $\mathtt{S}$ *commence dans un état satisfaisant* p*, alors il se termine forcément après un nombre fini d'étapes, et on se trouve dans un état satisfaisant* q*.*

Exemples : $\{y \leq 3\}$ $\mathtt{x} := 2 * \mathtt{y} + 1$ $\{(x \leq 7) \wedge (y \leq 3)\}$ est t.c.,

{faux} `S` $\{q\}$ pour tout `S` et q est p.c.,
$\{p\}$ `S` {vrai} pour tout `S` et q est p.c.

On dit qu'une formule B est *plus faible* qu'une formule A si $A \to B$.

Exemples : $y = 1 \vee y = 3$ est plus faible que $y = 1$, $y \leq 3$ est plus faible que $y = 1 \vee y = 3$, $y \leq 4$ est plus faible que $y \leq 3$, etc.

4.4 Précondition la plus faible

Définition 2. *Soit* `S` *un programme et* q *une formule, on note* $\mathrm{wp}(\mathtt{S}, q)$ *la* ***précondition la plus faible*** *telle que* $\{\mathrm{wp}(\mathtt{S}, q)\}$ `S` $\{q\}$.

Ainsi, pour tout p tel que $\{p\}$ `S` $\{q\}$, on aura $p \to \mathrm{wp}(\mathtt{S}, q)$.

Intuitivement on peut dire que l'on cherche la précondition la plus faible parce que son prédicat aura le plus grand nombre de cas pour lesquels notre programme sera correct (= donnera le résultat escompté).

4.5 Sémantique d'un langage de programmation

On peut considérer wp comme un **transformateur de prédicat**.

D'où une autre manière de formaliser un programme : comme *une transformation d'un prédicat de postcondition en un prédicat de précondition.* (Approche axée sur les objectifs.)

Pour définir la sémantique d'un langage donné, il suffit de la définir pour les instructions de base, et pour la manière de combiner les instructions (= la composition).

Nous allons définir la sémantique de :

1. l'affectation
2. la composition d'instructions
3. des tests `if`
4. des boucles `while`.

4.5.1 L'affectation

On définit l'instruction d'**affectation** `x := t` de la manière suivante :

$$\mathrm{wp}(\mathtt{x} := \mathtt{t}, p) = \textsc{Subst}(\{x/t\}, p)$$

pour tout p.

Explication : on veut que, en conclusion, le prédicat p soit vrai. Après l'opération `x := t`, la seule valeur que x peut prendre est t. Donc on voit ce que ça donne quand on substitue x par t dans p. Voici quelques exemples :

— $\{3 < 4 \wedge y \geq 5\}$ `x := 3` $\{x < 4 \wedge y \geq 5\}$ (on veut qu'à la fin $x < 4$; en faisant `x:=3`, ceci est toujours vrai (ce qui est le sens du prédicat $3 < 4$), donc il ne reste que la condition sur y) ;

— $\{3 < 2 \wedge y \geq 5\}$ `x := 3` $\{x < 2 \wedge y \geq 5\}$ (on veut qu'à la fin $x < 2$; en faisant `x:=3`, ceci est toujours faux (ce qui est le sens du prédicat $3 < 2$), donc le programme ne marchera jamais) ;

— $\{y = -1\}$ `x := 3` $\{2x + y = 5\}$ (on veut qu'à la fin $2x + y = 5$; en faisant `x:=3`, ceci n'est vrai que quand $2 \cdot 3 + y = 5$, c'est-à-dire quand $y = -1$.

4.5.2 La composition

Gérer la **composition** de deux instructions :

$$\mathrm{wp}(\mathtt{S}_1; \mathtt{S}_2, q) = \mathrm{wp}(\mathtt{S}_1, \mathrm{wp}(\mathtt{S}_2, q))$$

pour tout q.

Explication : pour arriver à q on passe par $\mathtt{S}_1$ et ensuite par $\mathtt{S}_2$. L'état intermédiaire entre ces deux opérations est la précondition la plus faible pour arriver à q à travers $\mathtt{S}_2$, donc $\mathrm{wp}(\mathtt{S}_2, q)$. On l'utilise en tant que postcondition de $\mathtt{S}_1$, ce qui nous donne la formule.

Exemple :

$$\begin{aligned}\mathrm{wp}(&\mathtt{x} := \mathtt{x} + 1; \mathtt{y} := \mathtt{y} + 2, x < y)\\ &= \mathrm{wp}(\mathtt{x} := \mathtt{x} + 1, \mathrm{wp}(\mathtt{y} := \mathtt{y} + 2, x < y))\\ &= \mathrm{wp}(\mathtt{x} := \mathtt{x} + 1, x < y + 2)\\ &= (x + 1 < y + 2) = (x < y + 1).\end{aligned}$$

4.5.3 Les tests

On définit l'opération de **test**

$$\mathtt{if\ B\ then\ S}_1\ \mathtt{else\ S}_2$$

de la manière suivante :

$$\mathrm{wp}(\mathtt{if\ B\ then\ S}_1\ \mathtt{else\ S}_2, q) = (\mathtt{B} \wedge \mathrm{wp}(\mathtt{S}_1, q)) \vee (\neg\mathtt{B} \wedge \mathrm{wp}(\mathtt{S}_2, q))$$

pour tout q.

Explication : pour arriver à la postcondition q il faut avoir soit B et la précondition de $\mathtt{S}_1$, soit $\neg B$ et la précondition de $\mathtt{S}_2$.

Exemple : en PC.

4.5.4 Les boucles

On définit l'opération de **boucle** `while B do S` de la manière suivante :

$$\mathrm{wp}(\mathtt{while\ B\ do\ S}, q) = ((\neg B \wedge q) \vee (B \wedge \mathrm{wp}(\mathtt{S}; \mathtt{while\ B\ do\ S}, q))$$

pour tout q.

Explication : pour arriver à la postcondition q il faut avoir soit $\neg B$ et q (le $\neg B$ fait que l'on s'arrête, et le q fait que l'on a q puisque rien n'a changé), soit B et la précondition de la composition de `S` et du test.

La définition est récursive : on va exécuter `S` aussi longtemps que nécessaire pour que B devienne faux. Pour éviter d'avoir une énorme disjonction de termes $\mathrm{wp}(\mathtt{S}; \mathtt{S}; ...; \mathtt{S}; \mathtt{while\ B\ do\ S}, q)$ on s'intéressera au cas particulier du prédicat p_{inv} tel que $\{p_{\text{inv}}\}\ \mathtt{S}\ \{p_{\text{inv}}\}$. On appelle p_{inv} un **invariant de boucle**.

Exemple : en PC.

4.6 Comportement de wp

La fonction wp a de très bonnes propriétés, qui nous permettent de faire des calculs : on peut distribuer une conjonction de prédicats, et une implication entre prédicats entraîne l'implication des wp correspondants.

Théorème 4.1 (Distribution). $\mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, q) \leftrightarrow \mathrm{wp}(\mathtt{S}, p \wedge q)$.

Démonstration. Soit s un état tel que $\mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, q)$ soit vrai. Alors les deux sont vrais. Donc après exécution de S, on obtient un état s' dans lequel aussi bien p que q sont vrais, et donc aussi $p \wedge q$. On a donc montré que

$$\{s \mid \ \mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, q)\} \subseteq \{s \mid \ \mathrm{wp}(\mathtt{S} p \wedge q)\},$$

c'est-à-dire le sens $\rightarrow$ du théorème.

Soit s un état tel que $\mathrm{wp}(\mathtt{S}, p \wedge q)$ soit vrai. Après exécution de S, on obtient s' dans lequel $p \wedge q$ est vrai, et donc aussi bien p que q. On a donc montré que

$$\{s \mid \ \mathrm{wp}(\mathtt{S}, p \wedge q)\} \subseteq \{s \mid \ \mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, q)\},$$

c'est-à-dire le sens $\leftarrow$ du théorème. □

Corollaire 4.2 (Il n'y a pas de miracle). $\mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, \neg p) \leftrightarrow \mathrm{wp}(\mathtt{S}, \mathit{faux})$.

Corollaire 4.3 (Dualité). $\neg \mathrm{wp}(\mathtt{S}, \neg p) \rightarrow \mathrm{wp}(\mathtt{S}, p)$.

Théorème 4.4 (Monotonicité). *Si* $p \rightarrow q$, *alors* $\mathrm{wp}(\mathtt{S}, p) \rightarrow \mathrm{wp}(\mathtt{S}, q)$.

Démonstration. D'après le théorème de la distribution ci-dessus,

$$\mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, \neg q) \rightarrow \mathrm{wp}(\mathtt{S}, p \wedge \neg q)$$

D'autre part, l'hypothèse est que $p \rightarrow q$, c'est-à-dire $\neg(p \wedge \neg q)$ et donc $p \wedge \neg q$ est toujours faux :

$$\mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, \neg q) \rightarrow \mathrm{wp}(\mathtt{S}, \mathsf{faux})$$

Le corollaire « il n'y a pas de miracle » nous donne $\mathrm{wp}(\mathtt{S}, \mathsf{faux}) \rightarrow \mathrm{wp}(\mathtt{S}, q)$. En combinant les deux dernières implications :

$$\mathrm{wp}(\mathtt{S}, p) \wedge \mathrm{wp}(\mathtt{S}, \neg q) \rightarrow \mathrm{wp}(\mathtt{S}, q)$$

En appliquant à cette dernière l'égalité $A \wedge B \rightarrow C \equiv C \vee \neg(A \wedge B) \equiv \neg A \vee \neg B \vee C \equiv A \rightarrow \neg B \vee C$, on obtient

$$\mathrm{wp}(\mathtt{S}, p) \rightarrow \neg \mathrm{wp}(\mathtt{S}, \neg q) \vee \mathrm{wp}(\mathtt{S}, q)$$

et donc en appliquant le corollaire de dualité :

$$\mathrm{wp}(\mathtt{S}, p) \rightarrow \mathrm{wp}(\mathtt{S}, q) \vee \mathrm{wp}(\mathtt{S}, q) \equiv \mathrm{wp}(\mathtt{S}, q).$$

□

4.7 Le système déductif de Hoare

Comme dans le cas des logiques propositionnelle et de premier ordre, en établissant un système déductif on peut raisonner...

1. **Axiomes de domaine** (toute formule vraie dans le(s) domaine(s) des variables).
2. **Axiome d'affectation**

$$\{\textsc{Subst}(\{x/t\}, p(x))\}\ \mathtt{x} := \mathtt{t}\ \{p(x)\}.$$

3. **Règle de composition**

$$\frac{\{p\}\ \mathtt{S}_1\ \{q\} \qquad \{q\}\ \mathtt{S}_2\ \{r\}}{\{p\}\ \mathtt{S}_1;\mathtt{S}_2\ \{r\}}.$$

4. **Règle de conséquence**

$$\frac{p_1 \to p \qquad \{p\}\ \mathtt{S}\ \{q\} \qquad q \to q_1}{\{p_1\}\ \mathtt{S}\ \{q_1\}}.$$

5. **Règle de test**

$$\frac{\{p \wedge B\}\ \mathtt{S}_1\ \{q\} \qquad \{p \wedge \neg B\}\ \mathtt{S}_2\ \{q\}}{\{p\}\ \mathtt{if\ B\ then\ S}_1\ \mathtt{else\ S}_2\ \{q\}}.$$

6. **Règle de boucle**

$$\frac{\{p \wedge B\}\ \mathtt{S}\ \{p\}}{\{p\}\ \mathtt{while\ B\ do\ S}\ \{p \wedge \neg B\}}.$$

Pour montrer que $\{p\}$ `while B do S` $\{q\}$, pour un certain q, on va chercher un **invariant de boucle** p_{inv} de S (c'est-à-dire tel que $\{p_{\text{inv}}\}$ S $\{p_{\text{inv}}\}$), on va montrer que $p \to p_{\text{inv}}$, et que $(p_{\text{inv}} \wedge \neg B) \to q$ est vraie : alors, on conclut par la règle de conséquence, que $\{p\}$ `while B do S` $\{q\}$.

4.8 Exemple de vérification de (petit) programme

$$P = \begin{cases} \{\text{vrai}\} \\ \mathtt{x} := 0; \\ \{x = 0\} \\ \mathtt{y} := \mathtt{b}; \\ \{x = 0 \wedge y = b\} \\ \mathtt{while\ y\ <>\ 0\ do} \\ \quad p_{\text{inv}} = \{x = (b - y) \cdot a\} \\ \quad \mathtt{begin\ x} := \mathtt{x} + \mathtt{a}; \mathtt{y} := \mathtt{y} - 1; \mathtt{\ end} \\ \{x = b \cdot a\} \end{cases}$$

Que fait ce programme ? Le vérifier :

Théorème 4.5. $\{\text{vrai}\}$ P $\{x = b \cdot a\}$.

Démonstration. On commence par la fin : ici, on nous donne l'invariant de boucle p_{inv}, il faut

1. vérifier que c'est bien un invariant de boucle : $\{x+a = (b-y)a+a\}$ x := x + a $\{x = (b-y)a + a\}$ y := y − 1 $\{x = (b-y)a\}$;
2. vérifier qu'il est plus faible que la précondition de la boucle : on a bien $\{(x = 0) \wedge (y = b)\} \to \{x = (b-y)a\}$ puisque pour $x = 0$ et $y = b$, p_{inv} s'écrit $0 = 0$ et est donc vrai ;
3. vérifier que $p_{\text{inv}} \wedge B$, B étant le test de la boucle, est plus fort que la postcondition du programme : la négation de B est $\neg(y \neq 0)$, c'est-à-dire $y = 0$, en l'appliquant à p_{inv} on trouve $\{x = b \cdot a\}$ qui est la postcondition.

Après ces trois vérifications, on sait que $\{x = 0 \wedge y = b\}$ est précondition de la boucle pour la postcondition voulue. On applique alors l'axiome d'affectation deux fois, et on trouve que la précondition globale du programme est $\{0 = 0\}$, c'est-à-dire qu'il est vrai sans condition sur x et y. □

4.9 Exercices

Exercice 18 Exercice 1 (Invariant)

Montrer que $\{x = (b - y) \cdot a\}$ `x:=x+1; y:=y-1` $\{x = (b - y) \cdot a\}$.

Exercice 19 Exercice 1 (Test)

Calculer wp(`if y` $= 0$ `then x` $:=$ `0 else x` $:=$ `y` $+ 1, x = y$).

Exercice 20 Exercice 3 (Boucle)

Calculer wp(`while x` > 0 `do x` $:=$ `x` $- 1, x = 0$). Que se passe-t-il si on remplace $x = 0$ par $x = 1$?

Exercice 21 Exercice 4 (Vérification de programme)

Vérifier la correction partielle du programme suivant : $\{a \geq 0\}$

```
x:=0; y:=1;
while y<=a do
begin
x:=x+1;
y:=y+2*x+1
end
```
$\{0 \leq x^2 \leq a < (x + 1)^2\}$

Exercice 22 Exercice 5 (Vérification de programme)

Vérifier la correction partielle du programme suivant : $\{a > 0 \wedge b > 0\}$

```
x:=a; y:=b;
while x<>y do
if x>y
then x:=x-y
else y:=y-x
```
$\{x = \mathrm{PGCD}(a, b)\}$

Exercice 23 Exercice 6 (Vérification de programme)

Vérifier la correction partielle du programme suivant : $\{a \geq 0 \wedge b \geq 0\}$

```
x:=a; y:=b; z:=1
while y<>0 do
if impair(y)
then begin y:=y-1; z:=x*z end
else begin x:=x*x; y:=y div 2 end
```
$\{z = a^b\}$

Index des notations

Index