- LOGIQUE -

10.09.2010

① Bibliographie :

Systèmes formels — Claude BENZAKEN
Série logique mathématique — Ed MASSON

Outils logiques pour l'IA — J.P. Delahaye
Ed Eyrolles

L'Art de Prolog — SHAPIRO - STERLING
Mit Press

- http://fr.wikipedia.org/wiki/Portail:logique

Ce semestre → 3 langages :

→ langage des propositions ②
→ langage des prédicats (utilisé en math) ③
→ Prolog (le seul qui permet de programmer)

② Le langage des propositions

* c'est une première couche (couche d'accroche)

- but :
  - étude de la modélisation de raisonnement
  - analyse d'argumentations simples
  - utilisation des systèmes formels de démonstration

2.1. La syntaxe (ce que trouve-t-on dans ce langag

La proposition est la construction de base du langag

Comme en français pour qu'un texte ait un sens il faut que les propositions soient connectés. Il faut des liens entre les propositions élémentaires et les propositions atomiques

Toute proposition est bâtie à partir des propositions élémentaires reliées par des connecteurs.

2.1.1. Les connecteurs usuels

On utilisera :

- → la négation
- → la conjonction
- → la disjonction
- → l'implication (l'équivalence)

idée de	usage	notation
négation	non, ne... pas... ne ... jamais	$\neg$
conjonction	et, mais, quoique	$\wedge$
disjonction	ou, ou bien, ou alors...	$\vee$
implication	si... alors...	$\Rightarrow$

Remarque :

Dans la logique vue ici, le temps n'est pas pris en compte. (i.e. le pouvoir d'expression du langage des propositions est largement moindre que celui du français)

Le *et* parfois peut signifier "puis"

! La logique temporelle n'est pas vue ici.

* L'équivalence : implication dans les 2 sens $\Leftrightarrow$

2.1.2. Propositions élémentaires (atomiques)

Ce sont des énoncés (des affirmations) sans connecteur.

Ces affirmations peuvent se révéler vrai ou fausses.

Remarques :

Dans ce cours on ne peut pas modéliser des affirmation vraies à 20%, + on travaille dans une logique à 2 valeurs ($\{v, f\}$), ($\{1, 0\}$)

Les logiques floues permettent de considérer une infinité de valeurs : $[0,1]$.

2.1.3 Grammaire des propositions

<prop> ::= <prop_atom> |
(<prop_comp>)
<prop_comp> ::= ¬ <prop> |
<prop> <connect_bin> <prop>
<connect_bin> ::= ⇒ | ∧ | ∨
<prop_atom> ::= ident en minuscule

Remarques :

* Toute proposition est construite à partir d'un nombre fini de prop_atom (si manque d'imagination on utilise un suffixe pour des idents).

i_n $\quad n \in \mathbb{N}$

* Le langage défini par la grammaire est noté $\mathcal{P}$.
Il arrivera que l'on réduise le nombre de connecteurs. Dans ce cas on mettra en indice les connecteurs retenus

$\mathcal{P}(\neg, \wedge)$

* On pourra utiliser des idents en majuscule pour noter une proposition quelconque.

Exemple :

$$\left.\begin{array}{l} a \\ p1 \\ (\neg a) \\ ((\neg a) \Rightarrow (b \vee (c \wedge d))) \end{array}\right\} \in \mathcal{P}$$

$A \Rightarrow B \quad \subset \mathcal{P}$

On "assouplit" la grammaire :

→ priorité décroissante des connecteurs

$\neg$

$\wedge$

$\vee$

$\Rightarrow$

14/09/2010

2.2 La sémantique de $\mathcal{P}$ (ou comment calculer avec des propositions)

idée : Propositions sont vues comme des expressions logiques

Connecteurs sont vus comme des opérateurs.

but : déterminer la valeur de toute $p \in \mathcal{P}$

2.2.1 Valuation

Définition

Une valuation est une application de $\mathcal{A}$ dans $\{1,0\}$

$\mathcal{A} \subset \mathcal{P}$ ensemble des propositions atomiques

Pour connaître la valeur de $P \in \mathcal{P}$ il faut construire une extension de val à $\mathcal{P}$ tout entier

$$\boxed{\begin{array}{l} v : \mathcal{P} \to \{0,1\} \\ \quad P \mapsto 0 \text{ ou } 1 \end{array}}$$

Construction de x:

Formellement cette construction se fait par récurrence sur la longueur de P.

$P \in \mathcal{P}$

3 cas:

1) $P \in \mathcal{A}$ dans ce cas on prend $v \equiv val$

$v(P) = val(P)$

2) P s'écrit $\neg Q$

avec $Q \in \mathcal{P}$ et Q plus courte que P

$v(P) = 0$ si $v(Q) = 1$

$v(P) = 1$ si $v(Q) = 0$

3) P s'écrit Q < connect_bin > R

avec $Q \in \mathcal{P}$, $R \in \mathcal{P}$ et Q, R plus courtes que P

$v(Q)$	$v(R)$	$v(P = Q \wedge R)$	$v(P = Q \vee R)$	$v(P = Q \Rightarrow R)$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	1

Remarque: L'implication pose souvent problème.

- Faut-il préciser une valeur pour $Q \Rightarrow R$ lorsque $v(Q) = 0$?

oui : $\Rightarrow$ est vue comme une opération, cela entraîne que $\Rightarrow$ doit être définie sur $\{0,1\} \times \{0,1\}$ tout entier

2.2.2. Table de vérité

Soit $P \in \mathcal{P}$ comportant n propositions atomiques. Il y a alors 2^n valuations différentes possibles pour P. Chacune de ces valuations amène à calculer une valeur de vérité pour P.

Définition : On appelle **table de vérité** d'une $P \in \mathcal{P}$ contenant n propositions atomiques, une table
- de $n+1$ colonnes (1 col. pour prop. atom. + 1 col. pour P : colonnes principales)
- de 2^n lignes (1 par valuation)

Techniquement on "ouvre" des colonnes intermédiaires pour les sous propositions de P.

2.2.3 Définitions

Définition : Une **réalisation** de $P \in \mathcal{P}$ est une valuation v telle que $v(P) = 1$

Définition : Une **réfutation** de $P \in \mathcal{P}$ est une valuation v telle que $v(P) = 0$

Définition : $P \in \mathcal{P}$ est **satisfaisable** si elle possède au moins une réalisation

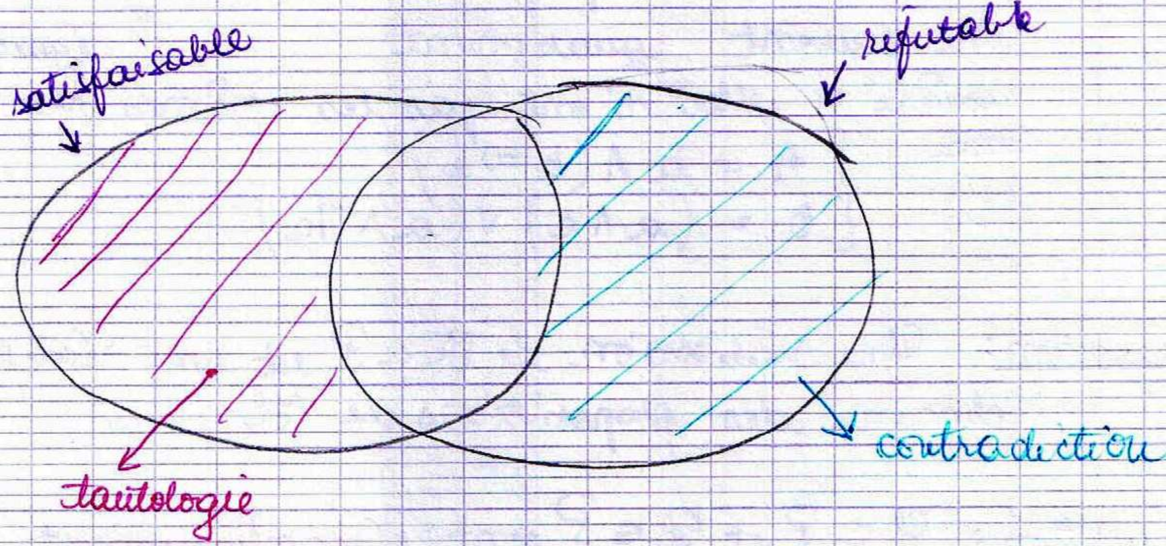
Définition : $P \in \mathcal{P}$ est **réfutable** si elle possède au moins une réfutation.

Remarque : Une même proposition peut être à la fois satisfaisable et réfutable.
!

Définition : Une **tautologie** est une $P \in \mathcal{P}$ non réfutable.
i.e. si P est une tautologie :

- toute valuation est réalisation
- $\forall v,\ v(P)=1$
- la colonne principale de P ne contient pas des 1

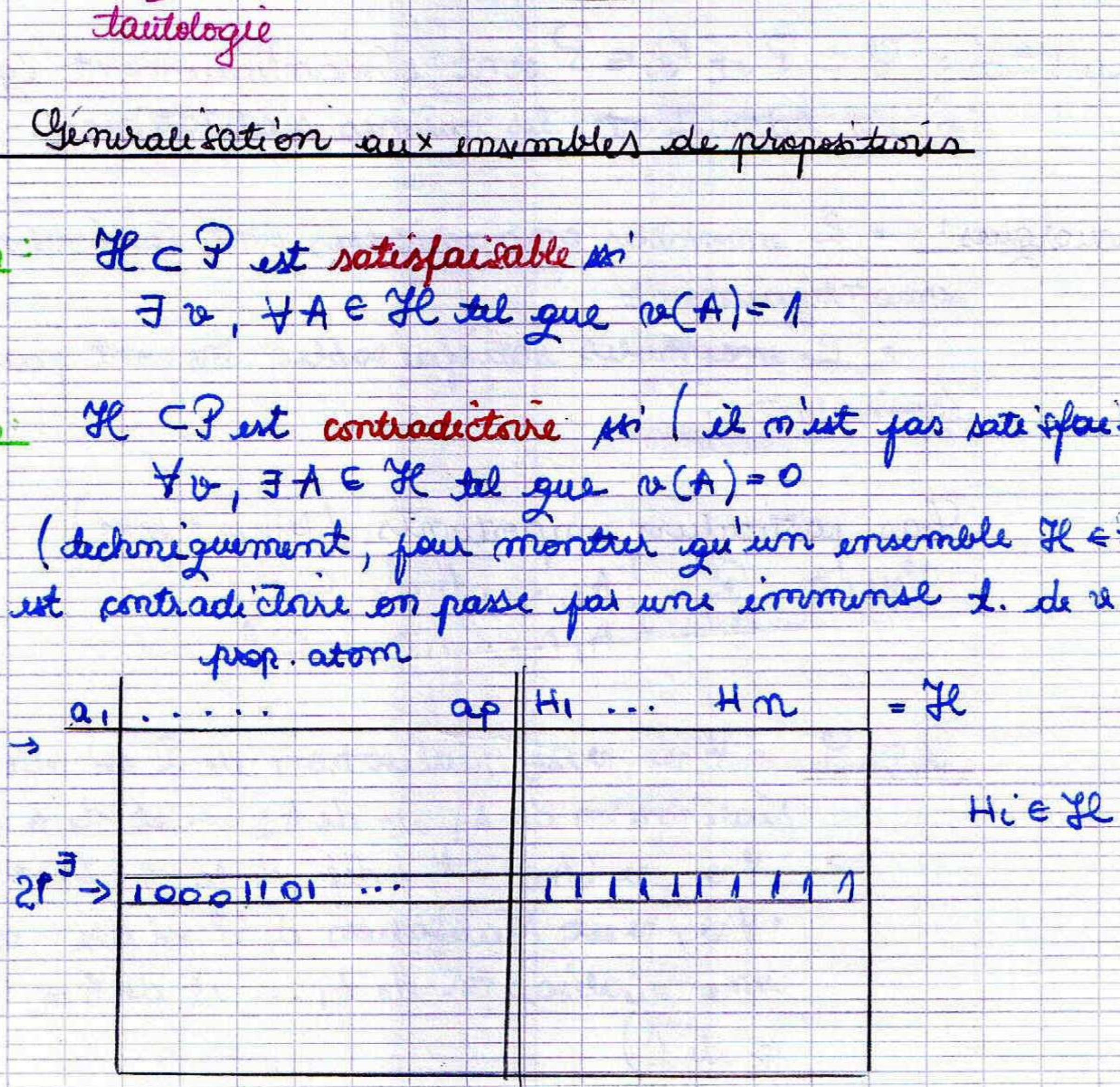
Définition : Une **contradiction** est une $P \in \mathcal{P}$ non satisfaisable



Généralisation aux ensembles de propositions

Définition : $\mathcal{H} \subset \mathcal{P}$ est **satisfaisable** ssi
$\exists v,\ \forall A \in \mathcal{H}$ tel que $v(A)=1$

Définition : $\mathcal{H} \subset \mathcal{P}$ est **contradictoire** ssi (il n'est pas satisfaisable)
$\forall v,\ \exists A \in \mathcal{H}$ tel que $v(A)=0$
(techniquement, pour montrer qu'un ensemble $\mathcal{H} \in \mathcal{P}$ est contradictoire on passe par une immense t. de v.)



Définition : équivalence sémantique

$A \in \mathcal{P}$, $B \in \mathcal{P}$ sont sémantiquement équivalents ssi $\forall v, v(A) = v(B)$

On note $\boxed{A \approx B}$

Remarque • 2 propositions atomiques ne sont jamais équivalents. Elles prennent leur valeur de manière indépendante.

• Par contre, 2 propositions A et B quelconques peuvent être souvent quasiment équivalentes même si elles n'ont pas les mêmes prop. atomiques.

$$\begin{cases} A = a \wedge (b \Rightarrow b) \\ B = (a \wedge c) \vee (a \wedge \neg c) \end{cases}$$

Définition : Une réalisation de $\mathcal{H} \subset \mathcal{P}$ est une réalisation de chacune des propositions de $\mathcal{H}$.

Définition : $\mathcal{S} \subset \mathcal{P}$ et $\mathcal{R} \in \mathcal{P}$ sont sémantiquement équivalents si ils admettent les mêmes réalisations

Remarques : • 2 ensembles contradictoires sont équivalents sémantiquement.

• 2 ensembles satisfaisables ne sont pas forcément équivalents.

Cas particulier important (repris + tard)

Soient $\mathcal{R} = \{A_1, \dots, A_m\} \subset \mathcal{P}$

$\mathcal{S} = \{A_1 \wedge \dots \wedge A_m\} \subset \mathcal{P}$

$\mathcal{R} \approx \mathcal{S}$ • $\forall v$, v est réalisation de $\mathcal{R}$ ssi elle est réalisation de A_1 et de A_2 ... et de A_m.

$\forall v, v(A_i) = 1$ (déf. de la réalisation)

• $\forall v$, v est réalisation de $\mathcal{S}$ ssi elle est une réalisation de A_1 ... et de A_m (cf + de v de $\wedge$)

donc $\mathcal{R}$ et $\mathcal{S}$ sont les mêmes réalisations
i.e.
$\mathcal{R} \approx \mathcal{S}$ (déf de l'équivalence sémantique étendue aux ensembles)

2.2.4. Équivalences remarquables

(souvent utilisées comme règles de réécriture)
chgt de syntaxe sans chgt de sémantique

dans P on ajoute 2 constantes
<u>vrai</u> telle que $\forall v,\ v(\text{vrai}) = 1$
<u>faux</u> telle que $\forall v,\ v(\text{faux}) = 0$

* idempotence
$A \vee A \approx A$ (1)
$A \wedge A \approx A$ (2)

* double négation
$\neg(\neg A) \approx A$ (3)

* commutativité
$A \vee B \approx B \vee A$ (4)
$A \wedge B \approx B \wedge A$ (5)

* associativité
$(A \vee B) \vee C \approx A \vee (B \vee C)$ (6)
$(A \wedge B) \wedge C \approx A \wedge (B \wedge C)$ (7)

* distributivité
$A \vee (B \wedge C) \approx (A \vee B) \wedge (A \vee C)$ (8)
$A \wedge (B \vee C) \approx (A \wedge B) \vee (A \wedge C)$ (9)

* absorption
$A \vee (A \wedge B) \approx A$ (10)
$A \wedge (A \vee B) \approx A$ (11)

* de Morgan
$\neg(A \wedge B) \approx \neg A \vee \neg B$ (12)
$\neg(A \vee B) \approx \neg A \wedge \neg B$ (13)

$A \vee vrai \approx vrai$ (14)
$A \wedge vrai \approx A$ (15)

$A \vee \neg A \approx vrai$ (16)
$A \wedge \neg A \approx faux$ (17)

! $A \Rightarrow B \approx \neg A \vee B$ (18)
$(A \Leftrightarrow B) \approx (A \Rightarrow B) \wedge (B \Rightarrow A)$ (19)

Remarques : • Une tautologie est sémantiquement équivalente à vrai.
• Une même réalité peut prendre des formes syntaxiques différentes.
• Le choix des connecteurs est luxueux.
! En particulier, $\mathcal{P}(\neg, \vee, \wedge)$ possède le même pouvoir d'expression que $\mathcal{P}$. (cf équivalence (18))

Définition : Principe de dualité

La liste des équivalences remarquables fait ressortir des réécritures "symétriques" pour $\wedge$ et $\vee$. Cela conduit au principe de dualité :

Soient A et $B \in \mathcal{P}(\neg, \vee, \wedge)$

Soit $A' \in \mathcal{P}(\neg, \vee, \wedge)$ réécrite à partir de A en inversant les $\wedge$ et les $\vee$.

Soit $B' \in \mathcal{P}(\neg, \vee, \wedge)$ — " — B — — " —.

Si $A \approx B$ alors $A' \approx B'$

2.2.6 Formes normales

idée : caractériser de parties de $\mathcal{P}$
- les critère pour déterminer les parties est le choix des connecteurs

"Proposition"

$\forall A \in \mathcal{P}$

$\exists A_1 \in \mathcal{P}(\vee, \neg)$ telle que $A_1 \approx A$

$\exists A_2 \in \mathcal{P}(\wedge, \neg)$ telle que $A_2 \approx A$

$\exists A_3 \in \mathcal{P}(\Rightarrow, \neg)$ telle que $A_3 \approx A$

Cette "proposition" indique qu'il n'y a pas de perte de pouvoir d'expression si on ne conserve qu'un connecteur binaire $+ \neg$.

Construction de A_1

$x \Rightarrow y \approx \neg x \vee y \in \mathcal{P}(\neg, \vee)$

$x \wedge y \approx \neg\neg(x \wedge y) \approx \neg(\neg x \vee \neg y) \in \mathcal{P}(\neg, \vee)$

Construction de A_2

$x \Rightarrow y \approx \neg x \vee y \approx \neg\neg(\neg x \vee y) \approx \neg(x \wedge \neg y) \in \mathcal{P}(\neg, \wedge)$

$x \vee y \approx \neg\neg(x \vee y) \approx \neg(\neg x \wedge \neg y) \in \mathcal{P}(\neg, \wedge)$

Construction de A_3

$x \vee y \approx \neg\neg x \vee y \approx \neg x \Rightarrow y \in \mathcal{P}(\neg, \Rightarrow)$

$x \wedge y \approx \neg\neg(x \wedge y) \approx \neg(\neg x \vee \neg y) \approx \neg(x \Rightarrow \neg y) \in \mathcal{P}(\neg, \Rightarrow)$

Définition : Une clause est une proposition particulière qui ne comporte que les connecteurs $\neg$ et $\vee$

⚠ ce n'est pas $\mathcal{P}(\neg, \vee)$

Grammaire :

< clause > ::= < littéral > | < littéral > $\vee$ < clause >

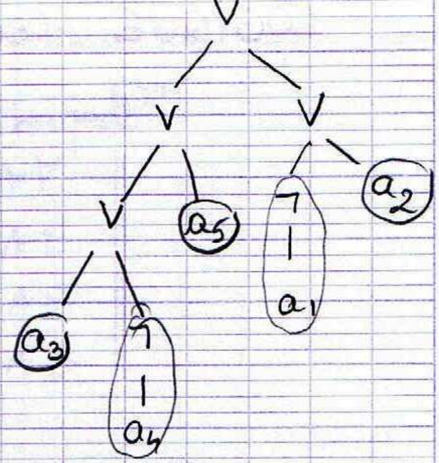
< littéral > ::= < propos. atom. > | $\neg$ < prop. atom >

Exemple :

$\neg(A \vee B) \in \mathcal{P}(\neg, \vee)$

$\neg(A \vee B) \notin \mathcal{C}$

$\mathcal{C} \subset \mathcal{P}(\neg, \vee) \subset \mathcal{P}$



"Proposition" (suite)

$\forall A \in \mathcal{P}$

$\exists A_4 \in \mathcal{P}(\vee, \wedge, \neg)$ telle que $A_4 \simeq A$

et A_4 de la forme $C_1 \wedge \ldots \wedge C_m$, $m \geq 1$

$C_i \in \mathcal{C}$

La forme A_4 est dite forme normale conjonctive (FNC) ou forme clausale

A_4 : conjonction de disjonction de littéraux

Construction de A_4

* 2 démarches

① syntaxique (par réécritures utilisant les équivalences remarquables)

(objet du TP8 Prolog)

② sémantique (par le calcul (table de vérité))

↳ $A \rightarrow$ t. de v. de $A \rightarrow A_4$

Supposons que A comporte n prop. atomiques

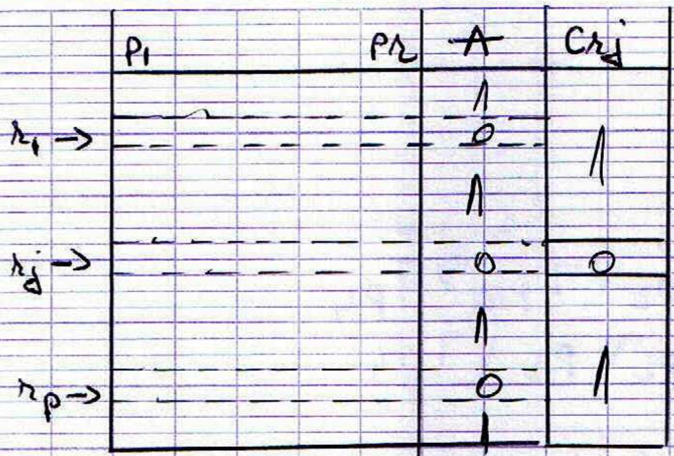
Supposons que la table de vérité de A montre p réfutations $(r_1 \ldots r_p)$

+ soit r_j une des p réfutations (i.e. $r_j(A) = 0$).

On construit une clause C_{r_j} de n littéraux

$C_{r_j} = l_1 \vee \ldots \vee l_n$

Q] Comment choisir les littéraux l_i $i \in [1, n]$ de sorte que cette valuation r_j (réfutation) soit aussi réfutation de C_{r_j} et que toute autre valuation soit réalisation de C_{r_j} ?



R] On choisit

$l_i = p_i$ si $r_j(p_i) = 0$

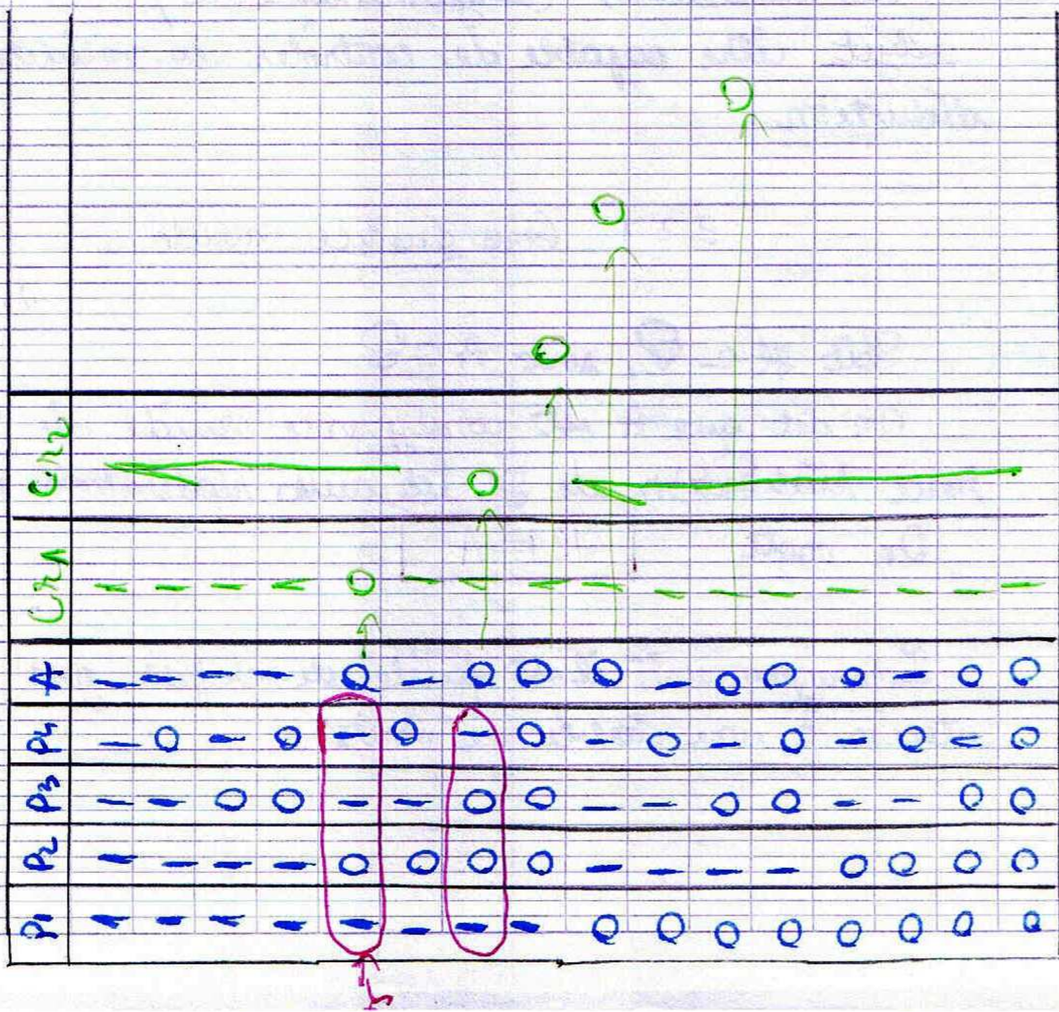
$l_i = \neg p_i$ si $r_j(p_i) = 1$

$$\boxed{A_4 = \bigwedge_{j=1}^{p} C_{r_j}}$$

A_4 cumule toutes les réfutations de A.

<u>Application</u>

↳ sur la table de vérité de l'exo 8.



$v_5 = r1$

$p_1 \to 1$

$p_2 \to 0$

$p_3 \to 1$

$p_4 \to 1$

$Cr_1 = \neg p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4$

$Cr_2 = \neg p_1 \vee p_2 \vee p_3 \vee \neg p_4$

17/06/2010

2.2.7 Théorème de compacité (non démontré ici)

Soit $\mathcal{H} \subset \mathcal{P}$

Si toute partie finie de $\mathcal{H}$ est satisfaisable, alors $\mathcal{H}$ est satisfaisable. (résultat intéressant en théorie, mais sans trop d'importance dans ce cours)

2.3. Sémantique de la déduction

Maintenant les propositions deviennent des éléments "d'objets" plus importants.

- les déductions (argumentations)

Objet: être capable de contrôler la validité d'une déduction.

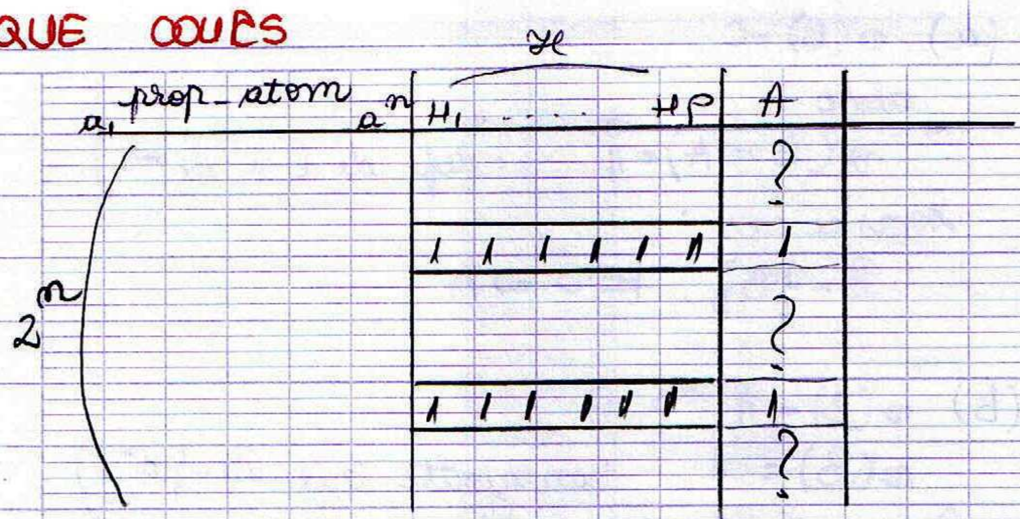
2.3.1 Conséquence valide

Définition: Soit $\mathcal{H} \subset \mathcal{P}$, soit $A \in \mathcal{P}$.

On dit que A est conséquence valide de $\mathcal{H}$ quand toute réalisation de $\mathcal{H}$ est aussi réalisation de A.

On note $\boxed{\mathcal{H} \models A}$

Techniquement il est facile de vérifier que $\mathcal{H} \models A$ en utilisant une table de vérité.



Corollaire : Les tautologies sont conséquences valides de tout ensemble de propositions. (def de $\models$ et de tautologie)

Cas limite de la définition de conséquence valide

$\mathcal{H} = \emptyset$ $\quad (\emptyset \models A)$

Dans ce cas toute valuation est réalisation de $\mathcal{H}$

$\forall v, \forall H \in \mathcal{H}, v(H) = 1$

donc $v(A) = 1$ d'après def de consig. valide.

En fait on note

$\boxed{\models A}$ (on prononce A est une tautologie)

2.3.2. Théorème sémantique de la déduction

Soit $B \in \mathcal{B}$ $\quad \mathcal{B} \subset \mathcal{P}$

Soit $A \in \mathcal{P}$

$\mathcal{B} \models A$ ssi $\mathcal{B} - \{B\} \models B \Rightarrow A$

Dém :

sens $\rightarrow$

Supposons $\mathcal{B} \models A$, (i.e. $\forall v$, v réalisation de $\mathcal{B}$ impose $v(A) = 1$)

Soit v une réalisation quelconque de $\mathcal{B} - \{B\}$

Il faut montrer que $v(B \Rightarrow A) = 1$ pour avoir $\mathcal{B} - \{B\} \models B \Rightarrow A$

(a) $v(\mathcal{B})=0$

alors

$v(B \Rightarrow A)=1$ (def de t.v. de $\Rightarrow$)

Dans ce cas :

$\mathcal{B}-\{B\} \models B \Rightarrow A$

(b) $v(B)=1$

$v(\mathcal{B})=1$ (on ajoute B à $\mathcal{B}-\{B\}$)

Donc $v(A)=1$ d'après def. de c.v.

Donc $v(B \Rightarrow A)=1$ d'après t de v de $\Rightarrow$)

Dans ce cas aussi on a

$\mathcal{B}-\{B\} \models B \Rightarrow A$

sens $\Leftarrow$

On suppose que $\mathcal{B}-\{B\} \models B \Rightarrow A$

(i.e. $v(\mathcal{B}-\{B\})=1$ impose $v(B \Rightarrow A)=1$ - def de c.v)

Soit v une réalisation quelconque de $\mathcal{B}$, il faut montrer que c'est aussi une réalisation de A.

$v(\mathcal{B})=1$

donc $v(\mathcal{B}-\{B\})=1$

donc $\boxed{\begin{array}{l} v(B \Rightarrow A)=1 \\ v(B)=1 \end{array}}$ def de c.v. / car $B \in \mathcal{B}$

donc $v(A)=1$ def de $\Rightarrow$

Donc une réalisation quelconque de $\mathcal{B}$ est réalisation de A

$\mathcal{B} \models A$ (def $\mathcal{B}$)

$B \Rightarrow A = 1$

	B	A	
$B=0$	0	0	1
	0	1	1
	1	0	0
$B=1$ $\rightarrow$	1	1	1 $\leftarrow$

2 corollaires importants :

Corollaire 1 : On peut ramener le contrôle de la validité d'une argumentation à la détermination d'une tautologie.

$$H_1, H_2, \dots, H_n \models A$$

ssi

$$H_1, H_2, \dots, H_{n-1} \models H_n \Rightarrow A$$

ssi

$$\vdots$$

$$\models (H_1 \Rightarrow (H_2 \Rightarrow \dots \Rightarrow (H_n \Rightarrow A)))$$

Application n fois du théorème sémantique de la déduction

Corollaire 2 : (réduction à l'absurde) Que dire de $\mathcal{H}$ quand $\mathcal{H} \models$ faux ?

$\mathcal{H}$ est contradictoire (aucune valuation ne satisfait en même temps toutes les propositions de $\mathcal{H}$)

Considérons $\mathcal{H} = \mathcal{B} \cup \{\neg A\}$ avec

$\mathcal{B} \cup \{\neg A\} \models$ faux ($\mathcal{B} \cup \{\neg A\}$ est contradictoire)

TSD

ssi $\mathcal{B} \models \neg A \Rightarrow$ faux

ssi $\mathcal{B} \models A \vee$ faux

ssi $\mathcal{B} \models A$

Pour montrer $\mathcal{B} \models A$

On peut montrer que $\mathcal{B} \cup \{\neg A\}$ est contradictoire

2.3.3- Théorème de finitude

Soit $\mathcal{B}$ ensemble infini de propositions

$\mathcal{B} \models A$ ssi $\exists \mathcal{F}$ partie finie de $\mathcal{B}$ telle que $\mathcal{F} \models A$

Le fait que A soit conséq. valide de $\mathcal{B}$ n'est pas dû à son caractère infini.

2.4 Systèmes formels

Rappel:

+ le niveau de propositions
  - syntaxe 2.1
  - sémantique 2.2.
+ le niveau de déduction
  - sémantique 2.3.
    - c. valide
    - T. sem de la déduction
  - syntaxe 2.4

La motivation

- il n'est pas raisonnable de valider une argumentation en utilisant une table de vérité.
  * taille qui croît exponentiellement en fonction de n (nbre de prop. atomiques)
  * n'existe pas en langage prédicats.

2.4.1 Généralités

- idée de base : trouver une suite de réécriture qui partant de $\mathcal{H} \subset \mathcal{P}$ aboutissent à A quand $\mathcal{H} \models A$
- au lieu d'envisager tous les cas on cherche à produire "mécaniquement des énoncés

Pour faire cela on a un outil : le système formel

Dans ce cours on n'étudie pas formellement les S.F.

On retiendra l'idée triviale :

- un SF est une machine à produire des énoncés appartenant à un certain langage $\mathcal{L}$.
- un analyseur syntaxique est une machine à reconnaître

Définition : Un système formel est la donnée :

→ d'un langage $\mathcal{L}$ (ensemble potentiellement infini d'énoncés)

→ d'un procédé de production d'énoncés $\in \mathcal{L}$
le procédé aboutit à un schéma d'induction qui possède :
- une base $\mathcal{B} \subset \mathcal{L}$
- des règles d'inférences

Définition: Les énoncés appartenant à la base d'un schéma d'induction sont appelés **axiomes**

Remarque. Construire un système formel est facile, mais un S.F. intéressant est + délicat

Définition: Tout énoncé produit en partant de la base, en utilisant les règles d'inférences est appelé **théorème**

On note $\vdash T$
(on prononce T est un théorème)

Remarque: théorème : objet syntaxique

2.4.2 Un premier système formel pour les propositions (SF.)

-**ambition:** Construire un S.F. capable de produire des propositions de $\mathcal{P}$ et si possible :
- toutes les tautologies de $\mathcal{P}$
- que des tautologies.

SF 1:
- langage : $\mathcal{P}(\Rightarrow, \neg)$
- schéma d'induction
  * base : $A \Rightarrow (B \Rightarrow A)$ $\quad Ax_1 \in \mathcal{P}(\neg, \Rightarrow)$

$(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$ $\quad Ax_1 \in \mathcal{P}(\Rightarrow, \neg)$

$(A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$ $\quad Ax_3 \in \mathcal{P}(\Rightarrow, \neg)$

* règle d'inférence (modus ponens)
$A, A \Rightarrow B \vdash B$

<u>Remarque</u> : ① le choix du langage n'est pas pénalisant
$\forall P \in \mathcal{P} \quad \exists P_1 \in \mathcal{P}(\neg, \Rightarrow)$
$P_1 \approx P$

② A, B, C sont des propositions quelconques de $\mathcal{P}(\neg, \Rightarrow)$
<u>conség</u> : 3 infinités d'axiomes

③ la règle $A, A \Rightarrow B \vdash B$ indique que si le SF_1 a déjà produit A a déjà produit $A \Rightarrow B$ il produit B.

2.4.3. SF_1 est sain

Tout théorème produit par SF_1 est une tautologie $\forall A \in \mathcal{P}(\neg, \Rightarrow)$ si $\vdash A$ alors $\models A$

(si A est un théorème, alors A est une tautologie)

<u>- idée de preuve</u> : cette propriété de SF se démontre par récurrence.

- cas de base :
  * les axiomes sont des tautologies (se montre avec t. de v.)
- cas général :
  * à partir de 2 tautologies précédemment produites, notre règle d'inférence ne peut produire qu'une tautologie
    • supposons | A et
    | $A \Rightarrow B$

2 tautologies déjà produites
qu'en est-il de B ?

A	B	A => B
0	0	1
0	1	1
1	0	0
1	1	1

d'après t. de v. de => v(B) = 1

2.4.4 Démonstration

sémantique	syntaxique
tautologie	théorème
c.v ⟷	démonstration
$\models$	$\vdash$

Définition : Soit $\mathcal{H} \subset \mathcal{P}(\neg, \Rightarrow)$

Soit $A \in \mathcal{P}(\neg, \Rightarrow)$

On dit que A se démontre à partir de $\mathcal{H}$ quand il existe une suite finie $\mathcal{D}$ de propositions :

$\mathcal{D}$ P_1 P_2 ... $P_n = A$

telle que $\forall i \in [1, n]$

1. soit P_i est une axiome
2. soit $P_i \in \mathcal{H}$
3. soit $\exists k < i, \exists j < i$ tels que $P_k, P_j \vdash P_i$

(i.e. P_i est obtenue par modus ponens à partir de 2 propositions démontrées avant dans la suite finie $\mathcal{D}$)

On note $\mathcal{H} \vdash A$, on prononce "A se démontre à partir de $\mathcal{H}$".

$\mathcal{D}$ est une démonstration.

Cas limite important :

$\mathcal{H} = \emptyset$

On note $\vdash A$ (A se démontre à partir d'aucune hypothèse)

A est une théorème

Exo : (Exercice 11)

Rédiger la démonstration du théorème $p \Rightarrow p$ dans SF1 i.e. $\vdash_{SF_1} (p \Rightarrow p)$

$$\mathcal{D} \quad \begin{array}{|l} P_1 \\ P_2 \\ \hline P_3 \\ P_4 \\ P_5 = p \Rightarrow p \end{array}$$

$A \Rightarrow B$

$$P_1 = \underbrace{(\overset{A}{p} \Rightarrow (\overbrace{(p \Rightarrow p) \Rightarrow p}^{B}))}_{} \Rightarrow ((\overset{A}{p} \Rightarrow \overbrace{(p \Rightarrow p)}^{B}) \Rightarrow (\underset{A}{p} \Rightarrow \underset{C}{p})) \quad Ax_2$$

A

$$P_2 = (\underset{A}{p} \Rightarrow (\underbrace{(p \Rightarrow p)}_{B} \Rightarrow \underset{A}{p})) \quad Ax\,1$$

$$P_3 = ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)) \quad \text{mp sur } P_1 \text{ et } P_2$$

$$P_4 = (p \Rightarrow (p \Rightarrow p)) \quad Ax\,1$$

$$P_5 = (p \Rightarrow p) \quad \text{m.p. sur } P_3 \text{ et } P_4$$

2.4.5 Théorème syntaxique de la déduction

$$\boxed{\begin{array}{l} \text{Soit } \mathcal{B} \subset \mathcal{P}(\neg, \Rightarrow) \\ \quad B \in \mathcal{B} \\ \quad A \in \mathcal{P}(\neg, \Rightarrow) \\ \mathcal{B} \vdash A \ \text{ ssi } \ \mathcal{B} - \{B\} \vdash B \Rightarrow A \end{array}}$$

Preuve : $\rightleftarrows$ (ssi)

(1) sens $\rightarrow$

Sachant $\mathcal{B} \vdash A$ il faut montrer $\mathcal{B} - \{B\} \vdash B \Rightarrow A$

i.e.

sachant qu' $\exists\, \mathcal{D}_{\mathcal{B}}$:

$\mathcal{D}_{\mathcal{B}}$ | P_1, …, $P_n = A$

Montrons qu'$\exists\, \mathcal{D}_{\mathcal{B}-\{B\}}$ | P'_1, P'_2, …, $P'_m = B \Rightarrow A$

Pour faire cette démo on procède par récurrence sur la longueur de $\mathcal{D}_{\mathcal{B}}$.

* $n = 1$ i.e. $\mathcal{D}_{\mathcal{B}}$ | $P_1 = A$

* 3 cas :

- A est un axiome

alors $\mathcal{D}_{\mathcal{B}-\{B\}}$
$P'_1 = A$ axiome
$P'_2 = A \Rightarrow (B \Rightarrow A)$ axiome
$P'_3 = B \Rightarrow A$ m.p. sur P'_1 et P'_2

donc $\vdash B \Rightarrow A$

A fortiori $\mathcal{B} - \{B\} \vdash B \Rightarrow A$

- $A \in \mathcal{B}$, $A \neq B$ (i.e. $A \in \mathcal{B} - \{B\}$)

alors : $\mathcal{D}_{\mathcal{B}-\{B\}}$
$P'_1 = A$ car $A \in \mathcal{B} - \{B\}$
$P'_2 = A \Rightarrow (B \Rightarrow A)$ axiome
$P'_3 = B \Rightarrow A$ m.p.

- $A \in \mathcal{B}$, $A = B$

alors : cf dans $p \Rightarrow p$

$\mathcal{D}_{\mathcal{B}-\{B\}}$ démo de l'exo

* hypo de récurrence : supposons que pour toute démo $\mathcal{D}_{\mathcal{B}}$ de A de longueur $n < N$, il existe une démo $\mathcal{D}_{\mathcal{B}-\{B\}}$ de $B \Rightarrow A$

* montrons que la propriété est encore vraie pour $n = N$

i.e. $\mathcal{D}_{\mathcal{B}}$
$$\left|\begin{array}{l} P_1 = \dots \\ \vdots \\ P_N = A \end{array}\right.$$

* <u>4 cas</u>:

- A axiome

  cf $n=1$ $\mathcal{D}_{\mathcal{B}-\{B\}}$ celle du cas $n=1$
- $A \in \mathcal{B}$, $A \neq B$

  cf $n=1$
- $A \in \mathcal{B}$, $A = B$

  cf $n=1$
- A obtenue par m.p.

  i.e. $\exists\ \mathcal{D}_{\mathcal{B}}$
$$\left|\begin{array}{l} \vdots \\ P_j = X \\ P_k = X \Rightarrow A \\ P_N = A \end{array}\right.$$

or $j < N$
$k < N$

<u>en utilisant l'hypo de récurrence</u>

$\exists\ \mathcal{D}_{\mathcal{B}-\{B\}}$
$$\left|\begin{array}{l} \\ \\ P'_m = B \Rightarrow X \end{array}\right.$$

$\exists\ \mathcal{D}_{\mathcal{B}-\{B\}}$
$$\left|\begin{array}{l} P''_1 \\ \\ P''_m = B \Rightarrow (X \Rightarrow A) \end{array}\right.$$

– la démonstration de $B \Rightarrow A$

$\mathcal{D}_{\mathcal{B}-\{B\}}$
$$\left|\begin{array}{ll} \mathcal{D}_{\mathcal{B}-\{B\}} & B \Rightarrow X \\ \mathcal{D}_{\mathcal{B}-\{B\}} & B \Rightarrow (X \Rightarrow A) \\ & (B \Rightarrow (X \Rightarrow A)) \Rightarrow ((B \Rightarrow X) \Rightarrow (B \Rightarrow A)) \quad Ax_2 \end{array}\right.$$

$(B \Rightarrow X) \Rightarrow (B \Rightarrow A)$ m.p. (de l'av. dern.)
$B \Rightarrow A$

(2) sens $\Leftarrow$

Supposons $\mathcal{B} - \{B\} \vdash B \Rightarrow A$

i.e. $\mathcal{D}_{\mathcal{B}\setminus\{B\}}$
$P'_1 =$
$\vdots$
$P'_i = B \Rightarrow A$

Montrons qu'$\exists$ $\mathcal{B} \vdash A$

i.e. $\mathcal{D}_{\mathcal{B}}$
$P'_1 =$

$P'_i = B \Rightarrow A$
$P'_{i+1} = B$ hyp.
$P'_{i+2} = A$ m.p.

Exercice: Montrer, dans SF_1, la transitivité de l'implication

Exercice 12

$$A \Rightarrow B,\ B \Rightarrow C \underset{SF_1}{\vdash} A \Rightarrow C$$

ssi $\underline{A \Rightarrow B,\ B \Rightarrow C,\ A} \underset{SF_1}{\vdash} C$ Th. syntax. déduction

$P_1 = A$	hyp
$P_2 = A \Rightarrow B$	hyp
$P_3 = B$	m.p.
$P_4 = B \Rightarrow C$	hyp
$P_5 = C$	m.p.

2.4.6. SF_1 est complet

Toute tautologie est un théorème qui peut être produit par SF_1.

i.e.

$$\forall A \in \mathcal{P}(\neg, \Rightarrow) \quad \text{si } \models A \text{ alors } \vdash A$$

SF_1 n'oublie aucune tautologie

- non démontré ici.

2.4.7 Conclusion

En associant les propriétés des 2.4.3 et 2.4.6 on énonce le résultat essentiel de SF_1.

$$\forall A \in \mathcal{P}_{(\neg, \Rightarrow)} \qquad \models A \quad \text{ssi} \quad \vdash A$$

plus fort encore

$$\forall A \in \mathcal{P}(\neg, \Rightarrow)$$

$$\forall \mathcal{H} \subset \mathcal{P}(\neg, \Rightarrow)$$

$$\mathcal{H} \models A \quad \text{ssi} \quad \mathcal{H} \vdash A$$

En utilisant les 2 théorèmes de la déduction.

SF_1 : excellent candidat pour faire de la démonstration automatique pour tout ...

2.5 La résolution

... SF_1 ne peut pas être implémenté efficacement. Ceci est dû aux 3 infinités d'axiomes ; la combinatoire est énorme pour démarrer la démo.

Pourtant l'approche syntaxique est la seule envisageable (elle nous libère des t. de v.)

Un système formel conçu J-A. Robinson est implémentable et utilisable dans le langage des prédicats. Son nom : Résolution (à la base de Prolog)

2.5.1 Langage des clauses (rappel)

Clause = disjonction de littéraux

Littéral = prop. atomique ou sa négation.

La **classe vide**, se note $\square$, elle ne possède aucune réalisation

$$\forall v, \; v(\square) = 0$$

Tout ensemble de propositions de $\mathcal{P}$ possède un ensemble de classes sémantiquement équivalent.

2.5.2. Un second système formel pour les propositions

SF_R - langage : $\mathcal{C} \subset \mathcal{P}$
- schéma d'induction
  - base : aucune axiome
  - règle d'inférence (Résolution)

$$a \vee A, \neg a \vee B \vdash A \vee B$$
$$A \in \mathcal{C}, B \in \mathcal{C}$$

Remarques: - a : prop. atom.

- $A \vee B$: classe produite par Résolution est appelé **resolvante** de $a \vee A$ et $\neg a \vee B$
- au moment de la production de la resolvante il y a une éventuelle **réduction** qui formellement devrait apparaître comme seconde règle d'inférence.

Les littéraux eventuellement en double dans $A \vee B$ ne sont conservés qu'une fois

$$(a \vee \underbrace{b \vee \neg c \vee d}_{A}, \neg a \vee \underbrace{b \vee e}_{B} \vdash b \vee \neg c \vee d \vee \cancel{b} \vee e$$

- le littéral a et son opposé $\neg a$ ne sont pas forcément en tête de classe

$$\underline{a} \vee b \vee c, b \vee \neg d \vee \underline{\neg a} \vdash c \vee \neg d \vee b$$

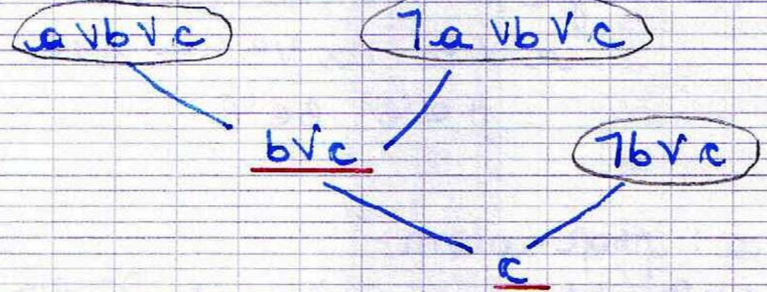
- aucune axiome !
  - génial : plus de combinatoire pour démarrer les démos
  - nul : on ne peut démontrer aucun théorème

Exercice : rediger

$$a \vee b \vee c, \neg a \vee b \vee c, \neg b \vee c \vdash_{SF_R} c$$

$$\mathcal{S}\ \left|\ \begin{array}{lll} C_1 = a \vee b \vee c & & H \\ C_2 = \neg a \vee b \vee c & & H \\ C_3 = b \vee c \vee b \vee c \text{ soit } b \vee c & & \text{résolution} \\ C_4 = \neg b \vee c & & H \\ C_5 = c \vee c \text{ soit } c & & \text{résolution} \end{array}\right.$$

Les démos avec SF_R sont toujours rédigées sous forme d'arbre.



Mais il faut revenir à la référence : le côté sémantique.
A-t-on vraiment ?

$$\neg a \vee b \vee c,\ a \vee b \vee c,\ \neg b \vee c \models c$$

On se rassure !

2.5.3 SF_R est sain

Gag ! Tout théorème est une tautologie ! (y a pas de théorème)

Il y a quand même une forme de "santé" intéressante.

$$\boxed{\text{si } \mathcal{H} \vdash_{SFR} C \text{ alors } \mathcal{H} \models C}$$

On ne peut démontrer que des conséquences valides

- idée de preuve : (par récurrence sur la longueur de la démo $\mathcal{H} \vdash C$)

① démo de lg 1 ou 2 $\Rightarrow C \in \mathcal{H}$

- dans ce cas $\underbrace{\forall v \text{ tel que } v(\mathcal{H}) = 1 \text{ impose } v(C) = 1}_{\text{déf de conséquence valide}}$ car $C \in \mathcal{H}$

② hypo. réc : toute démo de lg $< n$ démontre des conséquences valides de $\mathcal{H}$.

③ qu'en est-il des démos de $lg \geq n$?
si $\mathcal{H} \vdash C$
avec $lg = n$ et $C \notin \mathcal{H}$
C est nécessairement la résolvante de 2 clauses situées avant C dans la démo.

$\mathcal{D}$ | $P_i = a \vee A$; $P_j = \neg a \vee B$; $P_n = C = A \vee B$ — et les démos de P_i et P_j sont de $lg < n$

Donc d'après hypo. rec.
$\mathcal{H} \models P_i = a \vee A$ $\forall v$, si $v(\mathcal{H}) = 1$ alors $v(a \vee A) = 1$
$\mathcal{H} \models P_j = \neg a \vee B$ $\forall v$, si $v(\mathcal{H}) = 1$ alors $v(\neg a \vee B) = 1$
$v(A \vee B)$?
* 2 cas :
① soit $v(a) = 0$, alors $v(A) = 1$ t. de v. de $\vee$
donc $v(A \vee B) = 1$ t. de v. de $\vee$
② soit $v(a) = 1$, alors $v(B) = 1$ t. v. de $\neg$ et $\vee$
donc $v(A \vee B) = 1$
donc $\forall v$, v réalisation de $\mathcal{H}$
impose que v soit réalisation de $A \vee B$
i.e. $\mathcal{H} \underset{\text{déf.}}{\models} A \vee B$

2.5.4. SF_R est-elle complète ?

(SF_R démontre-t-il toutes les conséquences valides ?)

non

Exemple : On a $a \models a \vee b$
Si la résolution était complète, on aurait
$a \vdash_{SF_R} a \vee b$
or cette démo est impossible (1 seule hypothèse !)

Rappels : - réduction à l'absurde

$\mathcal{H} \models C$ ssi $\mathcal{H} \cup \neg C$ est contradictoire

- tout ensemble de propositions possède un ensemble de clauses sémantiquement équivalent.

i.e. $\exists \mathcal{H}_{\mathcal{C}} \approx \mathcal{H} \cup \neg C$

On a $\mathcal{H} \models C$ ssi $\mathcal{H}_{\mathcal{C}}$ est contradictoire

Ce résultat est exploité pour obtenir une forme de complétude de SF_R

On dit que SF_R est complet pour la réfutation

$$\boxed{\mathcal{H}_C \text{ est contradictoire ssi } \mathcal{H}_C \vdash_{SF_R} \square}$$ non démontré ici

Finalement,

$$\boxed{\mathcal{H} \models C \text{ ssi } \mathcal{H}_{\mathcal{C}} \approx \mathcal{H} \cup \{\neg C\} \vdash \square}$$

2.5.5. En pratique

On possède les grandes lignes d'un algo pour valider syntaxiquement $\mathcal{H} \models C$

début :

- construire $\mathcal{H} \cup \{\neg C\}$; $\longrightarrow$ dans $\mathcal{P}$
- construire $\mathcal{H}_{\mathcal{C}} \approx \mathcal{H} \cup \{\neg C\}$; $\rightarrow$ pour venir dans $\mathcal{C}$
- inférer $\square$ à partir de $\mathcal{H}_{\mathcal{C}}$;

fin

Les 2 premières instructions (facile = cf TP Prolog)

- algo naïf pour inférer

$Res\,\mathcal{E}_c$ = ensemble des classes qui sont à un pas d'inférence de $\mathcal{E}_c$.

inférer ($\mathcal{E}_c$) $\mathcal{E}_c \subset \mathcal{C}$

début si $\square \in \mathcal{E}_c$ alors sortir "$\mathcal{E}_c$ est contradictoire"

fsi

soit $Res\,\mathcal{E}_c = \{c \mid \exists c_i \in \mathcal{E}_c, \exists c_j \in \mathcal{E}_c, c_i ; c_j \vdash c\}$

dans si $\mathcal{E}_c \cup Res\,\mathcal{E}_c = \mathcal{E}_c$ alors

sortir "$\mathcal{E}_c$ est satisfaisable" **fsi**

inférer ($\mathcal{E}_c \cup Res\,\mathcal{E}_c$)

finsoit

fin

2.6 Conclusion

Retenir les 2 approches :
- syntaxique
- sémantique

qui permettent de valider des argumentations

Complexité :
- sémantique : t. de v. de 2^n lignes (n prop. atom)
- syntaxique : combinatoire de choix des 2 clauses par la Résolution

Constat ← Constatation qui ne surprend pas :
Le problème de la satisfaisabilité d'un ensemble de propositions (SAT) est np-complet (i.e on pense qu'il n'existe pas de solution algorithmique à ce problème en temps polynomial

(3) Le langage des prédicats

3.1. La syntaxe

3.1.1 Pourquoi un nouveau langage ?

- langage des propositions : découverte des principes
Mais ce langage n'est pas suffisamment puissant, précis pour certaines argumentations (pour faire des maths !)

Exemple
2 énoncés :
"Pierre aime Marie" p_1
"Pierre aime tout le monde" p_2

dans P ce serait 2 propositions atomiques
Mais survient un problème pour les valuations de p_1 et p_2 qui dépendent l'une de l'autre

$$\begin{cases} v(p_1) = 0 \\ v(p_2) = 1 \end{cases}$$

ici P est pris en défaut

3.1.2 Nouveautés dans le langage des prédicats

1) introduction des prédicats

Les **prédicats** sont des énoncés articulés autour d'un verbe (aime) et qui font intervenir 1 ou plusieurs acteurs. Ils remplacent les prop. atomiques.

Ainsi "Pierre aime Marie" et "Pierre aime tout le monde" sont 2 récurrences du m̂ prédicat aime dont les acteurs ont changé.

Arité d'un prédicat : nbre d'acteurs

Si arité 0 on retrouve les prop. atomiques

2) introduction de la quantification et des variables

"quelqu'un aime Marie" paraphrasé il existe x, t.q. "x aime Marie"

variable

"Jean aime tout le monde"

quelque soit x, "Jean aime x"

x désigne un acteur variable (qui varie sur un domaine)

3) introduction des fonctions

Les acteurs peuvent être désignés directement ou de façon détournée.

"il existe x tel que "x aime Marie"

"il existe x tel que $\underbrace{\text{le frère de } x}_{\text{fonction}}$ aime $\underbrace{\text{la tante de Marie}}_{\text{fonction}}$

$\exists\, x$ tq $52 = x + 2$

"il existe x tq 52 égale la somme de x et de 3

3.1.3. Une grammaire des prédicats

Le vocabulaire terminal :

- 4 connecteurs : ⇒, ¬, ∨, ∧
- 2 quantificateurs : ∃, ∀
- les parenthèses : (,)
- la virgule
- une infinité de identvar
- une infinité de identfonct

Grammaire en 2 niveaux :

• niveau 1 : <fbf> ::= <formule atomique> |
(<formule composé>)

<formule composée> ::= ¬<fbf> |
<fbf><connect_bin><fbf> |
∃ <identvar><fbf>

<connect_bin> ::= ⇒ | ∨ | ∧

<fbf> = formule bien formée

• niveau 2 :

<formule atomique> ::= <ident pred> |
<ident pred>
(<terme>{, <terme>}*)

<terme> ::= <identvar> |
<identfonct> |
<ident fonct>(<terme>,
{, <terme>}*)

Notation de ce cours :

<identvar> : dernières lettres de l'alphabet en minuscules (u, v, ..., z)

<identfonct> : f, g, h arité > 0

<ident fonct> : arité = 0 : <ident const> début

de l'alphabet (a, b, c ..., i)

<ident pred> : majuscule

si arité 0 <ident prop>

<u>Exemples de fbf</u> :

$$\forall x \, (P(f(x)) \Rightarrow Q(g(x)) \wedge \forall y \, (R(x,y)))$$

$$P(a, f(a)) \vee \forall y \, (R(y,b) \to S(f(b)))$$

3.1.4 Restrictions

* Toute variable dans une formule doit être quantifiée.

$P(x) \vee R(a)$ interdit

$P(x) \wedge \forall x R(x)$ interdit

* Les idents de variable sont tous différents quand ils notent des variables différentes

$\forall x \, (P(x) \Rightarrow \forall x \, (R(x)))$ interdit

$\forall x \, (P(x) \Rightarrow \forall y \, (R(y)))$ équivalent

Ces restrictions ne démontrent pas $\mathcal{L}$, mais simplifient notre étude

dans 1 ens. (occurence libre, variable libre, occurence lié, variable lié) cf λ-calcul

3.2. La sémantique de $\mathcal{L}$

Dans $\mathcal{P}$ il fallait savoir calculer la valeur (0 ou 1) de toute <u>proposition</u>.

Idem dans $\mathcal{L}$: calculer la valeur (0 ou 1) de toute <u>formule</u>.

Cela dépend du sens que nous donnerons aux symboles qui le composent.
Bcp + compliqué que dans P.

3.2.1 Interprétation

Définition Une interprétation d'une formule G est définie par la donnée de 5 composants :
1) le choix d'un domaine D, dit domaine d'interprétation
2) l'assignation d'un élément de D à chaque <identconst> $\{a, b, c, d, e\}$
3) l'assignation d'un élément de $\{0,1\}$ à chaque <identprop> P/0, Q/0
4) l'assignation d'une application de D^n dans D à chaque <identfonct> d'arité $n > 0$
5) l'assignation d'une application de D^n dans $\{0,1\}$ à chaque <identpred> d'arité $n > 0$

On dit qu'on a alors une interprétation de G sur D

Exemple: $G = \forall x \left(\underline{R}(\underline{x}) \Rightarrow \underline{T}(f(\underline{x}), \underline{a}) \right)$

Interprétation

I_1 : 1) $D = \{4,5\}$
2) $a \to 4$
3)
4) $f \to \begin{cases} 4 \to 5 \\ 5 \to 4 \end{cases}$
5) $R \to \begin{cases} 4 \to 1 \\ 5 \to 0 \end{cases}$

$T \to \begin{cases} 4,4 \to 1 \\ 4,5 \to 0 \\ 5,4 \to 1 \\ 5,5 \to 1 \end{cases}$

Interprétation :

$D = \{$ Jeanne, Marc, Valérie, Luc $\}$

$a \rightarrow$ Valérie

$f \rightarrow$ frère

$R \rightarrow$ est une femme

$T \rightarrow$ est plus âgé que ...

3.2.2. Tautologie, déduction, modèle

Il faut revenir sur les définitions. Mais avant nous donnons un procédé de calcul de la valeur de vérité de toute formule (pendant de t. de v.)

Mais ce procédé est purement théorique (mise en œuvre impossible car ne termine généralement pas)

Soit I une interprétation, raisonnons sur la forme de $G \in \mathcal{L}$

1) G est une proposition $\mapsto$ sa valeur est la valeur assignée à G dans I

2) G est un prédicat (ex : $P(x,y)$)

a) G ne contient aucune variable (ex : $P(a,b)$) $\rightarrow$ prendre les assignations spécifiées par I

b) G contient des variables (ex : $P(x,y)$) $\rightarrow$ on calcule une valeur par assignation possible de valeurs de D pour chacune des variables (ne termine pas si D infini)

3) G est de la forme $\forall x\, G'$: la valeur de G est 1 si toutes les valeurs de G' sont 1, 0 sinon

4) G est de la forme $\exists x\, G'$: la valeur de G est 0 si toutes les valeurs de G' sont 0, 1 sinon

5) G de la forme $\begin{cases} G_1 < ab > G_2 \\ \text{ou } \neg G_3 \end{cases}$ cf langage des propositions

Définition Une interprétation I est un modèle de G si la valeur de G est 1 selon I

Exemple: $G = \forall x\,(A(x) \vee B(x))$

$I_1 \begin{cases} D = \{true\} \\ A \to true \to 1 \\ B \to true \to 1 \end{cases}$

I_1 est un modèle de G.

$I_2 \begin{cases} D = \mathbb{N} \\ A \text{ être pair} \\ B \text{ être impair} \end{cases}$

I_2 est un modèle de G.

Définition: Une fbf est une tautologie si elle est vraie pour toutes les interprétations

Exemple: $\forall x\,(P(x) \vee \neg P(x))$ tautologie de $\mathcal{L}$

Définition: 2 fbf A et B sont logiquement équivalents lorsque, pour toute interprétation elles ont même valeur. On note $A \approx B$

- équivalence remarquable: $\neg(\exists x A) \approx \forall x (\neg A)$
  $\neg(\forall x A) \approx \exists x (\neg A)$

Définition: Un ensemble $\mathcal{H}$ de fbf est satisfaisable ssi $\exists$ une interprétation I telle que $\forall A \in \mathcal{H}$, I est modèle de A.

Définition: Un ensemble $\mathcal{H}$ de fbf est contradictoire ssi pour toute interprétation I $\exists A \in \mathcal{H}$, telle que I n'est pas modèle de A

Définition: Une fbf A est dite conséquence logique d'un ensemble $\mathcal{H}$ de fbf ssi tout modèle de $\mathcal{H}$ est aussi modèle de A. On note $\mathcal{H} \models A$

fbf = formule bien formulé

Cas limite : quand $\mathcal{H} = \emptyset$ on note $\models A$
on prononce "A est une tautologie"

Remarque importante : Les concepts passés en revue, sont similaires à ceux vus dans $\mathcal{P}$. Pourtant les choses ne sont pas si simples.

ON NE PEUT PAS, PAR LE CALCUL, DANS LE CAS GENERAL, VERIFIER QU'UNE FBF EST UNE TAUTOLOGIE !

i.e. il n'existe aucun algorithme qui, à la donnée d'une fbf quelconque, répond si A est ou non une tautologie.

LE CALCUL DES PREDICATS EST INDECIDABLE.

3.2.3 Principaux résultats

théorème sémantique de la déduction

Soit $\mathcal{H} \subset \mathcal{L}$
Soit A et B dans fbf, $A \in \mathcal{H}$
$\mathcal{H} \models A \Rightarrow B$ ssi $\mathcal{H} \cup \{A\} \models B$

Corollaire : (réduction à l'absurde)

$\mathcal{H} \models A$ ssi $\mathcal{H} \cup \{\neg A\}$ est contradictoire — idem $\mathcal{P}$

3.3 L'approche syntaxique

Dans $\mathcal{L}$ il existe un système formel qui est une adaptation de SF_1.

Il est bâti sur $\mathcal{L}(\Rightarrow, \neg, \forall)$

Il possède des axiomes supplémentaires

$\forall x \; A \Rightarrow A[x \rightarrow t]$ axiome de particularisation
$\forall x \; (D \Rightarrow B) \Rightarrow (D \Rightarrow \forall x \; B)$

1 règle d'inférence supplémentaire

$A \vdash_{SF_1} \forall x \; A$ (généralisation)

Mais ce SF a les m̂ défauts que SF_1 de $\mathcal{P}$
On jette non implémentable

3.3.1 En amont de la Résolution

Une extension de SF_1 pourra, dans certains cas, permettre de pratiquer des démonstrations automatiques.
L'étude de cette extension permet de comprendre le fonctionnement de PROLOG.

Evidemment, SF_R n'a pas pour langage $\mathcal{L}$ tout entier. Donc avant d'espérer valider une argumentation, on va faire subir des transformations aux fbf pour obtenir des clauses de $\mathcal{L}$.

Ce serait bien que ces transformations soient purement syntaxiques (qu'elles ne changent pas le sens).

Pour passer $\mathcal{L}$ aux clauses de $\mathcal{L}$, 2 séries de transformation.

1) mise sous forme prénexe (3.3.2)
2) mise sous forme clausale (3.3.3)

3.3.2 Mise en forme prénexe

Définition : Une fbf est sous forme prénexe quand elle s'écrit

$$\boxed{Q_1 x_1 \dots Q_n x_n B}$$

avec
B fbf sans quantification
$Q_i \in \{\exists, \forall\}$

Résultat important :
Toute fbf $\in \mathcal{L}$ admet une forme prénexe logiquement équivalente.

Démonstration - construction

Soit $F \in \mathcal{L}$

1) réécrire F en remplaçant les sous formules de la forme $A \Rightarrow B$ par $\neg A \vee B$ (equiv. remarquable)

On obtient

$$F_1 \simeq F$$

2) réécrire F_1 en accolant les $\neg$ aux formules atomiques. Pour cela on utilise les réécritures (issues des équiv. remarquables).

$$\neg\neg G \to G$$
$$\neg(A \vee B) \to \neg A \wedge \neg B$$
$$\neg(A \wedge B) \to \neg A \vee \neg B$$
$$\neg(\exists x P) \to \forall x \neg P$$
$$\neg(\forall x P) \to \exists x \neg P$$

On obtient $F_2 \simeq F_1 \simeq F$

3) réécrire F_2 en amenant les quantificateurs en tête et en respectant leur ordre.

On obtient $F_3 \simeq F_2 \simeq F_1 \simeq F$

3.3.3 Mise sous forme clausale

Bcp plus délicat au niveau des propriétés.

Soit F sous forme prénexe

1) skolémisation (SKOLEM)

La skolémisation consiste à réécrire F en éliminant les quantificateurs existenciels

• Que fait-on des variables concernées ?

Elles sont remplacées par une fonction des variables universellement quantifiées figurant avant dans F.

Exemples:

$$\forall x \cancel{\exists y}\ P(x, y)$$
$$\to \forall x (P(x, f(x)))$$

$$\forall x \forall y \forall z\ (P(x,z) \vee Q(f(y), g(z)))$$
$$\to \forall x \forall y\ (P(x, h(x,y)) \vee Q(f(y), g(h(x,y))))$$

⚠ les fonctions doivent être nouvelles.

$\cancel{\exists x}\ \forall y\ Q(\underline{x}, y, a)$

$\rightarrow \forall y,\ Q(\underline{b}, y, a)$

Cette transformation est intuitivement acceptable

$\Rightarrow \forall x \exists y\ P(x,y)$

$\forall x \exists y_z\ P(x, y_x)$ le y dépend de x (est fonction de) x

$\rightarrow \exists y \forall x\ P(x,y)$

le même y pour tous les x.

Pourtant la formule obtenue n'est pas logiquement équivalente

Ainsi la skolémisation de $F = \forall x \exists y\ P(x,y)$

produit $F_1 = \forall x\ P(x, f(x))$

Si $F \approx F_1$ elles auraient même valeur pour toute interprétation

or $I : D = \mathbb{Z}$ | la valeur de F est 1

$P = <$ | la valeur de F_1 est 0

$f =$ prédécesseur

Ben alors ? On jette ?

Pas de panique : la skolémisation possède une propriété intéressante pour nous (Résolution)

Soit F un ensemble de formules prénexes.

Soit F_1 l'ensemble F skolémisé.

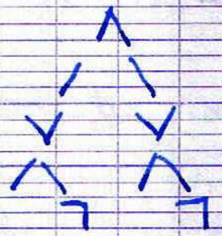
F est contradictoire $\Leftrightarrow$ F_1 est contradictoire

On est rassurés car résolution pratique des démos par l'absurde.

2) Réécrire les fbf de F_1 en éliminant les quantificateurs universels. (allègement d'écriture)

Les seules variables de F_D sont universellement quantifiées.

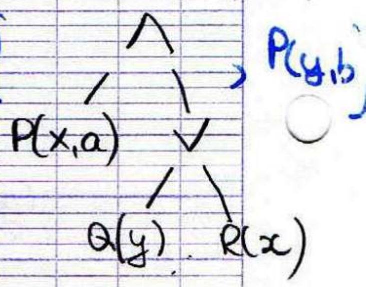
3) Passage en F.N.C. (cf proposition) $(\vee, \wedge, \neg)$



4) Elimination des connecteurs $\wedge$

On arrive à un ensemble de clauses.

$\{P(x,a) \wedge (Q(y) \vee R(x)), P(y,b)\}$ $\{$



$$\downarrow$$

$\{P(x,a), Q(y) \vee R(x), P(y,b)\}$

5) Réécrire l'ensemble de clauses (issu de 4) en veillant à ce que le même <identificateur> n'apparaisse dans 2 clauses différentes.

Pour cela, procéder à un renommage cohérent des variables.

$\{P(x,y) \vee \neg Q(z), \neg P(x,z) \vee R(x), Q(x,y)\}$

$\{P(x,y) \vee \neg Q(x), \neg P(u,z) \vee R(u), Q(v,w)\}$

3.3.4 La Résolution

Le SF attendu :

SF_R langage : le langage des clauses de $\mathcal{L}$

schéma d'induction

base : vide

règle d'inférence : Résolution

$l_1 \vee D_1, \neg l_2 \vee D_2 \vdash D_1' \vee D_2'$

Remarques : * l_1 et l_2 sont 2 formules atomiques ayant même <identpred>

ex : $\begin{cases} l_1 = P(x, b) \\ l_2 = P(f(y), z) \end{cases}$

* la clause $D_1' \vee D_2'$ produite par la règle d'inférence est dite résolvante.

* l_1 et $\neg l_2$ ne sont pas forcément en tête de leur clause.

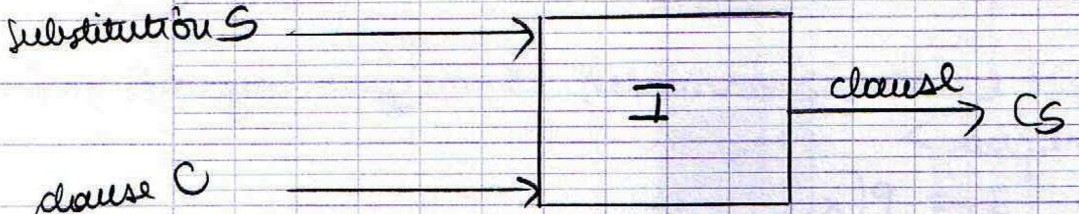
Le SF_R est utilisé comme le SF_R des propositions.
i.e. on va montrer (en essayant d'inférer $\square$) qu'un ensemble de clause est contradictoire.
Ce qui change, qu'on va étudier :
1) les conditions d'application de la règle d'inférence.
2) l'application elle-même de la règle.

3.3.4.1. Condition d'application de $l_1 \vee D_1, \neg l_2 \vee D_2 \vdash D_1' \vee D_2'$

Définition : Une substitution est un ensemble de couples $\{<var1, terme1>, \ldots, <var\ n, terme\ n>\}$ (voir syntaxe de $\mathcal{L}$) tel que :
- les var i sont toutes différentes et
$\forall i \in [1,n], \forall j \in [1,n]$ var i n'apparaît pas dans l'écriture de terme j

Exemple : $\{<x, \overline{f(a,y)}>, <z, \overline{y}>, <u, \overline{g(a)}>\}$ OK
$\{<x, f(a,y)>, <y, b>\}$ n'est pas une substitution

Définition : Une instanciation est une opération qui consiste à appliquer une substitution S à une clause C. Le résultat de cette opération est une instance $\underbrace{C_S}_{\text{clause}}$.



La clause C_S est obtenue en remplaçant les variables de C par le terme qui leur est associé dans S.

<u>Exemple</u>: $C = P(f(x,y), g(b,z))$
$S = \{ <x, h(a)>, <u, b> \}$

$C_S = P(f(h(a),y), g(b,z))$

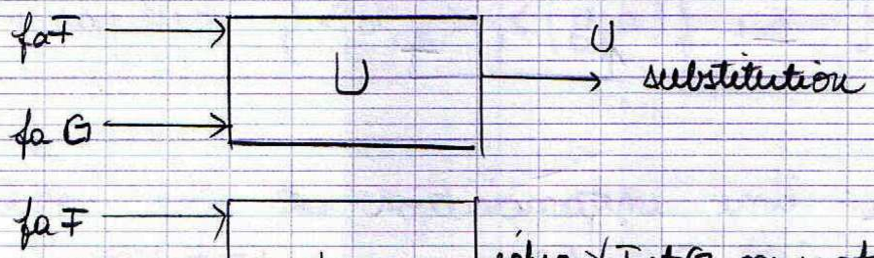
! Instancier c'est simplement remplacer les variables par leurs valeurs (c'est un terme)

* <u>le côté syntaxique</u> ⇒ utilisation de la Résolution
1) passer de $\mathcal{L}$ aux clauses de $\mathcal{L}$ en 2 étapes (forme prénexe)
2) le système formel SF_R sur <u>les clauses de $\mathcal{L}$</u>

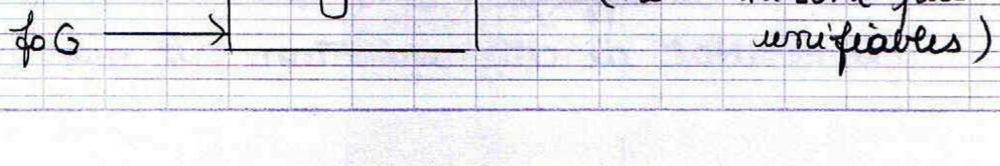
$\boxed{a \vee A ,\ \neg b \vee B \vdash}$

<u>Définition</u>: Deux fbf atomiques F et G sont unifiables si il existe une substitution S telle que $\boxed{F_S = G_S}$

<u>Définition</u>: L'unification est une opération qui consiste à confronter 2 formules atomiques pour leur trouver, si possible, une unification



<u>ou</u>



Quand l'opération U réussit on a évidemment
$$F_U = G_U$$

Unifier c'est simplement chercher une solution à l'équation syntaxique $F = G$

Autrement dit par quelles valeurs (termes) remplacer les variables de F et G pour obtenir l'identité syntaxique de 2 membres de l'équation $F = G$

Quand cette équation syntaxique a 1 solution cette solution est U

$$\underbrace{5x+2}_{\langle x, F \rangle} = \underbrace{18x-4}_{G}$$

Pour conclure ce paragraphe :

$$l_1 \vee D_1, \neg l_2 \vee D_2 \vdash D_1' \vee D_2'$$

est applicable si les formules atomiques l_1 et l_2 sont unifiables.

3.3.4.2 Production de la résolvante $D_1' \vee D_2'$

$D_1' \vee D_2'$ est une instance de $D_1 \vee D_2$ obtenue en appliquant l'unificateur de l_1 et l_2