Université de Marne-la-Vallée
Licence informatique et IUP GMI

Cours de Programmation Fonctionnelle

Marie-Pierre Béal
Loïc Colson

Transparents du cours, également disponibles sur : `http://www-igm.univ-mlv.fr/~beal`

Septembre 2002

BIBLIOGRAPHIE

Logique

- René Cori, Daniel Lascar, *Logique Mathématique*, Tome 1 : *Calcul Propositionnel*, Masson, 1993.
- Stephen C. Kleene, *Logique Mathématique*, Editions Jacques Gabay, 1987
- Thomas Ehrhard, Cours de Logique, Licence d'informatique, Polycopié, Université de Marne-la-Vallée, 1993.

Objective Caml

- Émmanuel Chailloux, Pascal Manoury et Bruno Pagano,
  Développement d'applications avec Objective Caml, Éditions O'Reilly, 2000.
- Pierre Weis et Xavier Leroy, *Le langage Caml*, InterEditions, Paris 1993, 2e tirage
- Xavier Leroy et Pierre Weis, *Manuel de Référence du langage Caml*, InterEditions, Paris 1993.

- Guy Cousineau et Michel Mauny, *Approche Fonctionnelle de la Programmation*, Ediscience (Collection Informatique), Paris 1995.
- Thomas Ehrhard, Note de cours, Deug, Polycopié, Université de Marne-la-Vallée, 1993.

LOGIQUE

Introduction à la logique

"Qu'est-ce qui se déduit de quoi ?" (Kleene)

$$
\begin{array}{ll}
A \vee B \longrightarrow B \vee A & : \texttt{MORPHOLOGIE} \\
\models A \vee B \longrightarrow B \vee A & : \texttt{SEMANTIQUE} \\
\vdash A \vee B \longrightarrow B \vee A & : \texttt{SYNTAXE}
\end{array}
$$

CALCUL DES PROPOSITIONS

Théorie des ensembles

$$\{a \mid P(a)\}$$

Exemple $\emptyset = \{a \mid a \neq a\}$.

- $E \subset F \equiv \forall a \;\; a \in E \Longrightarrow a \in F$
- $E \cap F = \{a \mid (a \in E) \wedge (a \in F)\}$
- $E \cup F = \{a \mid (a \in E) \vee (a \in F)\}$

Soit $(E_i)_{i \in I}$ une famille d'ensembles,

- $\bigcap_{i \in I} E_i = \{a \mid \forall i \;\; a \in E_i\}$
- $\bigcup_{i \in I} E_i = \{a \mid \exists i \;\; a \in E_i\}$

Produit cartésien

- $E \times F = \{(a, b) \mid a \in E \wedge b \in F\}$
  $(a, b) = (a', b') \longleftrightarrow [(a = a') \wedge (b = b')]$

- $A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i\}$

- $E^n = \overbrace{E \times E \times \cdots \times E}^{n \texttt{ fois}}$

- une relation sur $E \times F$ est un sous-ensemble $R \subset E \times F$.

- R est *fonctionnelle* ssi :
  $\forall a \in E \ \ \exists! b \in F \mid aRb \ \ ((a, b) \in R)$
  On notera $R(a)$ l'unique b tel que aRb.

- Fonction = Relation fonctionnelle.
  Notation: $R : E \longrightarrow F$

Propiétés des fonctions

Soit $f : E \longrightarrow F$

- f *surjective* : $\forall b \in F\ \exists a \in E \mid f(a) = b$
- f *injective* : $\forall a, a' \in E\ [f(a) = f(a')] \Longrightarrow a = a'$
- f *bijective* ssi f est injective et surjective.

Propriétés des relations

Soit $R \in E \times E$

- R *réflexive* ssi $\forall a \in E \;\; aRa$
- R *symétrique* ssi $\forall a, b \in E$

$$(aRb \Longrightarrow bRa)$$

- R *antisymétrique* ssi $\forall a, b \in E$

$$((aRb) \wedge (bRa)) \Longrightarrow a = b$$

- R *transitive* ssi $\forall a, b, c \in E$

$$((aRb) \wedge (bRc)) \Longrightarrow aRc$$

- R *relation d'ordre* ssi R est $\begin{cases} \texttt{réflexive} \\ \texttt{antisymétrique} \\ \texttt{transitive} \end{cases}$
- R *relation d'équivalence* ssi R est $\begin{cases} \texttt{réflexive} \\ \texttt{symétrique} \\ \texttt{transitive} \end{cases}$

Les entiers naturels

Les entiers

$$\begin{array}{lcccccc}
\texttt{BÂTONS} & & | & || & ||| & |||| & \cdots \\
\texttt{DECIMALE} & 0 & 1 & 2 & 3 & 4 & \cdots \\
\texttt{UNAIRE} & 0 & S & S & S & S & \cdots \\
 & & | & | & | & | & \\
 & & 0 & S & S & S & \\
 & & & | & | & | & \\
 & & & 0 & S & S & \\
 & & & & | & | & \\
 & & & & 0 & S & \\
 & & & & & | & \\
 & & & & & 0 &
\end{array}$$

$\mathbb{N} = \{0, 1, 2, \ldots\}$ = ensemble des entiers.

Récurrence faible
Soit $P(n)$ est une propriété des entiers, si

- $P(0)$ est vérifiée
- $\forall n\ P(n) \Longrightarrow P(n+1)$

Alors $P(n)$ est vérifiée pour tout n.

Récurrence forte
Soit $P(n)$ est une propriété des entiers,

- Si $\forall n \in \mathbb{N}$
  $(\forall m \in \mathbb{N}\ \ m < n \Longrightarrow P(m)) \Longrightarrow P(n)$

Alors $P(n)$ est vérifiée pour tout n.

Proposition 1 *Les deux principes de récurrence sont équivalents.*

Définitions par récurrence

Si on se donne :

- un objet $O_0 \in A$
- une loi permettant de défini O_{n+1} à partir de O_n

Alors O_n est défini pour tout n.

Ensembles dénombrables

Définition On dit qu'un ensemble A est *dénombrable* s'il existe une fonction $f : \mathbb{N} \longrightarrow A$ bijective.

Résultats classiques

- Si E est dénombrable et $F \subset E$, alors F est fini ou dénombrable
- Si E et F sont dénombrables, $E \cup F$ aussi.
- $\mathbb{N} \times \mathbb{N}$ est dénombrable.
  $f : \mathbb{N} \longrightarrow \mathbb{N} \times \mathbb{N}$

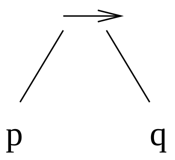
$$\begin{aligned} f(0) &= (0,0) \\ f(n+1) &= \begin{cases} (k+1,0) \; \texttt{si} \; f(n) = (0,k) \\ (k,l+1) \; \texttt{si} \; f(n) = (k+1,l) \end{cases} \end{aligned}$$

- Si E et F sont dénombrables, $E \times F$ aussi.
- Si $(E_i)_{i \in I}$ est une famille d'ensembles dénombables indexée par I dénombrable, alors $\bigcup_{i \in I} E_i$ est dénombrable.

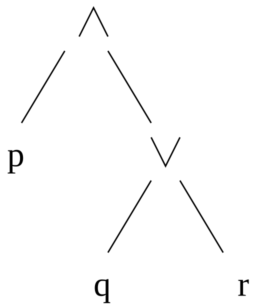
Morphologie

On veut abstraire la bonne définition à partir d'exemples connus :

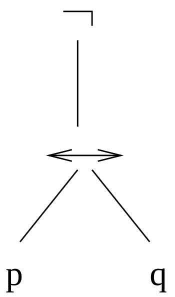
$p \to q$



$p \wedge (q \vee r)$



$\neg(p \leftrightarrow q)$



Les atomes : On se donne une fois pour toutes un ensemble dénombrable $\mathcal{P} = \{p, q, r \ldots\}$ d'*atomes propositionnels* (ou *formules atomiques*).

Les connecteurs : On se donne un ensemble fini $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ de *connecteurs*.
Chaque connecteur c_i est muni d'une *arité* entière notée $ar(c_i)$. L'arité correspond au nombre de formules qu'il regroupe dans la constitution d'une formule plus complexe.

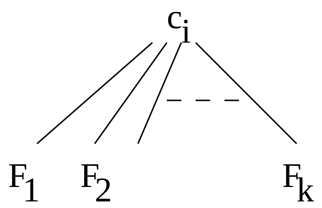
Exemple $\mathcal{C} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

$$\begin{aligned} ar(\neg) &= 1 \\ ar(\wedge) &= 2 \\ ar(\vee) &= 2 \\ ar(\rightarrow) &= 2 \\ ar(\leftrightarrow) &= 2 \end{aligned}$$

Les formules

Une *formule* F (bâtie sur $\mathcal{C}, \mathcal{P}$) est :

- soit un atome $p \in \mathcal{P}$
- soit



  où $k = ar(c_i)$ et où $F_1, \ldots, F_k$ sont des formules.

Une formule est donc un arbre.

Exemples voir les exemples ci-dessus.
Notation on notera F par $c_i F_1 \ldots F_k$ (notation préfixe) ou $F_1 c_i F_2$ si c_i est d'arité 2 (notation infixe).
On notera $\mathcal{F}(\mathcal{C}, \mathcal{P})$ l'ensemble des formules bâties sur $\mathcal{C}, \mathcal{P}$.
On notera $\mathcal{C}_n$ l'ensemble des connecteurs d'arité n.

Hauteur
La *hauteur* d'une formule F est définie comme suit :

- $h(F) = 0$ si $F \in \mathcal{P}$
- $h(F) = 1 + \mathtt{max}(h(F_1), h(F_2), \ldots, h(F_k))$ si $F = c_i F_1 \ldots F_k$

Exemple $h((p \vee q) \rightarrow (q \vee p)) = 2$.

Récurrence sur les formules
Soit P un prédicat sur les formules (une propriété des formules),

- Si $P(p)$ pour tout $p \in \mathcal{P}$
- Si $P(F_1), \ldots, P(F_k)$ entraîne $P(cF_1 \ldots F_k)$ pour tout $c \in \mathcal{C}$ d'arité k

Alors $P(F)$ pour toute $F \in \mathcal{F}(\mathcal{C}, \mathcal{P})$.
Preuve par récurrence sur la hauteur de F.
Difficulté : la hauteur d'une formule est elle-même définie par récurrence sur cette formule ...

Applications

Principe de définition récursive sur les formules

On définit une famille d'objets $(O(F))_{F\in\mathcal{F}(\mathcal{C},\mathcal{P})}$ si :

- on se donne $O(F)$ pour $F \in \mathcal{P}$
- $\forall n \geq 1$, $\forall c \in \mathcal{C}_n$, $\forall F_1 \dots F_n \in \mathcal{F}(\mathcal{C},\mathcal{P})$ pour lesquelles $O(F_1)\dots O(F_n)$ sont définis de façon unique, on se donne un unique $O(cF_1 \dots F_n)$ en fonction de $F_1, \dots, F_n$ et de $O(F_1), \dots, O(F_n)$.

Atomes d'une formule

On définit par récurrence sur F l'ensemble $At(F) \subset \mathcal{P}$ de ses *atomes* par :

- $At(p) = \{p\}$ si $p \in \mathcal{P}$
- $At(cF_1 \dots F_k) = \bigcup_{1\leq i\leq k} At(F_i)$

Remarque sur les formules

Un connecteur comme $\top$ ou $\bot$ peut être d'arité nulle. Les connecteurs d'arité nulle sont appelés *constantes propositionnelles*. Leur hauteur est 0.

Sous-formules

Soit F une formule. On définit $s(F) \subset \mathcal{F}(\mathcal{C}, \mathcal{P})$ par récurrence sur F.

- $s(p) = \{p\}$ pour $p \in \mathcal{P}$
- $s(cF_1 \ldots F_k) = \{cF_1 \ldots F_k\} \cup \bigcup_{1 \leq i \leq k} s(F_i)$

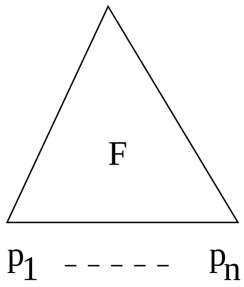
Exemple

$$s((A \wedge B) \to (C \vee D)) =$$

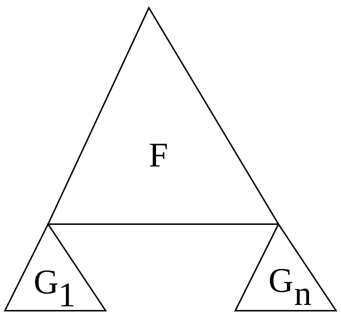
$$\{(A \wedge B) \to (C \vee D), A \wedge B, C \vee D, A, B, C, D\}$$

Substitutions

Soit F une formule :



On note $F[p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n]$ la formule :



Définition par récurrence

$H = F\ [p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n]$ est défini par récurrence sur F :

- Si $F = p_i$ alors $H = G_i$
- Si $F = q \in \mathcal{P}$, $q \neq p_i \forall i$, alors $H = q$
- Si $F = cF_1 \ldots F_k$ alors $H = cH_1 \ldots H_k$ où $H_i = F_i\ [p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n]$

Exemple

$$(p \to q)[p \leftarrow A \vee B, q \leftarrow C \vee D]$$
$$= (A \vee B) \to (C \vee D)$$

Sémantique

La sémantique est la théorie de la *signification* des formules.

Définition Soit $\mathbf{B} = \{0, 1\}$ l'ensemble des valeurs de vérités. $(0 = \mathtt{F}, 1 = \mathtt{V})$.

Définition Une *interprétation* est une fonction $I : \mathcal{P} \longrightarrow \mathbf{B}$.

On suppose donnée pour chaque connecteur $\gamma \in \mathcal{C}$ une fonction $\tilde{\gamma} : \mathbf{B}^n \longrightarrow \mathbf{B}$, (avec $n = ar(\gamma)$). (fonction booléenne).

Exemple Connecteurs usuels

p	q	$\neg p$	$p \rightarrow q$	$p \wedge q$	$p \vee q$	$p \leftrightarrow q$
V	V	F	V	V	V	V
V	F	F	F	F	V	F
F	V	V	V	F	V	F
F	F	V	V	F	F	V

Interprétation des formules

On étend $I : \mathcal{P} \longrightarrow \mathbf{B}$ en $\overline{I} : \mathcal{F}(\mathcal{C}, \mathcal{P}) \longrightarrow \mathbf{B}$ par récurrence sur les formules :

- $\overline{I}(p) = I(p)$ si $p \in \mathcal{P}$
- $\overline{I}(cF_1 \dots F_n) = \tilde{c}(\overline{I}(F_1), \dots, \overline{I}(F_n))$

Exemple

p	q	$p \vee q$	$p \wedge q$	$(p \vee q) \wedge (p \wedge q)$
V	F	V	F	F

Si $I(p) =$V et $I(q) =$F alors

$$\overline{I}((p \vee q) \wedge (p \wedge q)) = \mathtt{F}.$$

Proposition 2 *Soient I et I' deux interprétations telles que $\forall p \in At(F)\ \ I(p) = I'(p)$. Alors $\overline{I}(F) = \overline{I}'(F)$.*

Définitions

- $I \models F$ si $\overline{I}(F) = 1$
- F et F' sont *sémantiquement équivalentes* si $\forall I\ \overline{I}(F) = \overline{I}(F')$
- Si $T \subset \mathcal{F}(\mathcal{C}, \mathcal{P})$, $I \models T$ si $\forall F \in T\ I \models F$
- On dit que F est *conséquence sémantique* de T et on écrit $T \models F$ si

$$\forall I\ I \models T \Longrightarrow I \models F$$

- $\models F$ si $\emptyset \models F$. Ou encore $\forall I\ I \models F$. On dit que F est une *tautologie*.
- T est *insatisfiable* s'il n'existe pas d'interprétation I telle que $I \models T$.

Proposition 3 *Soit $\varphi : \mathcal{F}(\mathcal{C}, \mathcal{P}) \longrightarrow \mathbf{B}$ une fonction. Il existe $I : \mathcal{P} \longrightarrow \mathbf{B}$ telle que $\varphi = \overline{I}$ ssi : $\forall c \in \mathcal{C}\ ar(c) = n$ et $\forall F_1, \ldots F_n$,*

$$\varphi(cF_1 \ldots F_n) = \tilde{c}(\varphi(F_1), \ldots, \varphi(F_n))$$

L'interprétation I telle que $\varphi = \overline{I}$ est alors la restriction de φ à $\mathcal{P}$.

Complétude fonctionnelle

Un ensemble de connecteurs $\mathcal{C}$ est *fonctionnellement complet* ssi $\forall n \in \mathbb{N}\ \forall \varphi\ \mathbf{B}^n \longrightarrow \mathbf{B}$, et pour toute famille $p_1 \ldots p_n \in \mathcal{P}$ de propositions atomiques 2 à 2 distinctes, $\exists F \in \mathcal{F}(\mathcal{C}, \mathcal{P})$ telle que $\forall (x_1, \ldots, x_n) \in \mathbf{B}^n$ et $\forall I \mid I(p_i) = x_i$, $\overline{I}(F) = \varphi(x_1, \ldots, x_n)$. ($\varphi$ est la fonction booléenne calculée par F).

Proposition 4 *Les ensembles de connecteurs suivants sont fonctionnellement complets :*

$$\{\vee, \neg\}$$

$$\{\wedge, \neg\}$$

$$\{\rightarrow, \bot\}$$

$$\{\rightarrow, \neg\}$$

Proposition 5 *Soit $\mathcal{B}$ un ensemble de connecteurs fonctionnellement complet. Soit $\mathcal{C}$ un ensemble de connecteurs. Alors toute formule de $\mathcal{F}(\mathcal{C}, \mathcal{P})$ est sémantiquement équivalente à une formule de $\mathcal{F}(\mathcal{B}, \mathcal{P})$.*

Proposition 6 *Soit $F \in \mathcal{F}(\mathcal{C}, \mathcal{P})$ et I une interprétation. Soient $G_1, \ldots, G_n$ des formules. Alors $\overline{I}(F[p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n]) = \overline{J}(F)$ où J est définie par :*

$$\begin{aligned} J(p) &= I(p) \ \textbf{\textit{où}}\ p \neq p_i \ \forall i \\ J(p_i) &= \overline{I}(G_i) \end{aligned}$$

Corollaire 7 *Soit F une tautologie. Alors $F[p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n]$ est une tautologie.*

Exemple $p \to (q \to p)$ est une tautologie. Donc aussi :

$$(p \wedge (q \vee \neg r)) \to (\neg r \to (p \wedge (q \vee \neg r)))$$

Tautologies remarquables

- Tiers exclu : $p \vee \neg p$
- Pour la disjonction :
  - associativité : $(p \vee q) \vee r \leftrightarrow p \vee (q \vee r)$
  - commutativité : $(p \vee q) \leftrightarrow (q \vee p)$
  - idempotence : $(p \vee p) \leftrightarrow p$
  - élément neutre: $(p \vee \bot) \leftrightarrow p$
  - élement absorbant : $(p \vee \top) \leftrightarrow \top$
- Pour la conjonction :
  - associativité : $(p \wedge q) \wedge r \leftrightarrow p \wedge (q \wedge r)$
  - commutativité : $(p \wedge q) \leftrightarrow (q \wedge p)$
  - idempotence : $(p \wedge p) \leftrightarrow p$
  - élément neutre: $(p \wedge \top) \leftrightarrow p$
  - élement absorbant : $(p \wedge \bot) \leftrightarrow \bot$
- Distributivité :
  $p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$
  $p \wedge (q \vee r) \leftrightarrow (p \wedge q) \vee (p \wedge r)$

- Lois de De Morgan :
  $\neg(p \vee q) \leftrightarrow \neg p \wedge \neg q$
  $\neg(p \wedge q) \leftrightarrow \neg p \vee \neg q$
- Pour la négation :
  $\neg\top \leftrightarrow \bot$
  $\neg\bot \leftrightarrow \top$
  $(\neg\neg p) \leftrightarrow p$
- Définissabilité de $\rightarrow$:
  $(p \rightarrow q) \leftrightarrow (\neg p \vee q)$
- Contraposition :
  $(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$
- Sur l'implication :
  $p \rightarrow (q \rightarrow p)$
  $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$

Théorie de la démonstration (syntaxe)

Question : Qu'est-ce qu'une preuve correcte ?

On se place dans le système de connecteurs :

$$\boxed{\mathcal{C} = \{\rightarrow, \perp\}}$$

L'expression $\neg A$ sera une abréviation de $A \rightarrow \perp$. (On vérifie que l'on a pour toute I, $\bar{I}(\neg A) = \tilde{\neg}(\bar{I}(A))$.)

Définition On appelle *axiome hilbertien* toute formule de la forme :

(K) : $A \rightarrow (B \rightarrow A)$

(S) : $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

(N) : $(\neg\neg A) \rightarrow A$

Définition Soit $T \subset \mathcal{F}(\mathcal{C}, \mathcal{P})$. Une *preuve* à partir de T est une suite $F_1, \ldots, F_n$ de formules telles que pour tout $i \in \{1, \ldots, n\}$ l'une des conditions suivantes est réalisée :

- F_i est un axiome hilbertien
- $F_i \in T$
- $\exists j, k < i$ tels que $F_j = F_k \to F_i$

Dans le dernier cas, on dit que F_i se déduit par *modus ponens* de F_k et F_j.

Définition Une *preuve* de F à partir de T est une preuve $F_1, \ldots, F_n$ à partir de T telle que $F_n = F$.

Notation On écrira $T \vdash F$ s'il existe une démonstration ou preuve de F à partir de T.

Notation On écrira $\vdash F$ si $\emptyset \vdash F$ et on dira que F est *prouvable* ou *démontrable*.

Exemple $\vdash p \to p$

Proposition 8 *Si* $T \vdash F$ *alors* $\forall G_1, \ldots G_n$,

$$\{H[p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n] \mid H \in T\}$$
$$\vdash F[p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n]$$

Exemple Pour toute formule F on a

$$\vdash F \to F.$$

Proposition 9 *Si* $T \vdash F$ *et* $S \vdash (F \to G)$ *alors* $T \cup S \vdash G$

Preuve de $p \to p$

$$
\begin{aligned}
F_1 &= p \to ((p \to p) \to p) \ (K) \\
F_2 &= (p \to ((p \to p) \to p)) \\
&\quad \to ((p \to (p \to p)) \to (p \to p)) \ (S) \\
F_3 &= (p \to (p \to p)) \to (p \to p) \quad \mathtt{MP}(F_1, F_2) \\
F_4 &= p \to (p \to p) \ (K) \\
F_5 &= p \to p \quad \mathtt{MP}(F_4, F_3)
\end{aligned}
$$

Proposition 10 *Soit* $T \subset \mathcal{F}(\mathcal{C}, \mathcal{P})$ *(éventuellement infini) et* $F \in \mathcal{F}(\mathcal{C}, \mathcal{P})$ *telle que* $T \vdash F$. *Alors il existe* $S \subset T$, S fini *tel que* $S \vdash F$.

Proposition 11 *Si* $T \vdash (F \rightarrow G)$ *alors*

$$T \cup \{F\} \vdash G.$$

Réciproquement, on a le **théorème de déduction** :

Proposition 12 *Si* $T \cup \{F\} \vdash G$, *alors*

$$T \vdash (F \rightarrow G).$$

Preuve Récurrence sur la longueur de preuve de G.

Exemple $\vdash A \to \neg\neg A$.
Ne pas oublier que $\neg A$ est une abréviation de $A \to \bot$.

Définition Soit $T \subset \mathcal{F}(\mathcal{C}, \mathcal{P})$. On dit que T est *contradictoire* si $T \vdash \bot$.

Raisonnement par l'absurde :

Proposition 13 *$T \vdash F$ ssi $T \cup \{\neg F\} \vdash \bot$.*

Proposition 14 *$\{\bot\} \vdash F$.*

Preuve :

$$\begin{array}{rcl} & \vdash & \bot \to (\neg F \to \bot) \ \texttt{(K)} \\ \{\bot\} & \vdash & \neg F \to \bot \\ \{\bot\} & \vdash & \neg\neg F \\ & \vdash & \neg\neg F \to F \ \texttt{(N)} \\ \{\bot\} & \vdash & F \end{array}$$

Lemme 1 *Soit I une interprétation. Si $I \models F$ et $I \models F \to G$, alors $I \models G$.*

Proposition 15 *Si $T \vdash F$ alors $T \models F$.*

Preuve Par récurrence sur la longueur de la preuve de F.

Principes dérivés

Manipulation des autres connecteurs.

Définition On pose $F \wedge G \equiv \neg(F \to \neg G)$

Lemme 2 *Si* $T \vdash F$ *et* $T \vdash G$ *alors*

$$T \vdash F \wedge G.$$

*(Règle d'*introduction *de la conjonction).*

Lemme 3 *Si* $T \vdash F \wedge G$ *alors*

$$T \vdash F \ \texttt{et} \ T \vdash G.$$

*(Règle d'*élimination *de la conjonction).*

Définition On pose $F \vee G \equiv \neg F \to G$

Lemme 4 *Si* $T \vdash F$ *alors* $T \vdash F \vee G$.

Lemme 5 *Si* $T \vdash G$ *alors* $T \vdash F \vee G$. *(Règles d'*introduction *de la disjonction).*

Lemme 6 *Si* $T \vdash F \vee G$, $T \cup \{F\} \vdash H$ *et* $T \cup \{G\} \vdash H$ *alors* $T \vdash H$.

Exemples

$$\vdash (A \wedge B) \to (B \wedge A)$$
$$\vdash (A \wedge (B \wedge C)) \to ((A \wedge B) \wedge C)$$
$$\vdash (A \vee B) \to (B \vee A)$$
$$\vdash (A \wedge (B \vee C)) \to (A \wedge B) \vee (A \wedge C)$$
$$\vdash ((A \wedge B) \to C) \leftrightarrow (A \to (B \to C))$$
$$\vdash (A \to (B \wedge C)) \to ((A \to B) \wedge (A \to C))$$

Complétude

But : $T \models F$ entraîne $T \vdash F$.

Définition On dit que $S \subset \mathcal{F}(\mathcal{C}, \mathcal{P})$ est *saturé* si $\forall F \in \mathcal{F}(\mathcal{C}, \mathcal{P})$ on a :

$$(F \in S) \ \texttt{ou} \ (\neg F \in S)$$

Proposition 16 *Si T est non contradictoire, alors T est satisfiable.*

Preuve :
Soit T non contradictoire.
On construit un ensemble $T' \subset \mathcal{F}(\mathcal{C}, \mathcal{P})$ non contradictoire tel que $T \subset T'$ et qui est saturé.
Soit $F_1, F_2, \ldots$ une énumération de $\mathcal{F}(\mathcal{C}, \mathcal{P})$ ($\mathcal{F}(\mathcal{C}, \mathcal{P})$ est dénombrable). On construit une suite $T_1, T_2, \ldots$ de parties de $\mathcal{F}(\mathcal{C}, \mathcal{P})$ croissante pour $\subset$ par récurrence :

- $T_1 = T$
- Si $F_n \in T_n$ ou $\neg F_n \in T_n$, on pose $T_{n+1} = T_n$

- Sinon l'un de deux ensembles $T_n \cup \{F_n\}$, $T_n \cup \{\neg F_n\}$ est non contradictoire. On le prend pour T_{n+1}.

Soit
$$T' = \bigcup_n T_n$$
T' contient T, est saturé et non contradictoire.
Soit $\varphi : \mathcal{F}(\mathcal{C}, \mathcal{P}) \longrightarrow \mathbf{B}$ la fonction définie par :
$$\varphi(F) = \begin{cases} 1 & \texttt{si } F \in T' \\ 0 & \texttt{sinon} \end{cases}$$
Rappel : il existe une interprétation I telle que $\varphi = \overline{I}$ ssi φ vérifie :

- $\varphi(cF_1 \dots F_n) = \tilde{c}(\varphi(F_1), \dots, \varphi(F_n))$.
- $\forall c \in \mathcal{C}_0, \varphi(c) = \tilde{c}$.

On le vérifie et $I \models T'$. Donc T est satisfiable.
Remarque : $F \in T'$ ssi $T' \vdash F$.

Corollaire 17 *Si* $T \models F$ *alors* $T \vdash F$.

Preuve : Si $T \models F$ alors $T \cup \{\neg F\}$ est insatisfiable. Donc $T \cup \{\neg F\}$ est contradictoire. D'où $T \vdash F$ (démonstration par l'absurde).

RÉSOLUTION

Les clauses : morphologie et sémantique

Définition :
Une *clause* est un couple $C = (C^-, C^+)$ de parties finies de $\mathcal{P}$.

La clause vide est $(\emptyset, \emptyset)$, aussi notée $()$ ou $\Box$.
On note $\mathcal{L}$ l'ensembles des clauses.
Si C, C' sont deux clauses,

$$C \cup C' = (C^- \cup C'^-, C^+ \cup C'^+).$$

Une clause $C = (\{p_1, \ldots, p_n\}, \{q_1, \ldots, q_m\})$ est représentée par la formule :

$$\neg p_1 \vee \cdots \vee \neg p_n \vee q_1 \vee \cdots \vee q_m$$

Définition :
Soit I une interprétation. On étend I aux clauses en $\overline{I} : \mathcal{L} \longrightarrow \mathbf{B}$

$$\overline{I}(C) = \begin{cases} 1 \ \texttt{si} \ \exists u \in C^+ \mid I(u) = 1 \\ \quad \texttt{ou} \ \exists v \in C^- \mid I(v) = 0 \\ 0 \ \texttt{sinon} \end{cases}$$

Proposition 18 *On a $\overline{I}(()) = 0$.*
Soient L et M deux ensembles de clauses. On a

- $(I \models L \ \texttt{et} \ I \models M)$ *ssi* $(I \models L \cup M)$
- $(I \models L \ \texttt{ou} \ I \models M)$ *ssi* $(I \models N)$ *où* $N = \{C \cup D \mid C \in L \ \texttt{et} \ D \in M\}$

Le but de la suite est de décrire un algorithme permettant de transformer toute formule d'un système quelconque de connecteurs en un ensemble de clauses qui lui est sémantiquement équivalent.

On part de :

$$\boxed{\mathcal{C} = \{\neg, \wedge, \vee\}}$$

qui est fonctionnellement complet.

Définitions :
$\mathcal{F}'(\mathcal{C},\mathcal{P})$ est la plus petite partie de $\mathcal{F}(\mathcal{C},\mathcal{P})$ telle que

- $\forall p \in \mathcal{P}, p$ `et` $\neg p \in \mathcal{F}'(\mathcal{C},\mathcal{P})$
- $F, G \in \mathcal{F}'(\mathcal{C},\mathcal{P}) \implies F \wedge G, F \vee G \in \mathcal{F}'(\mathcal{C},\mathcal{P})$

Définitions : On “fait descendre les négations” à l'aide de f et g :

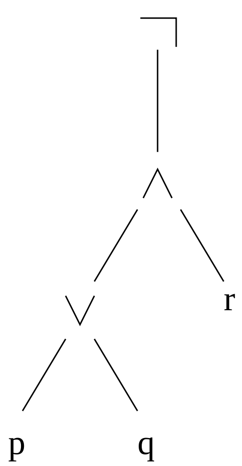
$$f(F) = \begin{cases} p & \texttt{si } F = p \in \mathcal{P} \\ g(G) & \texttt{si } F = \neg G \\ f(G) \vee f(H) & \texttt{si } F = G \vee H \\ f(G) \wedge f(H) & \texttt{si } F = G \wedge H \end{cases}$$

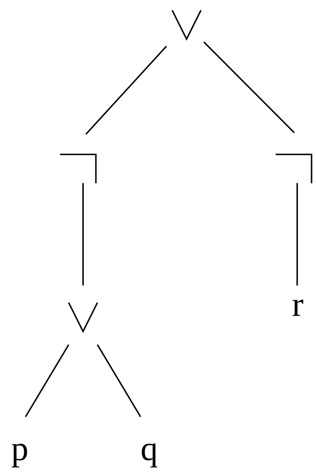
$$g(F) = \begin{cases} \neg p & \texttt{si } F = p \in \mathcal{P} \\ f(G) & \texttt{si } F = \neg G \\ g(G) \vee g(H) & \texttt{si } F = G \wedge H \\ g(G) \wedge g(H) & \texttt{si } F = G \vee H \end{cases}$$

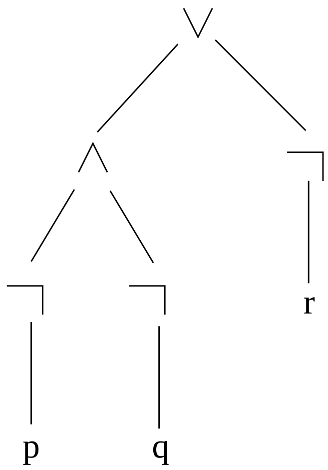
Proposition 19 *On a :*

- $\forall F \in \mathcal{F}(\mathcal{C}, \mathcal{P}), f(F), g(F) \in \mathcal{F}'(\mathcal{C}, \mathcal{P})$
- $\overline{I}(f(F)) = \overline{I}(F)$
- $\overline{I}(g(F)) = \tilde{\neg}(\overline{I}(F))$

Exemple







Pour finir, on associe à toute formule de $\mathcal{F}'(\mathcal{C}, \mathcal{P})$ un ensemble de clauses qui lui est sémantiquement équivalent :

$$i : \mathcal{F}'(\mathcal{C}, \mathcal{P}) \longrightarrow \mathcal{P}_{arties}(\mathcal{L})$$

- Si $F = p \in \mathcal{P}$, $i(p) = \{(\emptyset, \{p\})\}$
- Si $F = \neg G$, alors $G = p \in \mathcal{P}$ et $i(F) = \{(\{p\}, \emptyset)\}$
- Si $F = G \wedge H$, $i(F) = i(G) \cup i(H)$
- Si $F = G \vee H$, $i(F) = \{C \cup D \mid C \in i(G) \texttt{ et } D \in i(H)\}$

Proposition 20 *Soit* $F \in \mathcal{F}'(\mathcal{C}, \mathcal{P})$,
$I \models F$ *ssi* $I \models i(F)$.

Corollaire 21 *Soit* $F \in \mathcal{F}(\mathcal{C}, \mathcal{P})$,
$I \models F$ *ssi* $I \models Cl(F)$, *où* $Cl = i \circ f$.

Résolvante

Définition :
Soient C, C' deux clauses. La clause D *se déduit* de C et C' par *résolution* modulo p si $p \in C^+ \cap C'^-$ et si :

$$D = (C^- \cup (C'^- - \{p\}), (C^+ - \{p\}) \cup C'^+)$$

On dit que D est une *résolvante* de C et C'.

Exemple

$$C = \neg p \vee \neg q \vee r \vee s, \quad C' = \neg w \vee \neg r \vee x$$

$$D = \neg p \vee \neg q \vee \neg w \vee s \vee x$$

$$\boxed{\begin{array}{c} \bar{p}\bar{q}rs, \bar{w}\bar{r}x \\ \downarrow \\ \bar{p}\bar{q}\bar{w}sx \end{array}}$$

Une résolution modulo r.

Définition :
Soit Δ un ensemble de clauses. Une *preuve par résolution* à partir de Δ est une suite de clauses $C_1, \ldots, C_n$ telle que $\forall i \in \{1, 2, \ldots, n\}$ on a :

- $C_i \in \Delta$
- ou $\exists j, k < i$ tels que C_i est une résolvante de C_j et C_k.

On dit que C est *prouvable* par résolution à partir de Δ s'il existe une preuve par résolution $C_1, \ldots, C_n$ avec $C_n = C$.

$$\Delta \vdash C$$

Proposition 22 *Soit D est une résolvante de C et C'. Si $I \models C$ et $I \models C'$, alors $I \models D$.*

Proposition 23 *Soit Δ une ensemble de clauses et C une clause. Si $\Delta \vdash C$ alors $\Delta \models C$.*

Méthode de résolution

Le théorie de la démonstration est ici simple. Il est plus facile d'implémenter un démonstrateur automatique travaillant sur les clauses que sur des formules bâties sur l'ensemble de connecteurs $\{\rightarrow, \bot\}$ et qui aurait à synthétiser des preuves dans le système de Hilbert.

Idée : On va montrer que si Δ est un ensemble de clauses insatisfiable ($\Delta \models \Box$) alors Δ est contradictoire ($\Delta \vdash \Box$). On cherche à réfuter un ensemble de clauses Δ en arrivant à $\Box$.

Arbre de résolution

Soit Δ un ensemble de clauses. Un *arbre de résolution* à partir de Δ est un arbre binaire étiqueté par des clauses tel que

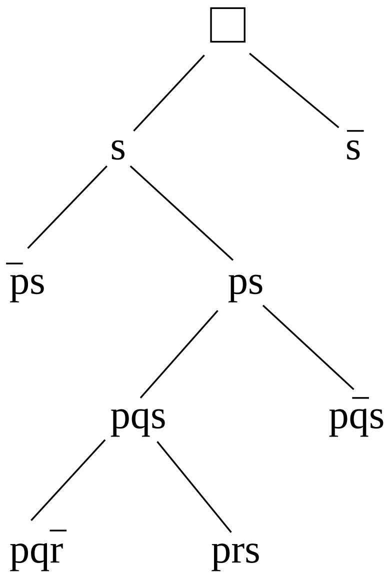
- les feuilles sont étiquetées dans Δ
- l'étiquette d'un noeud interne est une résolvante des étiquettes des fils

Exemple

`Notation :` $pq\bar{r}$ `est la clause` $(\{r\},\{p,q\})$

Voici un arbre qui réfute :

$$\Delta = \{pq\bar{r}, pr\,s, p\bar{q}s, \bar{p}s, \bar{s}\}$$



Complétude de la méthode de résolution

On va montrer que si Δ est insatisfiable, alors il existe un arbre de résolution qui réfute Δ.

Définition
Si $u \in \mathcal{P}$, on note

$$\Delta(u) = \{C - \{\bar{u}\} \mid C \in \Delta \ \texttt{et} \ u \notin C\}$$

Exemple :

$$\Delta = \{pq\bar{r}, prs, p\bar{q}s, \bar{p}s, \bar{s}\}$$

$$\Delta(r) = \{pq, p\bar{q}s, \bar{p}s, \bar{s}\}$$

Lemme 7 *Si Δ est insatisfiable, alors $\Delta(u)$ aussi, $\forall u$.*

Lemme 8 (Théorème de compacité) *Si Δ est insatisfiable, il existe une partie finie de Δ insatisfiable. On peut donc supposer par la suite Δ fini.*

Théorème 24 *Soit Δ un ensemble de clauses. Si Δ est insatisfiable, alors il existe un arbre de résolution qui réfute Δ.*

Preuve

On montre le résultat par récurrence sur le nombre n de variables apparaissant dans Δ.

cas $n = 0$ **:** $\Delta = \{\square\}$.

cas $n > 0$ **:** Δ est insatisfiable. Soit $u \in \mathcal{P}$ tel que u ou $\bar{u}$ figure dans Δ.

- Si $\bar{u}$ ne figure pas. On pose $I(u) = 1$ et on élimine les clauses contenant u.
- Si u ne figure pas. On pose $I(u) = 0$ et on élimine les clauses contenant $\bar{u}$.
- Si u et $\bar{u}$ figurent, alors $\Delta(u)$ et $\Delta(\bar{u})$ sont insatisfiables et ne comportent que $(n-1)$ variables. Donc il existe T_0 et T_1 qui réfutent (resp.) $\Delta(u)$ et $\Delta(\bar{u})$.

Regardons T_0 :

- Ses feuilles sont étiquetées par des clauses de $\Delta(u)$.
- Ni u ni $\bar{u}$ ne sont dans l'étiquette d'un noeud de T_0.

Soit $\tilde{T}_0$ obtenu à partir de T_0 comme suit :

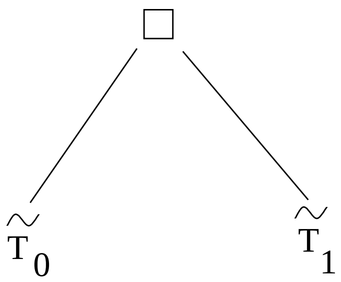
- On ajoute $\bar{u}$ aux étiquettes des feuilles de T_0 qui n'appartiennent pas à Δ.
- On ajoute $\bar{u}$ à un noeud interne si on l'a ajouté à l'un de ses fils.

Lemme 9 *L'arbre $\tilde{T}_0$ est un arbre de résolution à partir de Δ.*

De même on construit $\tilde{T}_1$ obtenu à partir de T_1 en ajoutant u aux feuilles qui n'appartiennent pas à Δ (et en faisant remonter aux pères).

On a alors deux cas :

- Toutes les feuilles de $\tilde{T}_0$ (ou $\tilde{T}_1$) sont étiquetées dans Δ. Alors $\tilde{T}_0 = T_0$ et a pour racine $\Box$.
- La racine de $\tilde{T}_0$ est étiquetée par $\bar{u}$. La racine de $\tilde{T}_1$ est étiquetée par u. Un arbre de résolution qui réfute Δ est :



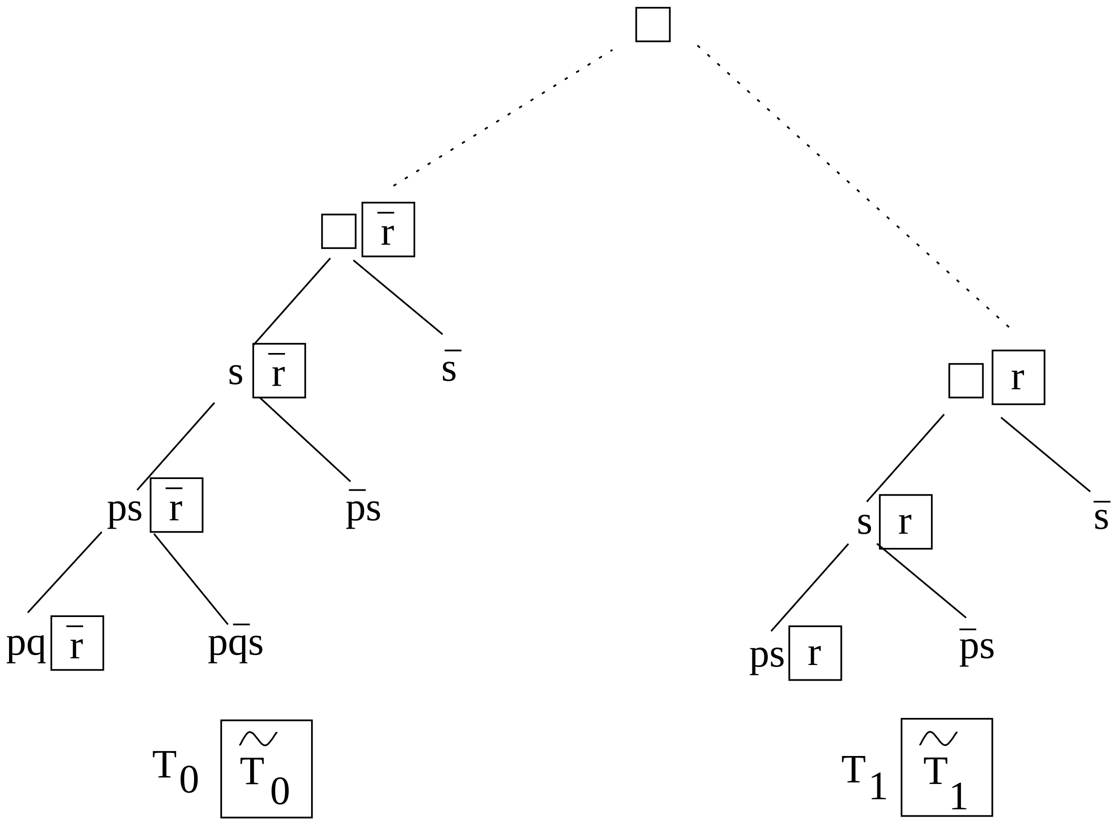
Corollaire 25 $\Delta \vdash \Box$ *ssi* $\Delta \models \Box$.

Exemple

$$\boxed{\Delta = \{pq\bar{r}, prs, p\bar{q}s, \bar{p}s, \bar{s}\}}$$

$$\Delta(r) = \{pq, p\bar{q}s, \bar{p}s, \bar{s}\}$$

$$\Delta(\bar{r}) = \{ps, p\bar{q}s, \bar{p}s, \bar{s}\}$$



Variantes sur la méthode de résolution

Méthode exhaustive

A partir de Δ, on construit $\Delta' = \Delta \cup$ toutes les résolvantes possibles. On itère cette construction jusqu'à trouver $\square$ ou à stabiliser. (Si Δ a n variables, il y a 2^{2n} clauses au plus).

Résolution restreinte

On écarte les clauses qui sont des tautologies (c.a.d qui contiennent v et $\bar{v}$). On les supprime de Δ et on ne les génère pas comme résolvantes.

Théorème 26 *La résolution restreinte est complète.*

Résolution ordonnée

On ordonne les variables et on n'autorise la résolution de C_1 et C_2 que modulo la variable p qui est minimale dans $At(C_1) \cup At(C_2)$.

Théorème 27 *La résolution restreinte et ordonnée est complète.*

Exemple de résolution restreinte et ordonnée

$$\Delta = \{1 = pq\bar{r}, 2 = prs, 3 = p\bar{q}s, 4 = \bar{p}s, 5 = \bar{s}\}$$
$$p < q < r < s$$

$p(1,4)$	$q\bar{r}s$	6
$p(2,4)$	rs	7
$p(3,4)$	$\bar{q}s$	8
$q(6,8)$	$\bar{r}s$	9
$r(7,9)$	s	10
$s(5,10)$	$\square$	`Arrêt`

$\boxed{\text{Résolution linéaire}}$

On construit une preuve $C_1, C_2, \ldots, C_n$ telle que ou bien $C_i \in \Delta$, ou bien C_i est une résolvante de C_{i-1} et d'un C_j avec $j < i$.

Théorème 28 *La résolution linéaire est complète.*

Calcul des prédicats

$\exists n.\exists x.\exists y.\exists z.$
$(n \geq 3) \wedge (x > 0) \wedge (y > 0) \wedge (z > 0) \wedge (x^n + y^n = z^n)$

Morphologie : les termes

On dit encore “termes du premier ordre”.

- X : ensemble de variables (dénombrable);
- O : ensemble d'opérateurs (fini);
- O_n : opérateurs d'arité n;
- $T(O, X)$ ou T : termes bâtis sur O et X :
  - Si $x \in X$, $x \in T$;
  - Si $a \in O_0$, $a \in T$;
  - Si $f \in O_n$, $t_1, \ldots, t_n \in T$, $f(t_1, \ldots, t_n) \in T$.

Les termes sont des arbres :

- récurrence sur les termes;
- schéma de définition par récurrence.

Exemples

- $\mathrm{Var}(t)$: ensembles des variables figurant dans le terme t.
- Soit la substitution $\sigma : X \longrightarrow T$ de support fini (support : $\{x \mid \sigma(x) \neq x\}$).
  On étend σ aux termes par :
  - $\sigma(a) = a$ si $a \in O_0$;
  - $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$ si $f \in O_n$, $t_i \in T$.

Notations : $\sigma(t)$, $t\sigma$, $t[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$.

Lemme 10 *Soit t un terme et σ, σ' deux substitutions telles que $\sigma(x) = \sigma'(x)$ pour tout x dans $\mathrm{Var}(t)$, alors $\sigma(t) = \sigma'(t)$.*

Définition : Un terme clos est un terme tel que $\mathrm{Var}(t) = \emptyset$.

Morphologie : les formules

On définit l'ensemble des formules logiques du premier ordre.

Ensembles : X, O, C (connecteurs logiques), $\mathcal{Q}$ (quantificateurs, $\subset \{\forall, \exists\}$), R (ensemble de symboles de relations ou symboles de prédicats).
Langage du premier ordre : Couple (O, R).

Ensemble de formules $\mathcal{F}$:

- Si $r \in R_n, t_1, \ldots t_n \in T$, alors $r(t_1, \ldots t_n) \in \mathcal{F}$. Ce sont les formules atomiques;
- Si $c \in C_0$, $c \in \mathcal{F}$;
- Si $c \in C_n, F_1, \ldots F_n \in \mathcal{F}$, $c(F_1, \ldots, F_n) \in \mathcal{F}$;
- Si $Q \in \mathcal{Q}, x \in X, F \in \mathcal{F}$, alors $((Q, x), F) \in \mathcal{F}$.

Attention : Les termes ne sont pas des formules.

Notations :
$((Q,x),F)$ ou QxF ou $Qx.F$.
$Qx_1.Qx_2.\dots Qx_k.F$ ou $Qx_1\dots x_k.F$.
Les formules sont des arbres.

- récurrence sur les formules;
- schéma de définition par récurrence.

Définitions :

- Une variable libre est une variable non liée par un quantificateur;
- L'ensemble des variables libres d'une formule F est noté $FV(F)$;
- Une formule close est une formule F telle que $FV(F)=\emptyset$.
- On appelle énoncé une formule close.

Si F a pour variables $x_1, x_2, \dots, x_n$, alors

- $\forall x_1 \dots x_n.F$ est une clôture universelle de F ;
- $\exists x_1 \dots x_n.F$ est une clôture existentielle de F.

Exemple

$$\forall x.\exists y.\ (x + y = x * x) \wedge (y \geq x)$$

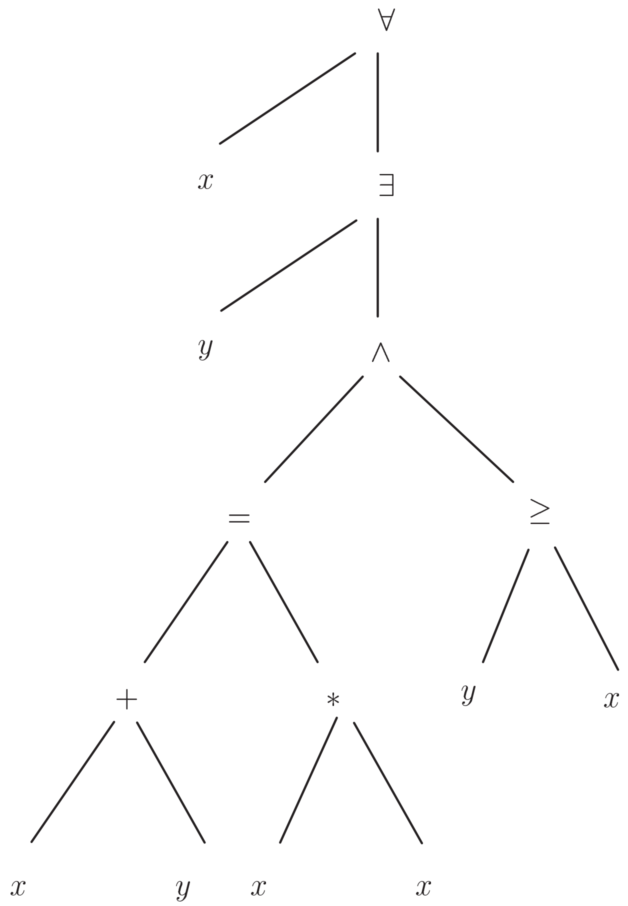


Figure 1: Formule du premier ordre.

L'opération de substitution sur les formules

Soit σ de X dans T, on note σ_x de X dans T l'application définie par :

- $\sigma_x(y) = \sigma(y)$ si $y \neq x$;
- $\sigma_x(x) = x$.

On étend σ aux formules de la façon suivante :

- Si $F \in R_0$, $\sigma(F) = F$;
- Si $F = r(t_1, \ldots, t_n), \sigma(F) = r(\sigma(t_1), \ldots, \sigma(t_n))$;
- Si $F \in C_0$, $\sigma(F) = F$;
- Si $F = c(F_1, \ldots, F_n), \sigma(F) = c(\sigma(F_1), \ldots, \sigma(F_n))$;
- Si $F = Qx.G$, alors $\sigma(F) = Qx.(\sigma_x(G))$.

Lemme 11 *Soit F une formule et σ, σ' deux substitutions telles que $\sigma(x) = \sigma'(x)$ pour tout x dans $FV(F)$, alors $\sigma(F) = \sigma'(F)$.*

Exemple

Soit R un relateur binaire et soient

$$F = \forall x.R(x, y),$$

$$F' = \forall z.R(z, y).$$

Du point de vue sémantique, on souhaite que F et F' aient même signification. On souhaite aussi que ceci reste vrai après une substitution. Or soit σ qui change y en $f(x)$,

$$G = \sigma(F) = \forall x.R(x, f(x));$$

$$G' = \sigma(F') = \forall z.R(z, f(x)).$$

On dit qu'un terme t est libre pour une variable x dans F si le problème ci-dessus ne se pose pas lors de la substitution de x par t.

- Si F est atomique, t est libre pour x dans F;
- Si $F = c(F_1, \ldots, F_n)$, t est libre pour x dans F ssi t est libre pour x dans chacun des F_i;
- Si $F = Qy.G$, t est libre pour x dans F si l'une des deux conditions suivantes est satisfaite :
  - $x \notin FV(F)$ (c'est-à-dire $x \notin FV(G)$ ou $x = y$);
  - $x \in FV(F)$ et $y \notin \mathrm{Var}(t)$.

On dit que σ est libre pour F si pour tout x dans le domaine de σ, $\sigma(x)$ est libre pour x dans F. On n'autorise que les substitutions libres pour une formule.

Notations :
Si F a toutes des variables libres et parmi $x_1, \ldots, x_n$, si $\sigma(x_1) = t_1$, ..., $\sigma(x_n) = t_n$, on écrit parfois

- $F(x_1, \ldots, x_n)$ au lieu de F;
- $F(t_1, \ldots, t_n)$ au lieu de $\sigma(F)$.

L'égalité

Un langage du premier ordre avec égalité est un langage dont l'ensemble de relateurs contient le symbole $=$.

Si t, t' sont deux termes, on note $t = t'$ pour $= (t, t')$.

Un traitement spécial est réservé à l'égalité.

- En sémantique, l'égalité est interprétée par l'égalité sur le domaine d'interprétation.
- En théorie de la démonstration, un certain nombre d'axiomes sont associés aux langages à égalité comme ceux exprimant le fait de l'égalité est une relation d'équivalence, et ceux indiquant que l'on peut remplacer un terme par un terme qui lui est égal sans changer le sens de la formule.

Sémantique

Le cas sans égalité.

Définitions :
Soit $\mathbf{B} = \{\mathbf{0}, \mathbf{1}\}$ l'ensemble des booléens ($\mathbf{0} = \mathtt{False}, \mathbf{1} = \mathtt{True}$).
On appelle structure ou interprétation un triplet $S = (D, \{\tilde{f} \mid f \in O\}, \{\tilde{c} \mid c \in C\})$, où

- Si $f \in O_n$, $\tilde{f} : D^n \longrightarrow D$;
- Si $r \in R_n$, $\tilde{r} : D^n \longrightarrow \mathbf{B}$.

L'ensemble D est le domaine d'interprétation. Pour interpréter une formule dans une structure, il faut attribuer de valeurs aux variables libres de cette formule. Cette opération est réalisée en définissant une valuation :

$$v : X \longrightarrow D$$

On étend tout d'abord v aux termes.

- $v(a) = \tilde{a}$ si $a \in O_0$;
- $v(f(t_1, \ldots, t_n)) = \tilde{f}(v(t_1), \ldots, v(t_n))$.

Lemme 12 *Soit t un terme et v, v' deux valuations, si $v(x) = v(x')$ pour tout x dans* $\mathrm{Var}(t)$ *alors $v(t) = v'(t)$.*

Si v est une valuation de X dans D, on note $v_{x \to d}$ la valuation de X dans D définie par :

$$v_{x \to d}(y) = \begin{cases} d & \text{si } y = x \\ v(y) & \text{si } y \neq x \end{cases}$$

On étend ensuite v aux formules par :

- $v(r(t_1, \ldots, t_n)) = \tilde{r}(v(t_1), \ldots, v(t_n))$;
- $v(c) = \tilde{c}$ si $c \in C_0$;
- $v(c(F_1, \ldots, F_n)) = \tilde{c}(v(F_1), \ldots, v(F_n))$;
- $v(\forall x.F) = \begin{cases} \mathbf{1} & \text{si } v_{x \to d}(F) = \mathbf{1} \text{ pour tout } d \in D \\ \mathbf{0} & \text{sinon} \end{cases}$
- $v(\exists x.F) = \begin{cases} \mathbf{1} & \text{s'il existe } d \in D \text{ tq } v_{x \to d}(F) = \mathbf{1} \\ \mathbf{0} & \text{sinon} \end{cases}$

Lemme 13 *Soit F une formule et v, v' deux valuations. Si $v(x) = v'(x)$ pour $x \in FV(F)$ alors $v(F) = v'(F)$.*

Le lemme justifie les notations suivantes. Soit F une formule dont les variables libres sont parmi $x_1, \dots x_n$. Si v est une valuation telle que $v(x_1) = v_1, \dots v(x_n) = v_n$, on écrit

- $F(x_1, \dots x_n)$ pour F;
- $F(v_1, \dots v_n)$ pour $v(F)$.

En effet, $v(F)$ ne dépend pas des valeurs prises par v sur les autres variables. Ce genre de notation mélange syntaxe et sémantique.

Définitions

Soit S une structure.

- Une formule F est valide dans S ssi pour toute valuation v, $v(F) = \mathbf{1}$.
  On écrit $S \models F$.
- Si E est un ensemble de formule, on écrit $S \models E$ si toute formule de E est valide dans S. On dit que E est valide dans S.
- On dit que F est valide si F est valide pour toute structure. On écrit $\models F$.
- Une formule F est satisfiable dans S s'il existe une valuation v telle que $v(F) = \mathbf{1}$.
- Une formule F est conséquence sémantique d'un ensemble de formules E ssi pour tout structure S et toute valuation v, si $v(G) = \mathbf{1}$ pour toute formule G de E, alors $v(F) = \mathbf{1}$. On écrit $E \models F$.

Exemple

$$F = \forall x . \exists y .\ (x + y = x * x) \wedge (y \geq x)$$

- Soit $S = (\mathbb{N}, \tilde{+}, \tilde{*}, \tilde{=}, \tilde{\geq})$. La formule F n'est pas valide pour S.
- Soit $S = (\mathbb{N}_{\geq 2}, \tilde{+}, \tilde{*}, \tilde{=}, \tilde{\geq})$, où $\mathbb{N}_{\geq 2}$ désigne les entiers supérieurs ou égaux à 2. La formule F est valide pour S.

Quelques propriétés

Proposition 29 *Une formule F est valide dans une structure ssi une quelconque de ses clôtures universelles est valide dans la structure.*

Proposition 30 *Soit v une valuation, σ une substitution. Soit v' la valuation définie sur X par $v' = v \circ \sigma$. Pour toute formule F telle que σ est libre pour F, $v(\sigma(F)) = v'(F)$.*

Preuve

La preuve est par récurrence sur F. Les cas de base et l'étape de la récurrence correspondant à une formule commençant par un connecteur sont faciles. On passe ensuite au cas où une formule commence par un quantificateur.

Supposons que $F = \forall x.G$.
Soient $x_1, \ldots . x_n$ les variables libres de G (on peut avoir x dedans).
Comme σ est libre pour F, $x \notin \mathrm{Var}(\sigma(x_i))$ pour tout i tq $x_i \neq x$.
On a :

$$\begin{aligned} \sigma(F) &= \forall x.(\sigma_x(G)). \\ v(\sigma(F)) &= \begin{cases} \mathbf{1} & \text{si } v_{x\to d}(\sigma_x(G)) = \mathbf{1} \text{ pour tout } d \in D \\ \mathbf{0} & \text{sinon} \end{cases} \end{aligned}$$

Par hypothèse de récurrence, $v_{x\to d}(\sigma_x(G)) = v''(G)$, où v'' est la valuation définie sur X par $v'' = v_{x\to d} \circ \sigma_x$.
Or pour tout i,

$$v_{x\to d} \circ \sigma_x(x_i) = \begin{cases} d & \text{si } x_i = x, \\ v \circ \sigma(x_i) & \text{sinon, car } x \notin \mathrm{Var}(\sigma(x_i)). \end{cases}$$

Ou :

$$v''(x_i) = v'_{x\to d}(x_i).$$

Par le lemme 13, on en déduit $v''(G) = v'_{x\to d}(G)$.
D'où le résulat car

$$v(\sigma(F)) = \begin{cases} \mathbf{1} & \text{si } v'_{x\to d}(G) = \mathbf{1} \text{ pour tout } d \in D, \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Exemple avec σ non libre pour F

Soit

$$F = \forall x.(x \leq y).$$

Soit S la structure $(\mathbb{N}, \tilde{\leq})$.
Soit σ telle que $\sigma(y) = x$.

$$\sigma(F) = \forall x.(x \leq x).$$

Soit v définie par :

$$\begin{cases} v(y) = 1, \\ v(x) = 2. \end{cases}$$

On a $v'(y) = 2$ d'où $v(\sigma(F)) = \mathbf{1}$ et $v'(F) = \mathbf{0}$.

Principes valides dans toute structure

Dans ce qui suit, F, G sont des formules, E est un ensemble de formules.

Proposition 31 *$E \cup F \models G$ ssi $E \models F \to G$.*

Proposition 32 *Si $E \models \forall x.F$, alors $E \models F[x \leftarrow t]$ pour tout terme t.*

Proposition 33 *Si $E \models F$ et si $x \in X$ n'est libre dans aucune formule de E, alors $E \models \forall x.F$.*

Proposition 34 *Soit t un terme et x une variable. Si $E \models F[x \leftarrow t]$, alors $E \models \exists x.F$.*

Proposition 35 *Soit x une variable ne figurant pas libre dans G, ni dans aucune formule de E. Si $E, F \models G$, alors $E, \exists x.F \models G$.*

Si A est une formule du calcul propositionnel d'atomes propositionnels $p_1, \ldots p_n$ et si $F_1, \ldots F_n$ sont des formules, on définit la formule $A[p_1 \leftarrow F_1, \ldots, p_n \leftarrow F_n]$.
(exercice : définir ceci rigoureusement).

Proposition 36 *Si A est une tautologie, $A[p_1 \leftarrow F_1, \ldots, p_n \leftarrow F_n]$ est valide.*

Ces formules $A[p_1 \leftarrow F_1, \ldots, p_n \leftarrow F_n]$, où A est une tautologie du calcul propositionnel, sont appelées tautologies du calcul des prédicats.

Exemples :

$(x \geq y) \rightarrow (x*x-y \leq x) \rightarrow (x \geq y)$ tautologie.

$\forall x.(x \geq x)$ formule valide.

Deux formules F, F' sont sémantiquement équivalentes si $F \leftrightarrow F'$ est valide.

Proposition 37 *Soit F une formule, Q est un quantificateur.*

- $\exists x.F$ *et* $\neg(\forall x.\neg F)$ *sont sémantiquement équivalentes.*
- $\forall x.F$ *et* $\neg(\exists x.\neg F)$ *sont sémantiquement équivalentes.*

Le cas avec égalité

On dit que $v(t = t') = \mathbf{1}$ ssi $v(t) = v(t')$.

Proposition 38 *Pour tout $x, y, z \in X$,*

- $\models x = x$,
- $\models x = y \rightarrow y = x$,
- $\models (x = y \wedge y = z) \rightarrow x = z$.

Pour $x_1, \ldots x_n, y_1, \ldots y_n \in X$, t un terme, F une formule,

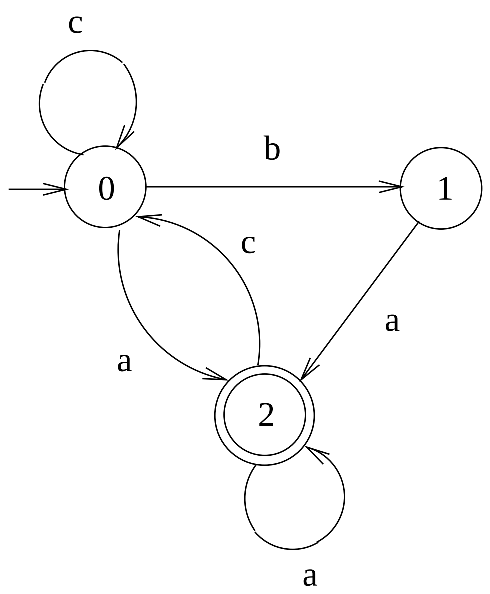
- $\models (y_1 = x_1 \wedge \ldots \wedge y_n = x_n) \rightarrow t[x_1 \leftarrow y_1, \ldots, x_n \leftarrow y_n] = t$,
- $\models (y_1 = x_1 \wedge \ldots \wedge y_n = x_n) \rightarrow (F[x_1 \leftarrow y_1, \ldots, x_n \leftarrow y_n] \leftrightarrow F)$.

à condition que la substitution soit libre (y_i libre pour x_i dans F).

Automates et logique

On décrit des langages de mots, d'arbres, ..., par des formules logiques.

Exemple 1



$$
\begin{aligned}
F_1 = & \ \neg \ \exists x \exists y (S(x,y) \land Q_a(x) \land Q_b(y)) \\
& \land \ \forall x (Q_b(x) \to \exists y (S(x,y) \land Q_a(y)) \\
& \land \ \exists x (\neg \exists y S(x,y) \land Q_a(x))
\end{aligned}
$$

L'alphabet est $A = \{a, b, c\}$.

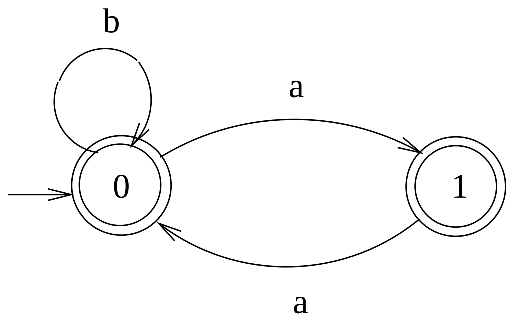
- aucun a n'est suivi d'un b;
- b est toujours suivi de a;
- la dernière lettre est un a.

Les lettres x, y sont des variables de positions. La formule $S(x, y)$ indique que la position y suit la position x. Le prédicat $Q_a(x)$ indique qu'il y a un a en position x.

La formule $\neg\exists y S(x, y)$ indique que x est la dernière position. La formule F_1 est une formule du premier ordre.

Exemple 2

Les variables $X, Y, \ldots$ désignent des ensembles de positions. La formule atomique $X(y)$ signifie $y \in X$.



Entre deux b il y a toujours un nombre pair de a.

$F_2 = \forall x \forall y$
$(Q_b(x) \wedge (x < y) \wedge Q_b(y)$
$\wedge \forall z(x < z \wedge z < y \to \neg Q_b(z))$
$\to \exists X(X(x) \wedge X(y) \wedge$
$\forall u \forall v(S(u, v) \to (X(u) \leftrightarrow \neg X(v))).$

La formule F_2 est une formule du second ordre.

Les modèles de mot

- A alphabet;
- $w = a_0 a_1 \dots a_{n-1}$ un mot sur A.

Le mot w est représenté par la structure :

$$\underline{w} = (\mathrm{dom}(w), S^w, <^w, (Q_a^w)_{a \in A}).$$

appelée modèle de mot.

- $\mathrm{dom}(w) = \{0, 1, \dots, n-1\}$ est l'ensemble des positions de w;
- S^w est la relation successeur sur $\mathrm{dom}(w)$: $(i, i+1) \in S^w$ pour $0 \leq i \leq n-1$;
- $<^w$ est l'ordre naturel sur $\mathrm{dom}(w)$;
- Q_a^w est un prédicat unaire indiquant les positions qui sont des a; $Q_a^w = \{i \in \mathrm{dom}(w) \mid a_i = a\}$. (prédicat de lettre).

Logique du premier ordre

On décrit le langage du premier ordre

- Variables : $x, y, \ldots$: elles représentent les positions dans les modèles de mots;
- Ensemble de relations :
  - $x = y$,
  - $S(x, y)$,
  - $x < y$,
  - $Q_a(x)$, $a \in A$;
- Connecteurs $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$;
- Quantificateurs $\exists, \forall$.

Un énoncé est une formule sans variable libre. On parle de signature $(S, <, Q_a)$, l'égalité est implicite.

Sémantique

Si les variables libres de F sont parmi $x_1, x_2, \ldots x_n$, on écrit $F(x_1, \ldots, x_n)$ au lieu de F.
Si $p_1, \ldots, p_n$ sont des positions dans dom(w) alors on écrit

$$(\underline{w}, p_1, \ldots, p_n) \models F(x_1, \ldots, x_n)$$

si F est valide dans $\underline{w}$ pour la valuation qui associe p_i à x_i.
Par convention $\underline{\varepsilon}$ satisfait $\forall x F(x)$ mais pas $\exists x F(x)$.

Le langage défini par une formule close F est :

$$L(F) = \{w \in A^* \mid \underline{w} \models F\}.$$

Quelques macros

Le relateur $S(x, y)$ peut être supprimé si on dispose de $<$:

$$x < y \wedge \neg\exists z(x < z \wedge z < y).$$

On peut ajouter les prédicats :

- first(x) : x est la première position, $\neg\exists y S(y, x)$;
- last(x) : x est la dernière postition, $\neg\exists y S(x, y)$.

On peut ajouter des symboles de fonctions comme $y = \mathrm{succ}(\mathrm{succ}(c))$ pour $\exists z(S(x, z) \wedge S(z, y))$.

Logique monadique du second ordre

- X, Y variables d'ensembles;
- $X(y)$ signifie $y \in X$.

Cette logique est appelée logique MSO.

On peut se passer de $<$ si on a S : $x < y$ s'écrit :

$\neg x = y$

$\wedge \forall X (X(y) \wedge \forall z \forall z' (X(z) \wedge S(z', z) \to X(z')) \to X(x))$

Sémantique de MSO

Si les variables libres de F sont parmi $X_1, X_2, \ldots X_n$, on écrit $F(X_1, \ldots, X_n)$ au lieu de F.
Une formule est interprétée dans un modèle de mot avec n sous-ensembles $P_1, \ldots, P_n$ de l'ensemble des positions dans $\mathrm{dom}(w)$.
Un tel modèle représente un mot dans l'alphabet $A' = A \times \{0, 1\}^n$. L'étiquette $(a, c_1, \ldots, c_n)$ à la position p indique que la lettre à la position p est a, et la position p appartient à l'ensemble P_j ssi $c_j = 1$.
On écrit

$$(\underline{w}, P_1, \ldots, P_n) \models F(X_1, \ldots, X_n)$$

si F est valide dans $\underline{w}$ pour la valuation qui associe P_i à X_i.

Exemple $w = abbaaa$. P_1 : positions paires, P_2 positions qui sont des nombres premiers.

$$\begin{array}{ccccccc} w & a & b & b & a & a & a \\ P_1 & 1 & 0 & 1 & 0 & 1 & 0 \\ P_2 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}$$

Le langage défini par une formule F est :

$$L(F) = \{w \in (A \times \{0,1\}^n)^* \mid (\underline{w}, P_1, \ldots, P_n) \models F\}.$$

Automates et logique MSO

Théorème 39 *Un langage de mots finis est reconnaissable ssi il est définissable en logique MSO.*

Preuve

$\Rightarrow$

On part d'un automate déterministe $\mathcal{A} = (Q, E, i, T)$.
Les états sont $Q = \{0, 1, \ldots, k\}$.
L'état initial a le numéro 0.
Les variables X_i représentent des ensembles disjoints de sous-ensembles de $\{0, 1 \ldots, n-1\}$ contenant les positions de w correspondant à l'état i de l'automate.

L'automate $\mathcal{A}$ accepte un mot non vide w ssi

$$
\begin{aligned}
\underline{w} \models \ \exists \ & X_0 \dots X_k \\
& (\ \bigwedge_{i \neq j} \forall x \neg (X_i(x) \wedge X_j(x)) \\
& \wedge \ \forall x (\mathrm{first}(x) \to X_0(x)) \\
& \wedge \ \forall x \forall y (S(x,y) \to \bigvee_{(i,a,j) \in E} (X_i(x) \wedge Q_a(x) \wedge X_j(y))) \\
& \wedge \ \forall x (\mathrm{last}(x) \to \bigvee_{\exists j \in T : (i,a,j) \in E} (X_i(x) \wedge Q_a(x)))
\end{aligned}
$$

$\Leftarrow$
On utilise la logique MSO_0 qui a même puissance d'expression que MSO et est obtenue à partir de MSO en enlevant le premier ordre.

- $X \subset Y$;
- $\mathrm{Sing}(X)$ (X est un singleton);
- $\mathrm{Succ}(X, Y)$ (X et Y sont deux singletons $\{x\}$ et $\{y\}$ et $S(x, y)$);
- $X \subset Q_a$ (Toutes les lettres aux positions de X sont des a).

Exemple de passage de MSO à MSO_0:

$$\forall x(Q_a(x) \to \exists y(S(x, y) \wedge Z(y))$$

se réécrit : $\forall X(\mathrm{Sing}(X) \wedge X \subset Q_a$
$\to \exists Y(\mathrm{Sing}(Y) \wedge \mathrm{Succ}(X, Y) \wedge Y \subset Z))$

Il faut montrer, que pour toute formule $F(X_1, X_2, \ldots, X_n)$, il existe un automate qui accepte les mots $w \in (A \times \{0,1\}^n)^*$ qui satisfont F.
Les mots sont représentés par les modèles de mots $(\underline{w}, P_1, \ldots, P_n)$.

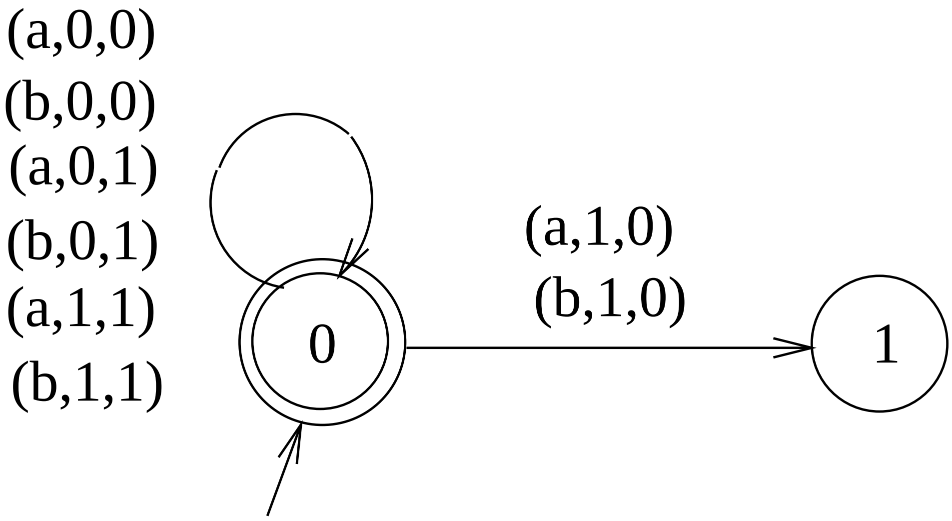
Formules atomiques

- $X_j \subset X_k$;
- $\mathrm{Sing}(X_j)$;
- $\mathrm{Succ}(X_j, X_k)$;
- $X_j \subset Q_a$.

Le passage aux formules complexes commençant par un connecteur $\wedge$ ou $\neg$ ne pose pas de problème. On utilise les opérations booléennes sur les automates.

Exemple de formule atomique

On prend $n = 2$, $j = 1$, $k = 2$, $A = \{a, b\}$.
Automate reconnaissant $L(X_1 \subset X_2)$.



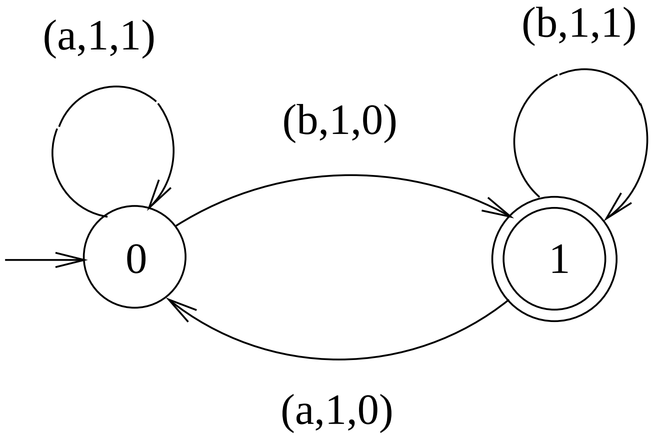
Exemple de formule avec quantificateur $\exists$

Soit $G(X_1, \ldots, X_n)$ sur $A \times \{0,1\}^n$ reconnu par un automate $\mathcal{A}$. La formule

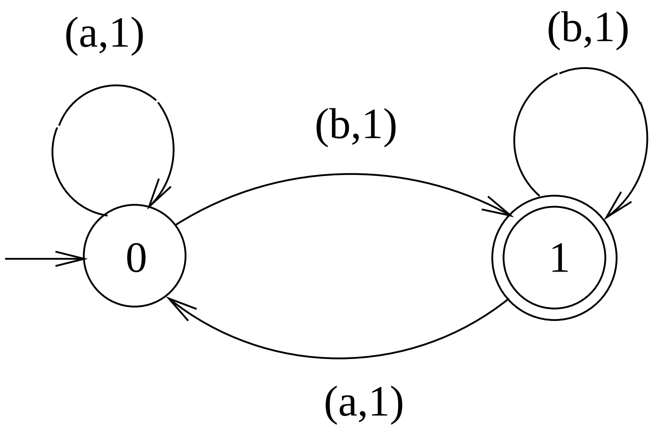
$$F(X_1, \ldots, X_{n-1}) = \exists X_n G(X_1, \ldots, X_n)$$

est reconnu par un automate $\mathcal{B}$.

Soit $\mathcal{A}$ qui reconnaît $L(F(X_1, X_2))$.

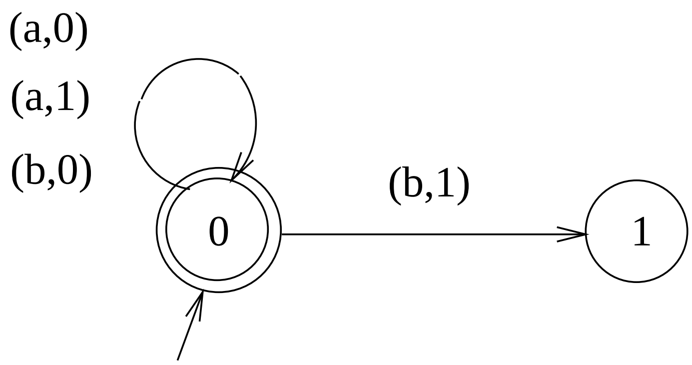


Alors $\mathcal{B}$ ci-dessous reconnaît $L(\exists X_2 F(X_1, X_2))$

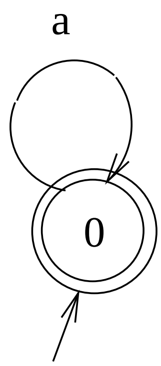


Exemple de formule avec quantificateur $\forall$

Soit $\mathcal{A}$ qui reconnaît $L(F(X_1))$, $F = X_1 \subset Q_a$.

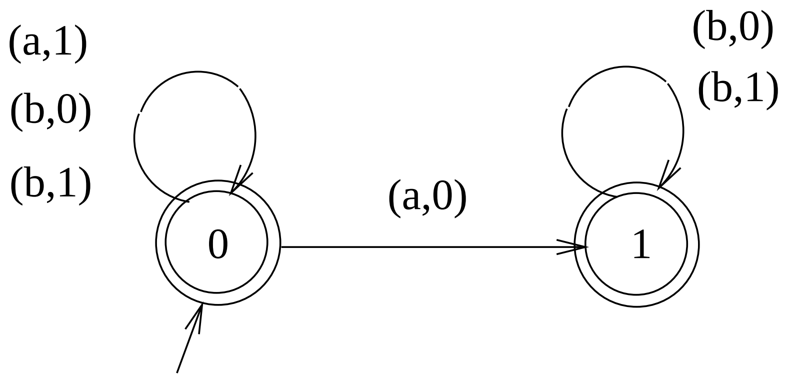


Alors $\mathcal{B}$ ci-dessous reconnaît $L(\forall X_1 F(X_1))$

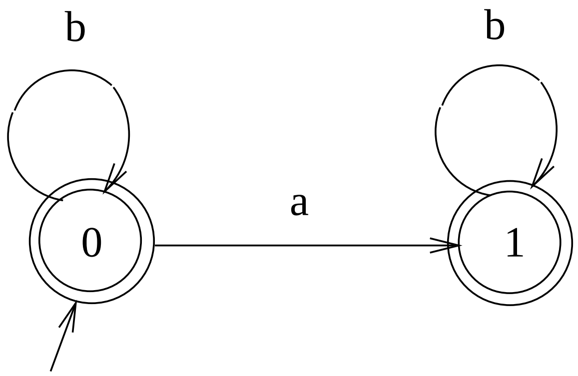


Autre exemple de formule avec quantificateur $\forall$

Soit $\mathcal{A}$ qui reconnaît $L(F(X_1))$.



Alors $\mathcal{B}$ ci-dessous reconnaît $L(\forall X_1 F(X_1))$



Trouver l'algorithme de construction.