Logique : choses importantes

Langage symbolique :
- Pour les propositions
- Et les relations entre propositions

La négation, la disjonction “ou” et l’implication par exemple.

Langages :

Langage formels et alphabets : Définitions :
- Alphabet : ensemble de symboles utilisables : {a, b, c}
- Mot ou expression : une suite d’élément de l’alphabet : aaba
- Longueur d’un mot : nombre de symboles qui le composent : long(aaba) = 4
- Le mot vide : longueur 0 noté epsilon
- Concaténation noté . : Opération binaire associative sur les mots qui désigne le mot obtenu en mettant bout à bout les deux opérandes.

Langage :
- Soit A un alphabet, un langage **L sur A** est un **sous-ensemble** des mots que l’on peut construire avec A. Exemple : A={a, b, c}
  L1={aba, ba, cabb}
- Soit A un Alphabet **A*** désigne le langage de tous les mots que l’on peut construire sur A : A*{epsilon, a, b, c, aa, ab, ac, ba, bc, ……. ,ccccaaaabbba….}
- Tout langage sur A est sous-ensemble A* etc

Les règles de construction, des fonction partiellement définies sur A* : n est appelé l’arité de la fonction.

Pour définir un langage L par induction structurelle on doit :
- Donner un alphabet A
- Donner un sous-ensemble B de A* appelé la base
- Donner un ensemble R de règles de construction sur A*

L est alors défini (par induction) comme le plus petit ensemble L inclu A* tel que :
(base) L contient B
(cons) pour toute r appartient à R et tout m1,…m_n appartient à L (où n est l'arité de r), si r(m1,…mn) est défini alors r(m1,…mn) appartient à L

Exemple de définition d’un langage Lp :
• Définition d’un langage Lp
– Alphabet A={1,p,e}
– (base) {pe}
– (cons) Soit m un mot de Lp
• (r1) 1.m.1 est un mot de Lp
• (r2) si m contient un e (c’est-à-dire m=m'.e.m”)
alors m'.1.e.1.m” est un mot de Lp
r1 : A* → A* **r2 : A* → A* définie uniquement pour les mots contenant au moins un e**

Logique : choses importantes

m↦r1(m)=1.m.1　　　　**m=m'.e.m'' ↦ r2(m)=m'.1.e.1.m"**

• Ainsi Lp={pe, 1pe1, p1e1, 11pe11, 1p1e11…}
Exemple de définition inductive d'une fonction sur les mots d'un langage défini par induction

• Définition d'une fonction donnant le nombre de 1 d'un mot de Lp
nbr1(1pe1)=2, nbr1(1p1e11)=4
• Par induction :
nbr1 : Lp→ℕ
m↦nbr1(m)
(base) si m=pe alors nbr1(pe) = 0
(cons)
- si m=r1(m')=1m'1 alors nbr1(m) = 2 + nbr1(m') on sait qu'il y a deux 1 et on regarde ensuite si m' n'est pas le résultat déjà d'une autre règle qui inclurait des 1.

- si m=r2(m'em")=m'1e1m"
alors nbre1(m)=2 + nbr1(m'em") Pareil ici on sait qu'il y a deux 1 et on regarde ensuite pour m' et m".

Syntaxe langage Lp :

Soit l'ensemble des mots définis sur l'alphabet **A=SuCuDuL** avec

- **S** : ensemble dénombrable de symboles propositionnels notés en lettres minuscules dans les exemples : S={ s, p, q, r}
- **C** : est l'ensemble des connecteurs logique {¬,^,v,→, ↔} non (négation), et (conjonction), ou (disjonction), implique /si alors (implication), équivalent/si-et-seulement-si (équivalence)
- **D** : {(,)} est un jeu de parenthèses
- L : {T,⊥} les constantes logiques Top(vrai) et Bottom(absurde)

FBF = formules bien formées, on définit **PROP(s)** l'ensemble des **FBF** de la logique des propositions (ou propositions), construites sur S par induction :

(base) S ∪ {⊥,T}
(cons) Soit P et Q des mots de (S∪C∪D∪L)*, on dispose de 5 règles de construction

r1(P) = ¬P
r2(P,Q) = (P ^ Q)
r3(P,Q) = (P v Q)
r4(P,Q) = (P → Q)
r5(P,Q) = (P ↔ Q)

Logique : choses importantes

Conventions :
- Les majuscules dénotent des propositions (FBF) : P, Q, R ..
- Les minuscules dénotent des symboles propositionnels p, q, r

Attention :
- Tout symbole propositionnel est une proposition (p est à la fois une FBF de PROP({p,q}) et un symbole de {p,q})
- Le contraire n'est pas vrai ! (p ^ q) est une fbf de PROP({p,q}) mais n'est pas un symbole
- Une formule réduite à un symbole propositionnel ou aux constantes logiques T ou ⊥ est appelée une formule atomique ou atome

Les macros :
- ⊥ pour (P ^ ¬P)
- T pour ¬⊥
- (P v Q) pour ¬(¬P ^ ¬Q)
- (P → Q) pour (¬P v Q)

L'ensemble des symboles propositionnels
d'une fbf : SP ensemble des symboles propositionnels

SP : PROP(S) à 2^S qui à une fbf P associe :
(base) si P appartient à S, SP(P) = {P}
si P=T ou P=⊥, SP(P) = {}
(cons) r1 : si P=¬Q, SP(P) = SP(Q)
r2, r3, r4, r5 :
si P=(Q^R) ou P=(QvR) ou P=(Q→R) ou P=(Q↔R),
SP(P) = SP(Q)∪SP(R)

Nombre de connecteurs d'une fbf :
PROP(s) l'ensemble des **FBF** de la logique des propositions (ou propositions), construites sur S par induction :

nbc: PROP(S) → N
(base) nbc(P) = 0
(cons)
r1 : P=¬Q, nbc(P) = 1+nbc(Q)
autres règles : P=(Q c R) avec c connecteur binaire
nbc(P) = 1+nbc(Q)+nbc(R)
Soit **ARBO(S)** l'ensemble des arborescences dont :
- Les feuilles sont étiquetées par des éléments de Su{T,⊥}
- Les autres noeuds par des connecteurs avec respect de l'arité
  - Un noeud étiqueté par un connecteur binaire a deux fils (gauche et droit)
  - Un noeud étiqueté par le connecteur unaire ¬ a un fils

- A toute FBF correspond une unique arborescence et vice versa :

Logique : choses importantes

Soit fbf2arb (l'arbor. associée à une fbf) : PROP(S) → ARBO(S)
(base) fbf2arb(P)= arbor. réduite à un sommet étiqueté par P
(cons)
r1: P=¬Q, fbf2arb(P)= arbor. de racine étiquetée par ¬ ayant comme unique fils la racine de fbf2arbor(Q)
autres r : P=(Q c R), fbf2arb(P)= arbor. de racine étiquetée par c ayant comme fils gauche la racine de fbf2arbor(Q) et comme fils droit la racine de fbf2arbor(R)

Conventions :

- Si les arbres montrent bien la structure d'une FBF, ils ne sont pas "pratiques" pour l'écriture.
- La notation **infixée** est la manière classique d'écrire les FBF mais elle nécessite l'utilisation de parenthèses : ex : ¬((¬q ^ ((p v (q v p)) → q)) → ¬p)
- On peut par des conventions de priorité de connecteurs éliminer des parenthèses :
  ¬ est prioritaire sur ^ et v qui sont prioritaires sur → et ↔
  ((p v (p → q)) → ¬p) devient par convention **p v (p → q) → ¬p**
- Les notations postfixée et préfixée ont l'avantage de ne pas nécessiter de parenthèses.

Sémantique formelle :

On oppose la sémantique formelle à la sémantique intuitive :

- La **sémantique formelle** consiste à associer des structures particulières prises dans une théorie mathématique (le plus souvent celle des ensembles) à chaque élément de syntaxe : on parle d'interprétation
- La **sémantique intuitive** consiste à associer une notion du monde réel à chaque élément de syntaxe.

La logique des propositions est une logique bivalente :

- Donner un sens **formel** aux propositions c'est leur **associer une valeur prise** dans un ensemble à **deux éléments Bool = {0, 1} ou faux/vrai**, la valeur de vérité
- La sémantique **formelle** de la logique des propositions se limite donc aux opérations **réalisables sur cet ensemble Bool** : le calcul booléen.

La valeur de vérité d'une FBF est calculée à partir de **l'interprétation de ces composants** : on parle de **sémantique compositionnelles**

On interprète les connecteurs et constantes logiques toujours de la même façon

Logique : choses importantes

Choisir une interprétation consiste donc uniquement à spécifier une application de S dans Bool. (L'interprétation des connecteurs et des symboles constants étant la même dans toutes les interprétations

Il faut fixer les fonctions sur les valeurs de vérité qui vont interpréter les connecteurs de la logique :

- Des fonctions d'arité 0 pour les constantes parmi les 2 possibles
  $f1_0()=0$ ou $f2_0()=1$

- Une fonction d'arité 1 pour la négation parmi les possibles
  $f1_1(0)=0$ ou $f2_1(0)=0$ ou $f3_1(0)=1$ ou $f4_1(0)=1$
  $f1_1(1)=0$ ou $f2_1(1)=1$ ou $f3_1(1)=0$ ou $f4_1(1)=1$
- Pour les fonction d'arité 2 vous aurez donc 16 possibilités

Interprétation :

- Définir une logique c'est choisir une interprétation des connecteurs (qui correspond à nos intuitions)
- Les constantes sont interprétées par chacune des 2 valeur de Bool : $I(\perp) = 0$, $I(T) = 1$

Les connecteurs sont interprétés par des fonctions de $Bool^n \rightarrow Bool$
- $I(\neg)$ = NON : Bool→Bool t.q. NON(0)=1 et NON(1)=0
- $I(\wedge)$ = ET : $Bool^2 \rightarrow Bool$ t.q. ET(a,b) = 1 ssi a=b=1
- $I(v)$ = OU : $Bool^2 \rightarrow Bool$ t.q. OU(a,b) = 0 ssi a=b=0
- $I(\rightarrow)$ = SI ALORS : $Bool^2 \rightarrow Bool$ t.q. SI ALORS(a,b)=0 ssi a=1 et b=0
- $I(\leftrightarrow)$ = SSI :$Bool^2 \rightarrow Bool$ t.q. SSI(a,b)=1 ssi a=b

Interprétation des symboles :

Choix d'une application de S→B :

Logique : choses importantes

- Soit S = {p,q,r} on peut choisir entre 8 applications
  I1: S→B
  I1(p)=0, I1(q)=0, I1(r)=0
  I2: S→B
  I2(p)=0, I2(q)=0, I2(r)=1
  …
  I8: S→B
- Si card(S)=n, il y a 2^n interprétations différentes
  (2n applications différentes de S dans B)

Valeur de vérité d'une proposition : La valeur de vérité d'une proposition P dépend :
- De la logique utilisée (Interpré des connecteurs)
- De l'application des symboles propositionnels dans B choisie (Interpré des symboles)

On note v(P, I) la valeur de vérité de P dans l'interprétation I. La définition de v se fait par induction :

Soit I une interprétation des symboles propositionnels d'une fbf P, on définit la valeur de vérité de P dans l'interprétation I notée v(P,I) :
(base) P appartient à Su{T,⊥}, v(P,I) = I(P)
(cons)
r1 : P=¬Q, v(P,I)=v(¬Q,I)=I(¬)(v(Q,I)) = NON(v(Q,I))
r2 : P=(Q^R), v(P,I)=ET(v(Q,I),v(R,I)) (valeur d'inter de Q et de I)
r3 : P=(QvR), v(P,I)=OU(v(Q,I),v(R,I))
r4 : P=(Q→R), v(P,I)=SI ALORS(v(Q,I),v(R,I))
r5 : P=(Q↔R), v(P,I)=SSI(v(Q,I),v(R,I))

Une table de vérité d'une FBF P est un tableau ayant :
- Pour indice de ligne les 2^n interprétations possibles des n symboles propositionnels de P
- Pour indice de colonne les sous-FBF de P(dernière colonne étant P)
- Dans une case de ligne I et de colonne Q la valeur v(Q,I)

Ainsi la dernière colonne contient les valeurs de P pour toutes les interprétations possibles des symboles de P.

Modèle et contre-modèle :
- Une interprétation I t.q. **v(P,I) = 1** est appelée un **modèle de P**
- on dit qu'elle **satisfait P** (et l'on note parfois I |= P)
- Une interprétation I t.q. **v(P,I) = 0** est appelée un **contre-modèle de P**

Logique : choses importantes

Caractérisation des propositions

- Une fbf est **satisfiable si elle possède au moins un modèle**
- Une fbf est **contingente** si elle possède au moins un **modèle et un contre-modèle**
- Une fbf est **valide** si **toute interprétation est un modèle**
- Une fbf est **insatisfiable** si elle ne **possède pas de modèle**

Propriétés :

- P est satisfiable ssi ¬P n'est **pas valide**
- P est insatisfiable ssi ¬P est **valide**
- P est contingente ssi ¬P est **contingente**

PROP(S) est partitionné en :
– Les propositions insatisfiables
– Les propositions contingentes
– Les propositions valides

On étend la satisfiabilité à un ensemble de FBF :

- {P1,P2,…P_n} est dit **consistant** ssi il existe un modèle commun aux n propositions
- sinon {P1,P2,…P_n} est dit **contradictoire** (ou inconsistant)

Propriété :

- {P1,P2,…Pn} est **consistant** ssi P1^P2^…^P_n est satisfiable
- {P1,P2,…Pn} est **contradictoire** ssi P1^P2^…^P_n est insatisfiable

Equivalence logique et substitution :

Deux FBF P et Q sont logiquement équivalentes ssi pour toute interprétation elles ont la même valeur de vérité. On note P≡Q.

- P≡Q ssi pour tout Interprétations : I |= P ssi I |= Q

Attention à ne pas confondre :

- Equivalence logique P≡Q (notion sémantique)
- Egalité syntaxique P=Q (notion syntaxique)
- Connecteur équivalent (P↔Q) fbf

Théorème : P≡Q ssi (P«Q) est valide

Propriétés des connecteurs vis à vis de l'équivalence :

- L'équivalence logique permet de démontrer les propriétés algébriques des connecteurs : les formulaires sont des équivalences logiques de base qui énoncent ces propriétés
- On peut les démontrer à l'aide des tables de vérité

Formulaires : P,Q,R appartient à PROP(S)

Logique : choses importantes

• **Idempotence** de ^ et v
$(P \wedge P) \equiv P$ $(P \vee P) \equiv P$

• **Associativité** de ^ et v :
$((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$ $((P \vee Q) \vee R) \equiv (P \vee (Q \vee R))$

• **Commutativité** de ^ et v
$(P \wedge Q) \equiv (Q \wedge P)$ $(P \vee Q) \equiv (Q \vee P)$

• **Distributivité** du ^ par rapport à v (et vice versa)

$((P \wedge (Q \vee R) \equiv ((P \wedge Q) \vee (P \wedge R))$

$((P \vee (Q \wedge R) \equiv ((P \vee Q) \wedge (P \vee R))$

• Double négation

$\neg\neg P \equiv P$

• Lois de De Morgan
$\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$ $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$

• Implication et équivalent
$(P \rightarrow Q) \equiv (\neg P \vee Q)$ $(P \leftrightarrow Q) \equiv ((P \rightarrow Q) \wedge (Q \rightarrow P))$

• Négation et True et Absurde
$\neg P \equiv (P \rightarrow \perp)$
$(P \vee \neg P) \equiv T$ $(T \wedge P) \equiv P$ $(T \vee P) \equiv T$
$(P \wedge \neg P) \equiv \perp$ $(\perp \wedge P) \equiv \perp$ $(\perp \vee P) \equiv P$

Les macros :

- $\perp$ pour $(P \wedge \neg P)$
- T pour $\neg \perp$
- $(P \vee Q)$ pour $\neg(\neg P \wedge \neg Q)$
- $(P \rightarrow Q)$ pour $(\neg P \vee Q)$

Étant donnée une fbf P, la substitution consiste à remplacer une occurrence (plusieurs ou toutes) d'une sous-fbf Q de P par une autre fbf R.

Conséquence logique :

Définition : Une FBF est conséquence logique d'un ensemble de FBF $\{H_1 \ldots\ldots H_k\}$ ssi tout modèle de H_j pour j de 1 à k est un modèle de C (Pour toute interprétation I telle que $v(H_j, I) = 1$, pour tout j de 1 à k on a $v(C, I) = 1$)

On note {H1, ..., Hk} |= C

Logique : choses importantes

- La notion de conséquence logique peut être considérée comme une modélisation d'un raisonnement valide.

Propriétés de la conséquence logique : Exemples :
{p,q} |= p^q
{p} |= pvq (pour un q quelconque)
{p,p→q} |= q (modus ponens)
{¬q,p→q} |= ¬p (modus tollens)

Propriétés
- Si P est valide alors E |= P (pour un E quelconque y compris ensemble vide, on note |=P)
- Si E est contradictoire alors E |= P (pour un P quelconque)
- P º Q ssi {P}|=Q et {Q}|=P

Théorème :
1 $\{H_1, ..., H_k\}$ |= C ssi
2 H_1^... ^H_k → C est valide ssi
3 H_1^... ^H_k^¬C est insatisfiable

Pour le prouver on va faire une démonstration circulaire, avec 1→2, 2→3 et 3→1.

1→2 = H_1 … H_k |= P
Règles : H1 |= H2 donc H_1… H_k |= H_2 .. H_k

Ainsi le problème de la validité d'un raisonnement peut se ramener à celui de la validité ou de la satisfiabilité d'une formule.

Soit H1 .. Hk, c telles que $\{H_1, … H_k\}$ |= C et soit H_1^...^H_k → C n'est pas valide.
On a donc pour tout I tel que $v(H_1, I) = … = v(H_k, I) = 1$ on a $v(c, I) = 1$.
Il existe I_0 tel que $v(H_1,$ ^...^H_k→$c, I_0) = 0$
Par sémantique de l'implication :
$v(H_1$^...H_k, I0) = 1 et $v(C, I_0) = 0$
Par la sémantique du et
$v(H_1$, I0) = … = $v(H_k$, I0) = 1
$v(c, I_0) = 1$ faux d'où contradiction. **A savoir faire.**

Les formes normales et clausales :

Un **littéral** est une fbf réduite à un symbole propositionnel ou à la négation d'un symbole propositionnel
- on parle de littéral positif ou négatif non(s), p, toto, q, p^q n'en n'est pas un tout comme les constantes logiques par exemple.

On parle du littéral **opposé** d'un littéral donné : q est l'opposé de non(q) et inversement.

Logique : choses importantes

Une fbf est dite sous **forme conjonctive** lorsqu'elle est composée d'une conjonction de littéraux : exemple (^vq) ^ (non(p)vqvr) dans l'arborescence il faut que le ^ soit tout en haut et que les v soit tous en dessous et inversement.

Une fbf est dite sous **formes disjonctive** lorsqu'elle est composée d'une disjonction de conjonctions de littéraux : exemple (p^q) v (p^q^r) v (p^non(q))

Une fbf est dite sous forme **normale conjonctive** lorsqu'elle est sous forme conjonctive telle que tous les symboles propositionnels apparaissent exactement 1 fois dans chaque disjonction. **Exemple** de FNC : (p v q v r) ^ (non(p) v q v r) ^ (p v non(q) v r

- Remarque : Un littéral seul est à la fois **FNC** et **FND**
- Propriété : A une permutation des littéraux, conjonctions et disjonctions près, les FNC et FND sont uniques pour un S donné. (On peut donc tester l'équivalence de deux fbf en comparant syntaxiquement leur FNC et FND)

Pour passer d'une FNC à une FND, on introduit les symboles manquants à l'aide des équivalences suivantes : D équi (D v p) ^ (Dv non(p))
Exemple : (pvq) ^(non(p) v q v r) **on veut :**   (pvq **r?**) ^(non(p) v q v r)
= (p v q v non(r)) ^(p v q v r) ^(non(p) v q v r)

Les FND et FNC s'obtiennent à partir d'une table de vérité :

- FND on fait la disjonction des conjonctions des littéraux associés aux interprétations donnant la valeur à 1 à la fbf.

Une **clause** est une représentation ensembliste de disjonction de littéraux.

- Exemple : {non(p), q, non(r), s}
- La sémantique d'une clause est complètement définie par l'ensemble des littéraux qui la composent
- Une **infinité** de disjonctions de littéraux peut être associée à une **clause** mais elles sont toutes sémantiquement équivalentes. (idempotence, associativité et commutativité de la disjonction)

On associe la proposition absurde à la clause vide (noté 0(barré))

- Toute disjonction associée à une clause C={L_1, ….L_k) v Absurde
- La sémantique de 0(arré) est donc 0.

Une forme clausale est une représentation ensembliste d'une forme conjonctive :

- Exemple : {{p, q, non(r)}, {p, non(s)}, {non(r), non(s)}, {q}}
- La sémantique d'une forme clausale est complètement définie par l'ensemble des clauses qui la composent.
- Une infinité de formes conjonctives peut être associée à une forme clausale mais elles sont toutes sémantiquement équivalentes.

On associe la porposition T à l'ensemble vide de clauses. (noté {})

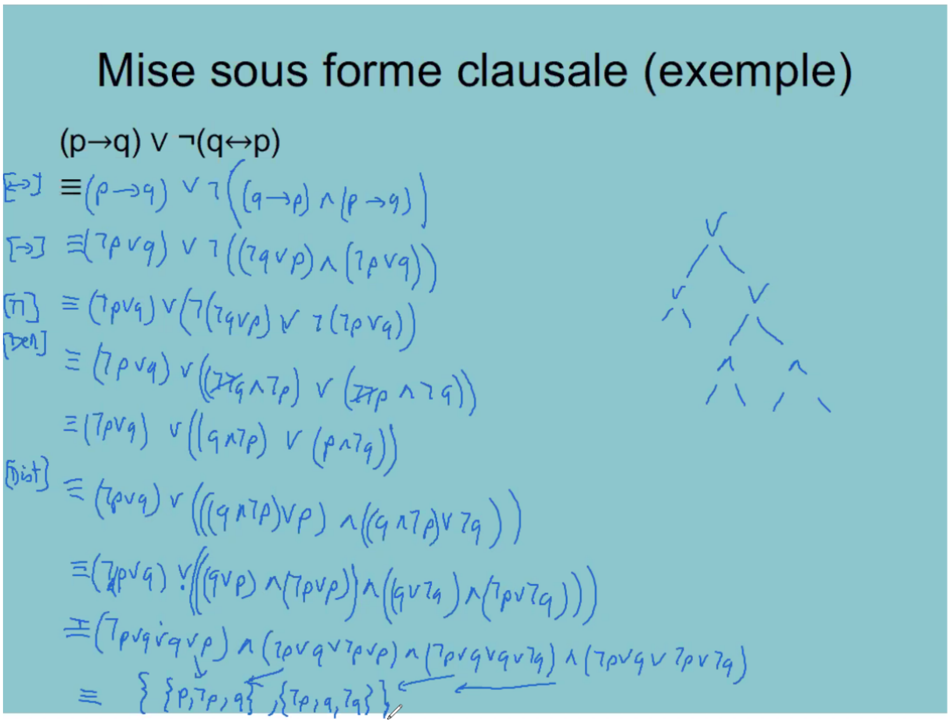
Logique : choses importantes

- Toute forme disjonctive associée à une forme clausale $F=\{C_1 \ldots C_k\}$ est logiquement équivalente à $((C_1 \wedge C_2 \ldots C_k) \wedge T)$
- La sémantique de l'ensemble d'une clause est donc 1

Mise sous forme clausale : théorème :

1. **Éliminer** les ↔ : (P↔Q) équi ((P→Q) ^ (Q→P))
2. **Éliminer** les → : (P→Q) équi (non(P) v Q)
3. **Ramener** la négation devant les symboles propositionnels et supprimer les négations multiples : non(non(P)) équi P **ou bien** non(PvQ) équi (non(P) ^ non(Q))
4. **Inverser** les disjonctions de conjonction : (Pv(QvR)) équi ((PvQ)^(PvR))
5. **Passer** de la forme conjonctive obtenue à sa forme clausale

Exemple de passage :



Dans la suite on emploie indifféremment le terme de clause pour parler de l'ensemble de littéraux ou d'une des fbf disjonctives que l'on peut lui associer.

Propriétés des **clauses** :

- Une clause est valide ssi elle contient un littéral et son opposé (on essaie de mettre tous les littéraux à 0)
- Une clause non vide ne contenant pas de littéraux opposés est contingente
- Seule la clause vide est insatisfiable

Logique : choses importantes

Propriétés :
- Si C inclu C' alors C|= C'
- Si C' non valide et C |= C' alors C inclu C'

Soit F une forme clausale et soit C une clause valide de F (qui contient un littéral et son opposé) on a F = F - {C}
- On peut donc éliminer les clauses tautologiques des formes clausales en conservant leur sémantique.

Une **clause de Horn** est une clause ayant au plus un littéral positif :
- Exemple : {non(p), q, non(r), non(s)} ou {non(r), non(s)}

On appelle règle de Horn, une fbf composée d'une implication entre deux conjonctions de littéraux positifs : exemple : $p_1 \wedge p_2 \ldots \wedge p_n \rightarrow c_1 \wedge c_2 \ldots \wedge c_n$
La forme clausale d'un ensemble de règles de Horn ne contient que des clauses de Horn

Méthode de résolution : CC jusqu'à forme clausale

Une formule est soit valide soit satisfiable sinon elle est contingente.

- Méthode est une répétition de choses à faire plusieurs fois une règle, permettant de produire une clause à partir de clauses existantes. Cette règle est appliquée itérativement jusqu'à tomber sur la clause vide.
  Cette méthode permet de déterminer si une forme clausale est insatisfiable, elle permet donc de savoir si
- Une fbf est insatisfiable (en passant à sa forme clausale)
- une fbf C est la conséquence logique d'un ensemble de fbf $\{H_1 \ldots H_k\}$

Cette méthode généralisée à la logique des prédicats et restreinte aux clauses de Horn est à la base du langage prolog.

Règle de résolution :
Soit C et C' deux clauses ayant des littéraux opposés, la résolvante de C et C' selon p est la clause res(C, C', p) = $\{L_1 \ldots L_k, M_1 \ldots M_k\}$ obtenue par union des littéraux restants.

exemple : chercher un littéral et son opposé, on l'enlève et on prend le reste
- res({**non(p)**, q, r}, {**p**, q, non(s)}, p) = {q, r, non(s)}
- res({**non(p)**, q, r}, {**p**, non(q), non(s)}, **p**) = {q, r, non(q), non(s)}
- res({non(p), **q**, r}, {p, **non(q)**, non(s)}, **q**) = {non(p), r, p, non(s)}

Logique : choses importantes

- res({**non(p)**}, {**p**}, **p**) = ensemble vide
- res({**p**, non(p), q}, {**non(p)**, non(t)}, p) = {non(p), q, non(t)}

Si plusieurs résolvantes peuvent être calculées à partir de deux clauses C et C’ alors ces résolvantes sont logiquement équivalentes et valides

- On se contentera donc de noter res(C,C’) la résolvante de deux clauses

La règle de résolution est une règle d’inférence (ie {C,C’} |= Res(C,C’))

La règle de résolution est 1 RI c'est à dire que

Si $\{C, C'\} \vdash_{Res} C'' = Res(C, C')$ alors $\{C, C'\} \models C''$

Preuve

1°) C et C' ne sont pas vides car sinon on n'aurait pas pu appliquer la règle de résolution (par ex. p et $\neg p$ dans C et C')

2°) Soit p le litt+ appartenant à C et $\neg p$ celui de C'

Montrons que pour tte int. I qui rend vrai C et C', elle rend vrai C''

- Si $I(p) = 0$ alors pour rendre vrai C dans I, il faut que $C \setminus \{p\}$ contienne 1 littéral l_c tel que $v(l_c, I) = 1$ [sém V]
  or $l_c \in C''$ donc, par la sém. V, $v(C'', I) = 1$
- Si $I(p) = 1$ —— C' dans I — $C' \setminus \{\neg p\}$ — $l_{c'}$ ——
  or $l_{c'} \in C''$ —— v — = 1

$C = (p \vee \dots l_c \dots)$ $\quad C' = (\neg p \vee \dots l_{c'} \dots)$ $\quad C'' = (\dots l_c \dots l_{c'} \dots)$

Définition d’une séquence de résolution :

Soit F une forme clausale et C une clause, on a F |= res C ssi il existe une séquence finie de clauses (C_1, C_2, ...C_r) avec :

- Cr = C
- et pour tout i=1 … r
- C_i est dans F

- où C_i est une résolvante de deux clauses avant Ci dans la séquence (ie il existe l < i et k < i C_i étant une résolvante de C_l et C_k)

Exemple de production de la clause vide :

Une forme F est insatisfiable ssi F|=(je peux trouver une dérivation) res ensemble vide.
Soit :
- C1={a,d} C4={a,e}
- C2={c, non(d)} C5={non(c),non(e)}
- C3={non(a), e} C6={non(a),d}

Dérivation 1 :
- C1,C2,{a,c},C5,,{a,non(e)},C4,{a},,C6,{d},C3,{e},{non(c)},{non(d)}, ensemble vide

---

Méthode de résolution :

Il est inutile de calculer les résolvantes des clauses contenant plusieurs littéraux opposés, car cela produit des clauses valides que l'on peut supprimer.

Soit le problème :
u, (w→u), (w→v), (t→v), (u→(w v t)) ⊨ v ?
ssi u^(w→u)^(w→v)^(t→v)^(u→(w v t))^¬v insatisfiable ?
ssi {{u},{u, ¬w},{v,¬w},{¬t,v},{t,¬u,w},{¬v}} insatisfiable ?
ssi {{u},{v,¬w},{¬t,v},{t,¬u,w},{¬v}} insatisfiable ?
ssi {{u},{v,¬w},{¬t,v},{t,¬u,w},{¬v}} |--res Ø ?

Résolution
– On a la dérivation : C1,C4,{t,w},C3,{v,w},C2,{v},C5,Ø
– Donc : u, (w→u), (w→v), (t→v), (u→(w v t)) |= v

Filtrage initial : on élimine les clauses tautologiques et les clauses qui contiennent une clause incluse
- On se dote d'une implémentation de la règle de résolution
  résolvable(c,c') vrai si deux clauses sont résolvables
  résolvante(c,c') qui retourne la clause résolvante de deux clauses

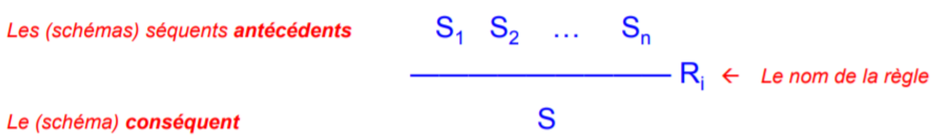
résolvables
- On met en œuvre une stratégie en largeur
- Soit E0 l'ensemble initial de clauses, on calcule E1 l'ensemble de clauses produites à partir des clauses de E0
- Puis E2 l'ensemble des clauses produites à partir d'une clause de E1 et d'une clause de E0 ∪ E1 (inutile de refaire les clauses de E0 entre-elles)
- A chaque étape, Ei est produit à partir d'une clause de Ei-1 et d'une clause de Ej avec j<i.
- Quand aucune nouvelle clause n'est produite on s'arrête

Méthode des séquents :

Deux aspects :
- sémantique : théorie des modèles, on définit par des interprétations les conditions de validité d'un raisonnement. $\vDash$
- Syntaxique : théorie de la démonstration, on définit par des règles d'inférence des pas élémentaires de raisonnement valide. $\vdash$

Théorie de la démonstration :
Objectif : formaliser les objets que manipulent les mathématiciens, les objets :
- Ils énoncent des assertions mathématiques qu'ils cherchent à démontrer.
- Ces conjectures portent sur une propriété P, qui bien souvent est énoncée sous certaines conditions, H1, … Hn
- On représente une telle conjecture par un séquent composé d'énoncés : H1, H2 … Hn $\vdash$ P

Les raisonnements :
- Ils réalisent des démonstrations en enchaînant des pas élémentaires de raisonnement permettant de passer d'un séquent à l'autre
- Ces pas élémentaires peuvent être schématisés en quelques types élémentaires de raisonnement formalisés par des règles d'inférence composées de séquents :



- Les séquents peuvent ne pas avoir d'hypothèses, ils représentent donc des propriétés assertées sans aucune condition (quand ils sont démontrés, on parle de tautologie) $\vdash$ P
- Les règles d'inférence peuvent n'avoir aucun antécédent, elles représentent donc des énoncés considérés comme toujours démontrés, on les appelle des axiomes. Ax H1, H2 … Hn $\vdash$ P -

Logique : choses importantes

- Un séquent que l'on peut produire (à partir des axiomes) en utilisant les règles d'inférence est appelé un théorème, ils représentent les conjectures démontrées. L'enchaînement des règles ayant permis leur production constitue une preuve de ce théorème.
- Remarque : tout séquent correspondant à un axiome est donc un théorème

Dans ce cadre, une théorie mathématique est composée :

- D'un langage formel définissant l'ensemble des énoncés (ou formules) syntaxiquement corrects
- D'un ensemble de règles d'inférences définies sur ces énoncés définissant l'ensemble des pas élémentaires de raisonnement admis dans cette théorie

Un système à la Hilbert (début XX^e) pour les fbf n'utilisant que ¬, →

- Un système de déduction dont les séquents n'ont pas d'antécédent composé de 4 règles (3 sont axiomes) dont les énoncés sont des fbf :

$$\frac{}{\vdash (P \to (Q \to P))}\ \text{Ax1} \qquad \frac{}{\vdash ((P \to (Q \to R)) \to ((P \to Q) \to (P \to R)))}\ \text{Ax2}$$

$$\frac{}{\vdash ((\neg P \to \neg Q) \to (Q \to P))}\ \text{Ax3} \qquad \frac{\vdash P \qquad \vdash (P \to Q)}{\vdash Q}\ \text{MP}$$

  - L'axiome 1 signifie : *soit P et Q deux formules bien formées quelconques de la LP, le séquent* $\vdash (P \to (Q \to P))$ *est un théorème.*
    - *Ex. :* $\vdash (a \to (a \to a))$ avec $P=Q=a$

Logique : choses importantes

Utilisation d'une règle d'inférence

- Une règle peut s'utiliser de deux manières :

$$\frac{\vdash P \qquad \vdash (P\to Q)}{\vdash Q}\ MP$$

1. En **analyse** (de bas en haut) : pour prouver la conclusion il faut prouver les antécédents
   - Pour la règle **M**odus **P**onens cela signifie : soit P et Q deux fbf quelconques de la LP, pour prouver le séquent $\vdash Q$ par utilisation de la règle MP, il faut prouver les séquents $\vdash P$ et $\vdash (P\to Q)$
   - Exemple : *pour prouver* $\vdash (a\to b)$ *par MP,*
     on doit avoir une preuve de $\vdash (a\to \neg c)$ *et* $\vdash ((a\to\neg c)\to(a\to b))$
     …ou n'importe quel autre couple de séquent de la forme $\vdash P$ *et* $\vdash (P\to(a\to b))$

2. En **synthèse** (de haut en bas) : si les antécédents sont prouvés alors la conclusion est prouvée.
   - Sur MP : si les séquents $\vdash P$ et $\vdash (P\to Q)$ sont des théorèmes alors le séquent $\vdash Q$ est un théorème.
   - Exemple : *les séquents* $\vdash (a\to (a\to a))$ *et*
     $\vdash ((a\to(a\to a)) \to ((a\to a)\to(a\to a)))$ *sont des théorèmes donc* $\vdash ((a\to a)\to(a\to a))$ *est un théorème*

Application :

- Montrons que $\vdash(a\to a)$ est un théorème
- Recherche d'une preuve :
- 1. $\vdash (a\to a)$ ne correspond pas à un schéma d'axiome
- 2. Donc la seule possibilité est d'utiliser MP
- 3. Il faut donc identifier une formule P et démontrer :
- 4. D'une part : $\vdash P$
- 5. Et d'autre part : $\vdash (P\to(a\to a))$

Propriété : Le système formel de Hilbert produit toutes les formules valides de la logique des propositions : pour toute fbf F : $\vdash F$ est un théorème ssi F est valide.

La déduction naturelle (G. Gentzen 1934)

1. Permettre la représentation d'énoncé sous hypothèses
   ➔ Les séquents peuvent avoir des hypothèses

2. Rendre ***naturelle*** l'exploitation des connecteurs logiques en donnant pour chaque connecteur des règles d'introduction et d'élimination des connecteurs
   - Exemple du connecteur →

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash (P\to Q)}\ \to_i \qquad\qquad \frac{\Gamma \vdash (P\to Q) \qquad \Gamma \vdash P}{\Gamma \vdash Q}\ \to_e$$

Ici les lettres latines représentent des fbf et les lettres grecques des ensembles de fbf

3. Un unique axiome représentant « une démonstration sous hypothèse triviale »

$$\frac{}{\Gamma, P \vdash P}\ ax$$

4. Pour des raisons techniques, il en propose une version plus symétriques (mais moins naturelle !) : le calcul des séquents. Ce système est très adapté à la démonstration automatique.

Logique : choses importantes

Le calcul des séquents ou Système LK

- Les séquents peuvent avoir plusieurs conclusions : H1, H2 … Hn ⊢ C1, C2 … Ck les énoncés hypothèses les énoncés conclusion

Attention :

- Un même énoncé peut être présent plusieurs fois dans l'hypothèse ou dans la conclusion
- Les énoncés de l'hypothèse sont interprétés comme une conjonction de formules ➔ Si n=0 alors cela correspond à une hypothèse toujours vraie
- Les énoncés de la conclusion sont interprétés comme une disjonction de formules ➔ Si k=0 alors cela correspond à une conclusion toujours fausse

Les règles du système LK

$$\frac{\Gamma \vdash \Delta, P}{\Gamma, \neg P \vdash \Delta}\ \neg_g \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash \Delta, \neg P}\ \neg_d$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta}\ \wedge_g \qquad \frac{\Gamma \vdash \Delta, P \qquad \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \wedge Q}\ \wedge_d$$

$$\frac{\Gamma, P \vdash \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \vee Q \vdash \Delta}\ \vee_g \qquad \frac{\Gamma \vdash \Delta, P, Q}{\Gamma \vdash \Delta, P \vee Q}\ \vee_d$$

Les règles du système LK

$$\frac{\Gamma \vdash \Delta, P \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \rightarrow Q \vdash \Delta}\ \rightarrow_g \qquad \frac{\Gamma, P \vdash \Delta, Q}{\Gamma \vdash \Delta, P \rightarrow Q}\ \rightarrow_d$$

$$\frac{\Gamma \vdash \Delta, P, Q \qquad \Gamma, P, Q \vdash \Delta}{\Gamma, P \leftrightarrow Q \vdash \Delta}\ \leftrightarrow_g \qquad \frac{\Gamma, P \vdash \Delta, Q \qquad \Gamma, Q \vdash \Delta, P}{\Gamma \vdash \Delta, P \leftrightarrow Q}\ \leftrightarrow_d$$

Logique : choses importantes

Propriété du système LK :
Théorème : Le système LK est adéquat et complet vis à vis de la sémantique de la logique des propositions :
– Il ne prouve que des conséquences logiques et les prouve toutes ! H1, H2… Hn ⊢ C1, C2… Ck est démontrable (prouvable) SSI { H1, H2, … Hn } ⊨ C1 ∨ C2 ∨ … ∨ Ck

F est valide
ssi
le séquent ⊢F est démontrable dans LK

F est insatisfiable
ssi
le séquent F⊢ est démontrable dans LK

{H1,H2…Hn} ⊨ **C**
ssi
le séquent H1,H2…Hn ⊢ C est démontrable dans LK

Feuille complémentaire de cours sur les calculs des séquents :

On a des séquents qui représentent des disjonctions de conclusion C1, C2, . . . Cn formulées sous une conjonction d'hypothèses H1, H2, . . . , Hk. Chaque énoncé, qu'il soit conclusion ou hypothèse, est une formule bien formée de la logique des propositions.

On représente un séquent en séparant les hypothèses et conclusions par le symbole ⊢.
On utilise également des schémas de séquent qui sont comme des filtres qui sélectionnent les séquents répondant à certains critères. En fait, un schéma de séquent défini un langage : l'ensemble de tous les séquents qui respectent ce schéma.

Ainsi, dans un schéma de séquent on a deux types de variables :
- les lettres grecques Γ (Gamma) ou Δ (Delta) qui peuvent être remplacée par n'importe quel ensemble (y compris l'ensemble vide) de formules bien formées de la logique des propositions,
- les lettres majuscules P ou Q qui peuvent être remplacées par n'importe quelle formule bien formée de la logique des propositions,
- les connecteurs et constantes logiques **ne peuvent pas** être remplacés.

Exemple le schéma $\Gamma \vdash \Delta, P \wedge Q$ reconnait les séquents :
1. $a \vdash b, c \wedge d$
2. $a, b \vee c \vdash a \wedge (e \rightarrow (c \vee a))$
3. $\vdash a \wedge b$
4. $a \wedge b \vdash c, d \rightarrow a, \neg c, a \wedge b$
5. $a \wedge b \vdash \neg c \wedge b, (b \rightarrow c) \wedge d, e \vee a$

Logique : choses importantes

Par contre le schéma ne reconnait pas les séquents :

1. $a \vdash b, c \vee d$
2. $a, b \vee c \vdash$
3. $b \wedge c \vdash a \to (c \wedge b)$

On dispose d'un ensemble de règles d'inférences qui permettent de dériver de nouveaux séquents à partir de séquents existants. Ces règles sont formulées en utilisant des schémas de séquent. Ces règles sont données sous la forme de "fraction" où la partie "numérateur" contient un ensemble de schémas de séquents, les séquents antécédents, hypothèses et la partie dénominateur "contient" un unique schéma de séquent appelé le conséquent.

Attention, lors de l'utilisation d'une règle, une même variable doit être remplacée par la même formule (ou le même ensemble de formules) dans tous les schémas de séquents de la règle.

Pour appliquer une règle à un séquent il faut que ce séquent soit reconnu par le conséquent de la règle. L'application consiste alors à produire les séquents obtenus à partir des antécédents de la règle en utilisant les mêmes remplacements de variables que pour le conséquent. Les règles s'appliquent donc de bas en haut (du dénominateur vers le numérateur).

Notation Il est d'usage de présenter cette application de règle par une fraction (étiquetée par le nom de la règle) dont le dénominateur contient le séquent sur lequel la règle est appliquée et le numérateur contient les séquents produits.

$$\frac{}{\Gamma, P \vdash \Delta, P}\ ax$$

$$\frac{}{\Gamma \vdash \Delta, \top}\ \top_{\mathsf{d}}$$

$$\frac{}{\Gamma, \bot \vdash \Delta}\ \bot_{\mathsf{g}}$$

$$\frac{\Gamma \vdash \Delta, P}{\Gamma, \neg P \vdash \Delta}\ \neg_{\mathsf{g}}$$

$$\frac{\Gamma, P \vdash \Delta}{\Gamma \vdash \Delta, \neg P}\ \neg_{\mathsf{d}}$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta}\ \wedge_{\mathsf{g}}$$

$$\frac{\Gamma \vdash \Delta, P \qquad \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \wedge Q}\ \wedge_{\mathsf{d}}$$

$$\frac{\Gamma, P \vdash \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \vee Q \vdash \Delta}\ \vee_{\mathsf{g}}$$

$$\frac{\Gamma \vdash \Delta, P, Q}{\Gamma \vdash \Delta, P \vee Q}\ \vee_{\mathsf{d}}$$

$$\frac{\Gamma \vdash \Delta, P \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \to Q \vdash \Delta}\ \to_{\mathsf{g}}$$

$$\frac{\Gamma, P \vdash \Delta, Q}{\Gamma \vdash \Delta, P \to Q}\ \to_{\mathsf{d}}$$

$$\frac{\Gamma \vdash \Delta, P, Q \qquad \Gamma, P, Q \vdash \Delta}{\Gamma, P \leftrightarrow Q \vdash \Delta}\ \leftrightarrow_{\mathsf{g}}$$

$$\frac{\Gamma, P \vdash \Delta, Q \qquad \Gamma, Q \vdash \Delta, P}{\Gamma \vdash \Delta, P \leftrightarrow Q}\ \leftrightarrow_{\mathsf{d}}$$

Logique : choses importantes

Une démonstration (ou preuve) d'un séquent consiste à enchaîner des applications de règles. Etant donné un ensemble E de séquents contenant initialement le séquent que l'on cherche à démontrer, on itère les pas suivants :

1. choisir un séquent S dans l'ensemble E ;
2. choisir une règle R pouvant s'appliquer sur ce séquent S ;
3. si aucune règle ne s'applique sur ce séquent alors Echec (le séquent initial n'admet pas de démonstration) ;
4. sinon (a) choisir une application de R sur S (cas où le schéma du conséquent de R permet différente reconnaissance de S) ;
   (b) supprimer le séquent S de E ;
   (c) ajouter à E les séquents produits par l'application de R sur S ;
5. si E est vide alors Succès (on a une démonstration du séquent initial) ;
6. sinon itérer à l'étape 1.

Notation Il est d'usage de présenter une telle démonstration par un arbre d'applications de règles (donc de fractions) dont la racine, que l'on place en bas, a pour dénominateur le séquent à démontrer et les feuilles (se trouvant en haut) ont des numérateurs vides (elles correspondent donc à des applications d'axiomes).

Exemple Voici une démonstration du séquent $a \vdash \neg(a \to b), b \wedge (c \vee a)$ obtenue, de bas en haut, en appliquant d'abord la règle $\neg_\mathsf{d}$, puis la règle $\to_\mathsf{g}$ et en appliquant la règle ax sur le séquent de gauche et les règles $\wedge_\mathsf{d}$ qui crée à nouveau 2 branches et ax pour l'un puis $\vee_\mathsf{d}$ et ax pour l'autre :

$$\dfrac{\dfrac{\dfrac{}{a \vdash a, b \wedge (c \vee a)}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{a, b \vdash b}\,\mathsf{ax} \quad \dfrac{\dfrac{}{a, b \vdash c, a}\,\mathsf{ax}}{a, b \vdash c \vee a}\,\vee_\mathsf{d}}{a, b \vdash b \wedge (c \vee a)}\,\wedge_\mathsf{d}}{a, a \to b \vdash b \wedge (c \vee a)}\,\to_\mathsf{g}}{a \vdash \neg(a \to b), b \wedge (c \vee a)}\,\neg_\mathsf{d}$$

Propriété (Adéquation). Si un séquent H1, H2, . . . , Hk ` C1, C2, . . . Cn est un théorème alors {H1, H2, . . . , Hk} |= C1 ∨ C2 ∨ . . . ∨ Cn.
Propriété (Complétude). Si {H1, H2, . . . , Hk} |= C1∨C2∨. . .∨Cn alors le séquent H1, H2, . . . , Hk ` C1, C2, . . . Cn est un théorème.

Le calcul des séquents :

Un même énoncé peut être présent plusieurs fois dans l'hypothèse ou dans la conclusion.
La déduction naturelle :
- permettre la représentation d'énoncés sous hypothèses.
- Rendre naturelle l'exploitation des connecteurs.

Dans les séquents les hypothèses sont séparées par des virgules. On peut avoir aucune hypothèse mais on doit forcément avoir un séquent. Les règles d'inférence sont des schémas de séquents. Pour faire une démonstration, on va utiliser des règles d'inférence.

Logique : choses importantes

Les règles axiomatiques n’ont rien au-dessus d’elles donc permettent de s’arrêter dans la démonstration.

Un théorème est une formule qui peut être mise en conclusion d”un séquent.
Si ma formule est valide alors on prouve. Les règles LK0 sont un peu à apprendre.

- Exemple du connecteur $\rightarrow$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash (P \rightarrow Q)} \rightarrow_i \qquad \frac{\Gamma \vdash (P \rightarrow Q) \quad \Gamma \vdash P}{\Gamma \vdash Q} \rightarrow_e$$

Ici les lettres latines représentent des fbf et les lettres grecques des ensembles de fbf