Logique MP2I

Victor ROBERT

March, 2025

Syntaxe de la logique propositionelle

Définition

L'ensemble $\mathcal{P}$ des formules propositionelles est défini par induction par:

$\mathcal{B}$ contient $\top$, $\bot$ et un certain nombre de variables propositionelles dont on note $\mathcal{V}$ l'ensemble.

$\mathcal{C}$ contient un constructeur d'arité 1
- $\neg : \mathcal{P} \to \mathcal{P}$

et 4 constructeurs d'arité 2:
- $\wedge : \mathcal{P}^2 \to \mathcal{P}$
- $\vee : \mathcal{P}^2 \to \mathcal{P}$
- $\implies : \mathcal{P}^2 \to \mathcal{P}$
- $\iff : \mathcal{P}^2 \to \mathcal{P}$

Notation

On utilise souvent φ, ψ, θ pour noter les formules

Remarque

- $\top$ représente le vrai
- $\bot$ représente le faux
- $\vee$ représente le ou
- $\wedge$ représente le et
- $\neg$ représente le non
- $\implies$ représente l'implication
- $\iff$ représente l'équivalence

Exemple

On peut déjà exprimer des propositions logiques avec ce formalisme.
"Soit il pleut et le facteur ne passe pas, soit il fait beau et le facteur passe."

On définit une variable p qui représente *"il pleut"* et une variable f qui va représenter *"le facteur passe"*:

$$\Big[(p \wedge \neg f) \vee (\neg p \wedge f)\Big] \wedge \Big[(p \wedge \neg f) \wedge (\neg p \wedge f)\Big]$$

Définition

Pour φ une formule propositionelle et $\mathcal{A}$ son AST associé
- sa *hauteur* est $h(\mathcal{A})$
- sa *taille* est $n(\mathcal{A})$
- une *sous-formule* de φ est un sous-arbre de $\mathcal{A}$

Définition (Egalité syntaxique)

Deux formules sont égales si elles ont le même AST associé.

Sémantique du calcul propositionel

On va donner du sens aux formules

Evalutation d'une proposition logique

Définition

On définit l'*Algèbre de Boole*: $\mathbb{B} := \{V,\ F\}$

On peut définir sur cet ensemble les opérations *ou*, *et*, *non*, *implique* et *équivaut* caractérisées par leur table de vérité.

Définition

Un valutation v est une fonction de $\mathcal{V}$ dans $\mathbb{B}$.

Avec ces deux définitions on définit la valeur d'une formule.

Définition

Soit v une valutation. *L'évalutation* d'une formule φ dans le contexte v, noté $[\![\varphi]\!]_v$

$[\![\varphi]\!]_v$ est définie par:

- $[\![\top]\!]_v = V$
- $[\![\bot]\!]_v = F$
- $[\![x]\!]_v = v(x)$
- $[\![\neg\psi]\!]_v = non\Big([\![\psi]\!]_v\Big)$
- $[\![\psi \wedge \theta]\!]_v = et\Big([\![\psi]\!]_v, [\![\theta]\!]_v\Big)$
- $[\![\psi \vee \theta]\!]_v = ou\Big([\![\psi]\!]_v, [\![\theta]\!]_v\Big)$
- $[\![\psi \implies \theta]\!]_v = implique\Big([\![\psi]\!]_v, [\![\theta]\!]_v\Big)$
- $[\![\psi \iff \theta]\!]_v = équivaut\Big([\![\psi]\!]_v, [\![\theta]\!]_v\Big)$

Remarque

$[\![\varphi]\!]_v$ est une fonction de $\mathcal{P}$ dans $\mathbb{B}$

Définition

On dit qu'une formule φ est satisfaite par une valutation v si

$$[\![\varphi]\!]_v = V$$

On dit aussi que v est une modèle de φ

Remarque

Si une formule φ contient n variables, il faut tester 2^n valutations pour savoir si φ admet un modèle dans le pire cas.

En effet, le graphe d'une valuation est un élement de $\mathbb{B}^n$

Conséquence logique

Dans cette partie on va exprimer la relation de conséquence logique. C'està dire que si une formule φ est vraie, alors une autre formule ψ l'est également.

Définition

On prend φ et ψ deux formules propositionelles. ψ est conséquence de φ, ce qu'on note $\varphi \vDash \psi$ si tout modèle de φ est une modèle de ψ.

C'est à dire

$$\forall v \in \mathcal{V},\ [\![\varphi]\!]_v = V \implies [\![\psi]\!]_v = V$$

Pour $\Gamma := \{\varphi_1, ..., \varphi_n\}$ un ensemble de propositions, ψ est conséquence logique de Γ, noté $\Gamma \vDash \psi$ si pour tout modèle de Γn c'est un modèle de ψ.

C'est à dire

$$\forall v \in \mathcal{V},\ [\![\varphi_1]\!]_v = ... = [\![\varphi_n]\!]_v = V \implies [\![\psi]\!]_v = V$$

Définition

Deux formules propositionelles φ et ψ sont sémantiquement équivalentes si $\varphi \vDash \psi$ et $\psi \vDash \varphi$

On note l'équivalence sémantique $\varphi \equiv \psi$

Remarque

Cela revient à dire que $\forall v \in \mathcal{V},\ [\![\varphi]\!]_v = [\![\psi]\!]_v$

Propriété

Si $\varphi \equiv \varphi'$ et $\psi \equiv \psi'$ alors:

- $\varphi \wedge \psi \equiv \varphi' \wedge \psi'$
- $\varphi \vee \psi \equiv \varphi' \vee \psi'$
- $\varphi \implies \psi \equiv \varphi' \implies \psi'$
- $\neg\varphi \equiv \neg\varphi'$

Preuve

Montrons que $\varphi \wedge \psi \equiv \varphi' \equiv \psi'$.
Soit v une valuation quelconque. On a:

$$[\![\varphi' \wedge \psi']\!]_v = et\Big([\![\varphi']\!]_v, [\![\psi']\!]_v\Big)$$

Or $\varphi \equiv \varphi'$ et $\psi \equiv \psi'$ donc $[\![\varphi]\!]_v = [\![\varphi']\!]_v$ et $[\![\psi]\!]_v = [\![\psi']\!]_v$ donc

$$[\![\varphi' \wedge \psi']\!]_v = et\Big([\![\varphi]\!]_v, [\![\psi]\!]_v\Big) = [\![\varphi \wedge \psi]\!]_v$$

Satisfiabilité

Définition

Une formule φ est *satifisfiable* s'il existe $v : \mathcal{V} \to \{V, F\}$ telle que $[\![\varphi]\!]_v = V$

On appelle un tel v un *témoin de satisfiabilité*

Définition

Une *tautologie* est une formule φ telle que

$$\forall v : V \to \mathbb{B},\ [\![\varphi]\!]_v \vDash \varphi$$

On peut noter $\vDash \varphi$

Remarque

"φ est une tautologie" est équivalent à $\varphi \equiv \top$

Propriété

Quelques équivalences sémantiques classiques:

- $\varphi \vee \varphi \equiv \varphi$
- $\varphi \wedge \varphi \equiv \varphi$
- $\neg\neg\varphi \equiv \varphi$
- $\varphi \vee \neg\varphi \equiv \top$ *(Tiers exclu)*
- $\varphi \wedge \neg\varphi \equiv \bot$ *(Absurde)*
- $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
- $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
- $\varphi \wedge (\psi \vee \theta) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \theta)$
- $\varphi \vee (\psi \wedge \theta) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \theta)$
- $\varphi \wedge \psi \equiv \psi \equiv \varphi$
- $\varphi \vee \psi \equiv \psi \equiv \varphi$
- $(\varphi \vee \psi) \vee \theta \equiv \varphi \vee (\psi \vee \theta)$
- $(\varphi \wedge \psi) \wedge \theta \equiv \varphi \wedge (\psi \wedge \theta)$

Exercice

Montrer que $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$

$[\![\varphi]\!]_v$	$[\![\psi]\!]_v$	$[\![\neg(\varphi \wedge \psi)]\!]_v$	$[\![\neg\varphi \wedge \neg\psi]\!]_v$
V	V	F	F
V	F	F	F
F	V	F	F
F	F	V	V

Définition

Soient φ et ψ des formules propositionelles. Soit x une variable.

On note $\varphi[x\backslash\psi]$ la *substitution de x par ψ dans la formule φ*.

Cette opération est définie par induction:

- Si x n'apparait pas dans φ, $\varphi\Big[x\backslash\varphi\Big] = \varphi$
- $x\Big[x\backslash\psi\Big] = \psi$
- $\Big(\neg\varphi\Big)\Big[x\backslash\psi\Big] = \neg\Big(\varphi\Big[x\backslash\psi\Big]\Big)$
- $\forall\Diamond \in \{\wedge, \vee, \implies, \iff\}$, $\Big(\varphi_1\Diamond\varphi_2\Big)\Big[x\backslash\psi\Big] = \varphi_1\Big[x\backslash\psi\Big]\Diamond\varphi_2\Big[x\backslash\psi\Big]$

Problème SAT

Formule normale

La formule normale d'une formule est une manière de l'écrire avec les $\vee$ et $\wedge$ regroupés, ce qui permet un meilleur traitement informatique des formules.

Définition

Un *littéral* est une formule formée d'une variable avec un éventuel $\neg$.

Définition

Une *clause* (sous-entendue disjonctive) est une formule qui est une disjonction ($\vee$) de littéraux. C'est à dire une formule de la forme:

$$\bigvee_i \ell_i$$

où (ℓ_i) est une famille de littéraux.

Définition

Une *clause conjonctive* est une formule qui est une conjonction ($\wedge$) de littéraux. C'est à dire une formule de la forme:

$$\bigwedge_i \ell_i$$

où (ℓ_i) est une famille de littéraux.

Définition

Une *forme normale conjonctive* (resp. *disjonctive*) est une formule qui est une conjonction de clauses (resp. disjonction de clauses conjonctives). C'est à dire une formule de la forme:

$$\bigwedge_i \bigvee_j \ell_{ij}$$

(resp. $\bigvee_i \bigwedge_j \ell_{ij}$)

où (ℓ_{ij}) est une famille de littéraux.

Exemple

Soit $\varphi = ((x \wedge \neg y) \vee \neg z) \wedge y$

Une FND de φ est $\varphi_1 = (x \wedge \neg y \wedge y) \vee (\neg z \wedge y)$

Une FNC de φ est $\varphi_2 = (x \vee \neg z) \wedge (\neg y \vee \neg z) \wedge y$

Définition

On appelle *forme normale (conjonctive ou disjonctive) canonique de* φ une forme FNC/FND de φ telle que toutes les clauses (conjonctives ou disjonctives) contiennent une et une seule fois chaque variable.

Exemple

Une FNC canonique φ est $\varphi_3 = (\neg z \wedge y \wedge x) \vee (\neg z \wedge y \wedge \neg x)$

Remarque

La première étape pour trouver une FNC ou FND d'une formule, c'est de supprimer les $\Longrightarrow$ et $\Longleftrightarrow$

On peut montrer que:

- $\varphi \Longrightarrow \psi \equiv \neg\varphi \vee \psi$
- $\varphi \Longleftrightarrow \psi \equiv (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)$

En remplaçant tous les $\Longrightarrow$ et $\Longleftrightarrow$ dans une formule de cette manière, on obtient une formule sémantiquement équivalente.

Remarque

Si $\varphi \equiv \psi$ et $\psi \equiv \theta$ Alors $\varphi \equiv \theta$.

On considère une désormais une formule φ contenant uniquement $\wedge$, $\vee$, $\neg$

Méthode n°1 pour trouver une FND

1. Ecrire la table de vérité de φ
2. Pour chaque ligne telle que $[\![\varphi]\!]_v = V$ on ajoute à la FND une clause conjonctive définie ainsi: Pour chaque variable x_i, si $[\![x_i]\!]_v = V$ alors la clause conjonctive contient le littéral x_i, sinon elle contient le littéral $\neg x_i$

Exemple: Soit $\varphi = \neg(x \vee \neg y) \wedge z$

$[\![x]\!]_v$	$[\![y]\!]_v$	$[\![z]\!]_v$	$[\![x \vee \neg y]\!]_v$	$[\![\varphi]\!]_v$
V	V	V	V	F
V	V	F	V	F
V	F	V	V	F
V	F	F	V	F
F	V	V	F	V
F	V	F	F	F
F	F	V	V	F
F	F	F	V	F

$(\neg x \wedge y \wedge z) \vee ...$

Méthode n°2

Utiliser les lois de De Morgan et la distributivité entre $\wedge$ et $\vee$.

Exemple: $\neg(x \vee \neg y) \wedge z \equiv (\neg x \wedge y) \wedge z$

Méthode n°3

Trouver une FND de $\neg\varphi$

$$\neg\varphi \equiv \bigvee_i \bigwedge_j \ell_{ij}$$

puis

$$\neg\neg\varphi \equiv \neg\Big(\bigvee_i \bigwedge_j \ell_{ij}\Big) \equiv \bigwedge_i \bigvee_j \neg\ell_{ij} \equiv \varphi$$

Le problème SAT

Définition

Le *problème SAT* prend en entrée une formule φ et doit déterminer si elle est satisfiable.

Remarque

Le problème SAT donne naissance à d'autres problèmes semblables:

- k-SAT: prendre en entrée une formule sous forme FNC telle que toutes les clauses contiennent au plus k littéraux et renvoit si elle est satisfiable ou pas. Par exemple $\varphi = (x_1 \vee \neg x_2) \wedge (x_3 \vee x_4) \wedge \neg x_1$ est une entrée possible pour 2-SAT (et k-SAT pour $k \geq 2$)

- MAX-SAT: on prend une formule quelconque sous forme de FNC. Il faut déterminer le nombre maximal de clauses que l'on peut satisfaire un même temps. Par exemple pour $\varphi = x_1 \wedge \neg x_1 \wedge (x_2 \vee x_3) \wedge x_3$, on peut rendre vraies x_1, $x_2 \vee x_3$ et x_3 en même temps avec:

$$\begin{aligned} v : \mathcal{V} &\to \mathbb{B} \\ x_1 &\mapsto V \\ x_2 &\mapsto F \\ x_3 &\mapsto V \end{aligned}$$

- HORN-SAT: on prend en entrée une formule φ en FNC telle que chaque clause peut contenir au plus un littéral positif (de la forme $\ell = x$ et pas $\ell = \neg x$) et on détermine si elle est satisfiable. Par exemple $\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge x_3$ est une entrée possible pour le problème HORN-SAT.

Remarque

On peut considérer que l'entrée du problème SAT est sous forme FNC.

Remarque

Les problèmes de satisfaction de contraintes se modèlisent par des formules logiques. Alors en étudiant la satisfiabilité de leur formules on peut déterminer s'ils ont des solutions. Par exemple on peu técrire une formule propositionelle qui exprime les règles du Sudoku et dont les variables représentent "Pour la case $(i, j) \in [\![0, 9]\!]^2$, elle vaut $k \in [\![1, 9]\!]$"

Propriété

La résolution par force brute du SAT coûte dans le pire cas $O(2^n)$ où n est le nombre de variables apparaissant dans la formule.

Algorithme de Quine

Soit $\mathcal{P}$ la formule prise en entrée. On note $x_1, ..., x_n$ les variables apparaissant dans $\mathcal{P}$. Alors SAT est un problème de satisfaction de contraintes dont les variables sont les $(x_i)_{1 \leq i \leq n}$,le domaine est $\mathbb{B}$ et la contrainte est "Rendre $\mathcal{P}$ vraie".

On peut appliquer le principe du retour sur trace

Exemple

$\mathcal{P} = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3)$

Si $x_1 = V$: $\mathcal{P} \longrightarrow (\top \vee x_2) \wedge (\neg\top \vee x_3) \longrightarrow \top \wedge x_3 \longrightarrow x_3$

Règles de simplifications de Quine

- $\neg\bot \equiv \top$
- $\neg\top \equiv \bot$
- $\forall Q(\top \vee Q \equiv \top)$
- $\forall Q(\bot \vee Q \equiv Q)$
- $\forall P((\top \implies P) \equiv P)$
- $\forall P((P \implies \top) \equiv \top)$
- $\forall P((\bot \implies P) \equiv \top)$
- $\forall P((P \implies \bot) \equiv \neg P)$
- $\forall P((\top \iff P) \equiv P)$
- $\forall P((\bot \iff P) \equiv \neg P)$

+ les commutations

Remarque

> La formule à droite de $\equiv$ est toujours strictement plus petite (en nombre de constructeurs) que celle à gauche.

Remarque

> Toute formule ne contenant pa de variable se ramène à $\top$ ou $\bot$ en utilisant ces simplifications locales.

Définition

Soient $\mathcal{P}$ une formule, φ une sous-formule de $\mathcal{P}$ n'apparaissant qu'une fois et soit φ' une autre formule.
Alors on appelle *remplacement* de φ par φ', noté $\mathcal{P}\Big[\varphi := \varphi'\Big]$ la formule $\mathcal{P}$ où le sous-arbre correspondant à φ à été remplacé par celui correspondant à φ'.

Remarque

C'est de la substitution d'une variable.

Propriété

Soient $\mathcal{P}$ une formule, φ une sous-formule apparaissant une unique fois et φ' une formule telle que $\varphi' \equiv \varphi$. Alors $\mathcal{P} \equiv P\Big[\varphi := \varphi'\Big]$

Preuve

Par induction structurelle sur la structure de $\mathcal{P}$

Cas de base
Si $\mathcal{P} = \varphi$, alors $\mathcal{P}\Big[\varphi := \varphi'\Big] = \varphi' = \varphi = \mathcal{P}$

Hérédité
Si $P = \neg Q$ et $Q \neq \varphi$, en supposant l'hypothèse d'induction pour Q, alors $\mathcal{P}\Big[\varphi := \varphi'\Big] = \neg Q\Big[\varphi := \varphi'\Big]$

Si $\mathcal{P} = \neg\varphi$, alors $\mathcal{P}\Big[\varphi := \varphi'\Big] = \neg\varphi' \equiv \neg\varphi = \mathcal{P}$

Si $\mathcal{P} = \mathcal{P}_1 \wedge \mathcal{P}_2$ et $\mathcal{P}_1, \mathcal{P}_2 \neq \varphi$ et on suppose l'hypothèse d'induction sur la sous-formule qui contient φ, disons que c'est $\mathcal{P}_1$, alors $\mathcal{P}\Big[\varphi := \varphi'\Big] = \mathcal{P}_1\Big[\varphi := \varphi'\Big] \wedge \mathcal{P}_2 \equiv \mathcal{P}_1 \wedge \mathcal{P}_2$ puisque $\mathcal{P}_1\Big[\varphi := \varphi'\Big] \equiv \mathcal{P}_1$

et de même pour $\vee$

Remarque

On a montré qu'on peut appliquer une simplification de Quine à n'importe quelle sous-formule de $\mathcal{P}$ et obtenir une formule sémantiquement équivalente à $\mathcal{P}$.

Algorithme $Quine(\mathcal{P}$, valuation$)$

Tant que possible, Simplifier $\mathcal{P}$ en utilisant les formules de simplification.

Ensuite, si $\mathcal{P} = \top$, renvoyer *vrai* Sinon si $\mathcal{P} = \bot$, renvoyer *faux*.

Choisir une variable x_i.
Si $Quine\Big(\mathcal{P}\Big[x_i \backslash \top\Big]$, valuation avec x_i vraie$\Big)$ est *vrai*, renvoyer *vrai*
Sinon renvoyer $Quine\Big(\mathcal{P}\Big[x_i \backslash \bot\Big]$, valuation avec x_i faux$\Big)$.

Propriété

L'algorithme de Quine termine.

Un variant de la boucle "Tant que" est le nombre de constructeurs dans $\mathcal{P}$

Un variant pour les appels récursifs est le nombre de variables pas encore affectées.

Propriété

L'algorithme de Quine est correcte.

$Quine(\mathcal{P}$, valuation vide$)$ renvoit vrai si, et seulement si $\mathcal{P}$ est satisfiable.

Preuve

On note $Var(Q)$ l'ensemble des variables présentes dans Q.
On procède par induction structurelle sur la structure des appels récursifs.

Supposons que l'appel sur $\mathcal{P}$ se termine dans le premier cas de base. On note $\mathcal{P}'$ la formule obtenue après la simplification de $\mathcal{P}$, donc $\mathcal{P}' = \top$

Or on peut montrer par récurrence sur le nombre d'itérations de la boucle "Tant que" que $\mathcal{P} \equiv \mathcal{P}'$.

Donc $\mathcal{P} \equiv \top$, $\mathcal{P}$ est une tautologie, donc satisfiable et donc *vrai* est la bonne réponse.

Pour l'autre cas de base, c'est la même chose avec $\mathcal{P} \equiv \mathcal{P}' = \bot$. $\mathcal{P}$ est une antilogie, donc n'est pas satisfiable et donc *faux* est la bonne réponse.

On note $\mathcal{P}'$ la formule obtenue après simplification $\mathcal{P}'$ nest ni $\top$ ni $\bot$, donc elle contient une variable x.

On suppose l'hypothèse d'induction: $Quine(\mathcal{P}'[x\backslash\top])$ et $Quine(\mathcal{P}'\backslash\bot)$ envoient la bonne réponse.

Si $Quine(\mathcal{P}'[x\backslash\top])$ est vrai, alors il existe une valuation sur $Var(P'[x\backslash\top])$ notée v telle que $[\![\mathcal{P}'[x\backslash\top]]\!]_v = V$.

On considère

$$\begin{aligned} v' : \quad & x \mapsto V \\ & x' \neq x \mapsto v(x') \end{aligned}$$

Alors $[\![\mathcal{P}']\!]_v = V$.
Donc $\mathcal{P}'$ est satisfiable donc $\mathcal{P}$ aussi et l'algorithme renvoit *vrai* qui est la bonne réponse.

Sinon $Quine(\mathcal{P}'[x\backslash\bot])$

Si c'est vrai, on refait la même preuve.

Si c'est faux, donc ni $\mathcal{P}'[x\backslash\top]$ ni $\mathcal{P}'[x\backslash\bot]$ ne sont satisfiables, ce qui signifie que peu importe la valeur choisie par x, $\mathcal{P}'$ n'est pas satisfiable et $\mathcal{P}$ aussi. L'algorithme renvoie *faux* et c'est la bonne réponse.

Logique du premier ordre

Motivation

On veut rajouter les quantificateurs $\forall$ et $\exists$ à notre logique. Pour pouvoir exprimer des propriétés de la forme:

$$\forall A \in \mathbb{R}_+^*,\ \exists n_0 \in \mathbb{N},\ \forall n \in \mathbb{N},\quad n \geq n_0 \implies u_n \geq A$$

Il faut pouvoir définir:

- les ensembles dans lesquels les variables sont prises
- les différentes fonctions qu'on leur applique

On ne définit que la syntaxe, la sémantique est hors-programme.

Termes

En logique du premier ordre, on manipule des variables, des termes et des prédicats.

Définition

Soit X un ensemble infini-dénombrable de symboles de variables et soit $\mathcal{S}_f$ un ensemble de symboles de fonctions, chacun associé à une arité.

On note $\mathcal{S}_f^k$ les symboles de fonctions d'arité $k \in \mathbb{N}$
En particulier $\mathcal{S}_f^0$, dont on peut appeler les élements des constantes.

La signature $(X, \mathcal{S}_f)$ définit un ensemble inductif:

- La base est $\mathcal{B} = X \cup \mathcal{S}_f^0$
- Pour tous $t_1, ..., t_k$ des termes et $f \in \mathcal{S}_f^k$ un symbole de fonction, $f(t1, ..., t_k)$ est un terme.

Remarque

L'idée est que:

- Les variables prendront leur valeur dans un domaine.
- Les symboles de fonctions seront associésà un graphe allant du domaine dans le domaine.

Prédicats

Définition

On considère un ensemble $\mathcal{S}_p$ de symboles de prédicat, chacun associé à une arité (on peut définir $\mathcal{S}_p^k$).

Les élements de $\mathcal{S}_p^0$ sont appelés des propositions.

L'idée est que les prédicats sont des fonctions qui envoient les termes sur $\mathbb{B}$.

Exemple

$$\forall i \quad (leq(0,i) \land leq(i,3)) \to even(pos(a,i))$$

Définition

Une *formule atomique* sur $(X, \mathcal{S}_f, \mathcal{S}_p)$ est une expression de la forme

$$\mathcal{P}(t_1, ..., t_n)$$

avec $\mathcal{P} \in \mathcal{S}_p$ et $t_1, ..., t_n$ des termes.

Définition

L'ensemble des formules du premier ordre sur $(X, \mathcal{S}_f, \mathcal{S}_p)$ est défini par induction avec:

- Les élements de bases sont les formules atomiques
- Les constructeurs sont:
  - $\varphi \mapsto \neg\varphi$
  - $\varphi \mapsto \psi \mapsto \varphi \land \psi$
  - $\varphi \mapsto \psi \mapsto \varphi \lor \psi$
  - $\varphi \mapsto \psi \mapsto \varphi \implies \psi$
  - $\varphi \mapsto \psi \mapsto \varphi \iff \psi$

Si x est une variable de X alors on a également deux constructeurs

- $\varphi \mapsto \exists x.\ \varphi$
- $\varphi \mapsto \forall x.\ \varphi$

Exemple

Si on veut raisonner sur des entiers:

$$\begin{aligned} X :=& \{n, m, i, j\} \\ \mathcal{S}_f^0 :=& \{Zéro\} \\ \mathcal{S}_f^{\neq 0} :=& \{Successeur, \oplus, \otimes\} \\ \mathcal{S}_p :=& \{>, <, \geq, \leq, \stackrel{?}{=}\} \end{aligned}$$

Variables libres et liées

Définition

Une variable $x \in X$ qui apparait juste après un constructeur $\forall$ ou $\exists$ est une variable *liée*.

Les autres sont des variables *libres*

Exemple

Dans $\forall x \ \forall y \quad p(x, f(y))$, x et y sont des variables liées.

Dans $\exists x \quad q(g(x), g(y))$, x est liée et y est libre.

Dans

$$\Big(\forall x \ \forall y \quad p(x, f(y))\Big) \wedge \Big(\exists x \quad q(g(x), g(y)\Big)$$

on n'a pas les mêmes variables à gauche et à droite du $\wedge$. Ca ne change pas pour autant (en renommant les variables).

Définition

Soit x une variable intruduite par $\exists x.\varphi$ ou $\forall x.\varphi$ alors sa porte est limitée à φ.

Les variables libres sont globales.

Substitution

Il est intérressant d'étudier comment substituer un variable par un terme.

Attention On ne substitue pas les variables liées si le quantificateur est encore là.

Définition

On définit la substitution d'une variable libre par induction sur la structure des formules:

- $p(t_1, ..., t_n)^{\{x \leftarrow t\}} = p\Big(t_1^{\{x \leftarrow t\}}, ..., t_n^{\{x \leftarrow t\}}\Big)$
- $(\neg\varphi)^{\{x \leftarrow t\}} = \neg\varphi^{\{x \leftarrow t\}}$
- $(\varphi \wedge \psi)^{\{x \leftarrow t\}} = \varphi^{\{x \leftarrow t\}} \wedge \psi^{\{x \leftarrow t\}}$
- $(\varphi \vee \psi)^{\{x \leftarrow t\}} = \varphi^{\{x \leftarrow t\}} \vee \psi^{\{x \leftarrow t\}}$
- $(\varphi \implies \psi)^{\{x \leftarrow t\}} = \varphi^{\{x \leftarrow t\}} \implies \psi^{\{x \leftarrow t\}}$
- $(\varphi \iff \psi)^{\{x \leftarrow t\}} = \varphi^{\{x \leftarrow t\}} \iff \psi^{\{x \leftarrow t\}}$

Si $y \neq x$

- $(\exists y.\varphi)^{\{x \leftarrow t\}} = \exists y\Big(\varphi^{\{x \leftarrow t\}}\Big)$
- $(\forall y.\varphi)^{\{x \leftarrow t\}} = \exists y\Big(\varphi^{\{x \leftarrow t\}}\Big)$

Remarque

Une formule $\forall x \quad \varphi$ sera vraie en gros si pour tout terme t, $\varphi^{\{x \leftarrow t\}}$ est vraie.

Une formule $\exists x \quad \varphi$ sera vraie en gros si il existe un terme t_0 tel que $\varphi^{\{x \leftarrow t_0\}}$ est vraie.

Exo

$\varphi = \forall x\ \forall y\ \exists z \quad \Big(\neg f(x,y) \vee g(a,x) \wedge g(p(a,x),z)\Big)$

On a:
- $X = \{x, y, z, a\}$
- $\mathcal{S}_p = \{f, g\}$
- $\mathcal{S}_f = \{p\}$

Formules atomiques: $f(x,y)$, $g(a,x)$, $g(p(a,x),z)$

Termes: $p(a,x)$, x, y, z, a

Variables libres: a

Variables liées: x, y, z

Fin du chapitre