Logique de base
L2 informatique

S. Cerrito

2016, Université d'Evry

Plan du Cours

Merci à Francesco Belardinelli pour certaines améliorations de mes diapositives.

Informations

1h30 de cours, suivie par 1h30 de TD

serenacerrito@ibisc.univ-evry.fr

Références

- Les notes du cours.
- Livre :
  Serenella Cerrito, *Logique pour l'informatique Introduction à la déduction automatique* – Cours et exercices, Vuibert.

  Disponible à la B.U.

  NB : Contient plus que le cours.

Introduction : c'est quoi la logique ?

« Logique », du grec *logikê*, est un terme dérivé de *lógos* — signifiant à la fois « raison », «langage », et « raisonnement » — est, dans une première approche, l'étude des règles formelles que doit respecter toute argumentation correcte (Wikipedia).

Introduction

Logique par rapport à l'informatique, 2 rôles :

1. rôle **fondateur** :
   - La logique est une fondation mathématique pour définir proprement et traiter des concepts fondamentaux de l'informatique.
     exemple : définir proprement la notion de problème ayant une solution algorithmique
     $\Rightarrow$ cours Calculabilité et Complexité du L3, parcours CILS.
   - elle fournit également une bonne formation pour le raisonnement correct et précis, la description de concepts sans ambiguïté.
2. rôle **interne** : modéliser, spécifier, vérifier, raisonner automatiquement, fournir les bases de certains langages de programmation, etc.

Rôle interne : quelques exemples

La logique est nécessaire dans plusieurscours/domaine de la formation en informatique :

- Base de la vérification et de l'inférence de type en programmation fonctionnelle, et de la façon de calculer (CAML, par exemple ; paradigme : calculer = prouver).
- Base de la façon de calculer des langages de programmation logique (exemple : PROLOG, même paradigme).
- Fondement du langage de requêtes Calcul Relationnel dans les bases de données relationnelles $\Rightarrow$ cours SGBD du L3, tout parcours
- Base des systèmes de spécifications formelles « statiques » en génie logiciel , par exemple : preuve de programme, atélier B...
- Outil pour la modélisation, la spécification et vérification automatique du comportement dynamique de systèmes reactifs (logiques temporelles) $\Rightarrow$ cours Spécifications Formelles et Vérification du M1, plateforme CILS
- Base des certaines techniques de l'IA (en planification, en apprentissage automatique, etc.)

Donc, c'est quoi la logique ?

Un *Système Logique* se compose généralement de trois choses :

1. *Syntaxe* - un langage formel (comme pourun langage de programmation) qui sera utilisé pour exprimer (*formaliser*) des concepts
2. *Sémantique* - signification precise du langage formel
3. *Théorie de la preuve* - mécanismes purement syntaxiques d'obtenir ou d'identifier les énoncés « valides » du langage.
   Elle fournit un moyen d'argumenter dans le langage formel

Introduction (suite)

- Beaucoup plus que cela peut être fait avec la logique.
- La Logique a maintenant atteint la profondeur et la sophistication des plus hautes sphères des mathématiques modernes.
- Mais ceci est un premier cours de Logique, et nous resterons ancrés au sol !
- La logique propositionnelle et la logique du premier ordre sont le plus fondamentales de tous les systèmes logiques.
- Il faut commencer par celles la !

- Dans un *ensemble*, ni l'ordre des élements ni les répetitions d'un élement comptent :
$$\{1, 2, 9\} = \{9, 1, 2\} = \{9, 1, 2, 1\}$$
- Dans un *multi-ensemble*, l'ordre des élement ne compte pas mais les répétitions , si :
$$\{1, 2, 9\} = \{9, 1, 2\} \neq \{9, 1, 2, 1\}$$
- Dans une *liste*, autant l'ordre des élements que les répetitions d'un élement comptent :
$$[1, 2, 9] \neq [9, 1, 2] \neq [9, 1, 2, 1]$$
- Rappel des opérations sur les ensembles.

Définition (Relation)

Une relation à n arguments sur un ensemble E est un ensembles de n-uplets sur E, c.à.d., $R \subseteq E^n$.

Exemple

La relation $<$, qui est une relation à 2 arguments sur $\mathbb{N} = \{0, 1, 2, ...\}$: l'ensemble des couples $\langle n, m \rangle$ t.q. m est plus grand que n.
$\langle 0, 0 \rangle \in r$, $\langle 1, 2 \rangle \in r$, $\langle 2, 4 \rangle \in r$ *etc.*

Définition (Réfléxivité, Symétrie, Transitivité)

Une relation à 2 arguments R est dite :

- Refléxive *si* $\forall x \in E$, *alors* $(x, x) \in E$.
- Symétrique *si* $\forall x, y \in E$, *si* $(x, u) \in E$ *alors* $(y, x) \in E$ *aussi.*
- Transitive *si* $\forall x, y, z \in E$, *si* $(x, y) \in E$ *et* $(y, z) \in E$, *alors* $(x, z) \in E$ *aussi.*

Définition (Rélation d'équivalence)

Une relation d'équivalence est une relation à 2 arguments qui est réflexive, symétrique et transitive.

Exemple

La relation r ci-dessus n'est pas une équivalence, mais l'identité :

$$\{\langle n, n\rangle \mid n \in N\}$$

si.

Définition (Fonction (totale))

Une fonction (totale) $f : E_1 \to E_2$ *est une relation* <u>*binaire*</u> f *sur* $E_1 \times E_2$ *t.q., qque soit* $a \in E_1$, $\exists\, !\, b \in E_2$ *t.q.* $\langle a, b\rangle \in E_1 \times E_2$.

E_1 : *domaine* de f
E_2 *ensemble d'arrivé (ou co-domaine)* de f
On note $f(a)$ l'unique b t.q. $\langle a, b\rangle \in f$

Exemple

$<$ *n'est pas une fonction sur* $\mathbb{N} \times \mathbb{N}$, *mais*

$$\{\langle n, 2n\rangle \mid n \in \mathbb{N}\}$$

oui.

Définition (Fonctions)

Une fonction $f : E_1 \to E_2$ *est*

- injective *si* $f(a) = f(a')$ *implique* $a = a'$
- surjective *si, qque soit* $b \in E_2$, $\exists a \in E_1$ *t.q.* $b = f(a)$
- *une* bijection *si* f *est injective* **et** *surjective (correspondence 1-1).*

Définition

Un ensemble E est denombrable *s'il existe une bijection $f : E \to E'$ oú $E' \subseteq \mathbb{N}$.*

Exemple

- *tout ensemble fini est denombrable*
- *l'ensemble des nombres relatifs pairs est infini et denombrable*
- *l'ensemble des nombres réels n'est pas denombrable (Cantor)*
  Intuition *: on ne peut pas compter les éléments d'un ensemble non-denombrable de la façon : élément 0, élément 1, élément 2, etc.*

1. *Base de la Définition Récursive de E* : à l'étape e_0 on définit les éléments de E qui ont une structure élémentaire ;
2. *Etape de Récurrence de la Définition de E* : en supposant que des objets $o_0, o_1, \ldots$ ont été déjà définis en un nombre fini de pas à des étapes inférieures $e_0, \cdots, e_n$, on définit des nouveaux éléments de E - de structure plus complexe -, à partir des éléments anciens, à l'étape e_{n+1} immédiatement supérieure.
3. Finalement, on stipule que les *seuls* objets de E sont ceux ainsi engendrés (énoncé souvent implicit dans la suite).

Exemple (Définition par récurrence de $\mathbb{N}$)

1. Base de la Définition Récursive *: on a un seul objet élémentaire de* $\mathbb{N}$, *c.à.d.*, $0 \in \mathbb{N}$.
2. Etape de Récurrence de la Définition Récursive *: si* $n \in \mathbb{N}$, *alors le* successeur *de* n, *que l'on note* $s(n)$ *ou bien* $n+1$, *est aussi dans* $\mathbb{N}$.
3. Fin de la Définition Récursive *: rien d'autre est un élément de* $\mathbb{N}$.

NB : tout nombre $n \in \mathbb{N}$ (dit « nombre naturel ») est généré en exactement n étapes.

Exemple (Définition par récurrence de AR, un sous-ensemble des expressions arithmétiques)

1. Base de la Définition Récursive *: tout expression pour nommer un élément de* $\mathbb{N} = \{0, 1, 2, ...\}$ *(numéral) est un élément de AR.*
2. Etape de Récurrence de la Définition Récursive *:*
   - *si* $e \in AR$ *alors* $(- e) \in AR$.
   - *si* $e1, e2 \in AR$ *alors op* $(e1, e2) \in AR$, *pour op* $\in \{+, -, \times\}$
3. Fin de la Définition Récursive *: rien d'autre est un élément de AR*

Autre façon de présenter cette définition :

$$exp \;=\; num \;\mid\; (- exp) \;\mid\; op\,(exp, exp)$$

où $num \in \mathbb{N} = \{0, 1, 2, ...\}$ et $op \in \{+, -, \times\}$.

Pour prouver que tous les éléments d'un ensemble E défini par récurrence ont une propriété P :

1. On prouve que les objets élémentaires définis dans la base de la définition récursive de E ont bien la propriété P.
   Cette partie de la démonstration est dite *Base (de la démonstration Récursive)*.
2. L'étape suivante est dite *Cas de Récurrence (de la démonstration Récursive)*.
   On suppose que tous les objets construits (à partir des objets élémentaires) en un nombre quelconqueque d'étapes k, oú $k \leq p$ et $p \in \mathbb{N}$ ont la propriété P ; cette hypothèse est dite *Hypothèse de Récurrence* (notée *HR*).
   Puis, en utilisant *HR*, on montre que tous les nouveaux objets construits à l'étape $p+1$ à partir des anciens ont encore la propriété P.

Exemple

On prouve la propriéte P suivante des éléments de $\mathbb{N}$ *:*

Si $n > 0$, la somme $\sum(n)$ des n premiers éléments de $\mathbb{N} \setminus \{0\}$ est égale à : $\frac{n\times(n+1)}{2}$

1. Base
   Si n est généré en 0 étapes, alors $n = 0$.
   $\Rightarrow$ *P est banalement vraie, car on suppose* $n > 0$ *!*
2. Cas de Récurrence.
   On veut prouver que P est valable si n est généré en plus que 0 étapes, c'est-à-dire que $n > 0$.
   Si $n = 1$, *c'est banal, car* $s(n) = 1 = \frac{1\times2}{2}$.
   Sinon, $n \geq 2$. *Alors :*
   HR : qque soit $m \in N$ généré en un nombre d'étapes inférieur ou égal à $n - 1$, c.à.d. qque soit $m \leq (n - 1)$, si $m > 0$ alors $s(m) = \frac{m\times(m+1)}{2}$.
   Evidemment : $\sum(n) = \sum(n - 1) + n$. *Comme cas particulier de HR :* $\sum(n - 1) = \frac{(n-1)\times n}{2}$.
   Donc : $\sum(n) = (\frac{(n-1)\times n}{2}) + n = \frac{(n-1)\times n+2n}{2} = \frac{n\times(n+1)}{2}$, c.q.f.d.

Etude des tests conditionels
Considérez l'instruction suivante :

```
if count>0 and not found then
  decrement count; look for next entry;
end if
```

Caractéristiques :

- Descriptions des états de base (propositions "atomiques") - ici : `count> 0`, `found` - sont vrais ou faux selon les circonstances
- on peut utiliser des opérations booléennes - `and`, `or`, `not`, etc. - pour construire des énoncées de test plus complexes à partir des propositions atomiques
- l'énoncé complexe final est vrai ou faux

Toute personne qui a écrit des programmes impératifs connaît les tests conditionels. Pourquoi les étudier à nouveau ?

- Toutes les logiques sont basées sur la logique propositionnelle dans une certaine mésure. On doit commencer par ici.
- Nous voulons gérer des tests conditionels arbitrairement compliqués et étudier leur caractéristiques générales
- Nous ne voulons pas seulement évaluer les tests. Nous voulons aussi savoir quand deux tests signifient la même chose, quand un test implique un autre, si un test (ou un test de boucle) peut être jamais rendu vrai ou faux.
- La logique propositionnelle réprésente une classe importante de problèmes computationels. Elle apparaît dans les algorithmes et dans la théorie de la complexité.

Syntaxe de la logique propositionnelle (ou booléenne)

Le langage formel

But : Spécifier quelles expressions sont considérées “bien formées”, ou “légales” indépendamment de leur signification et de leur vérité.

Ces expressions constitueront le langage formel de la logique propositionnelle.

Syntaxe de la logique propositionnelle (ou booléenne)

Les propositions atomiques

- Nous ne sommes pas concernés par quels énoncés atomiques (`count>0`, `found`) sont utilisés, mais juste par le fait qu'ils peuvent avoir une valeur de vérité (vrai ou faux).
- Donc, nous fixons une collection de symboles qui réprésententeront ces énoncés atomiques.
- Ces symboles sont appelés *propositions atomiques*, ou *variables propositionnelles*, ou encore *variables booléennes*.
  Plus rapidement : *atomes*.
- Ils ont la même fonction que des variables x, y, z en mathématiques.
- Mais comme ils sont propositionnels, nous utilisons d'habitude les lettres p, p', p_0, p_1, p_2, ..., ainsi que q, r, s, ...

Alphabet de la Logique Propositionnelle

Définition (Alphabet)

Un alphabet $\mathcal{A}$ *d'un langage propositionnel* $\mathcal{L}_P$ *est constitué par les ensembles suivants :*

- *Un ensemble dénombrable* $P = \{p, q, r, ...\}$ *de symboles dits* lettres (ou variables) propositionnelles.
  Cet ensemble est aussi dit signature *de* $\mathcal{L}_P$.
- *Un ensemble de symboles dits* connecteurs logiques, *contenant deux connecteurs à 0 arguments* $\top$ *et* $\bot$, *aussi dits* constantes propositionnelles, *un connecteur à un argument* $\neg$ *et quatre connecteurs binaires :* $\wedge, \vee, \rightarrow, \leftrightarrow$.
- *Les parenthèses : une ouvrante,* (, *et une fermante,*).

Les Formules de la Logique Propositionnelle

Une formule propositionnelle est une séquence de symboles construite à partir des atomes propositionnels, en utilisant les connecteurs booléens ci-dessus, et les parenthèses, selon la façon appropriée.

Plus précisément :

Définition (Formules)

Les expressions bien formées ou formules *d'un langage propositionnel $\mathcal{L}_P$ sont les mots sur l'alphabet de $\mathcal{L}_P$ qui sont éléments de l'ensemble défini par récurrence comme suit.*

- *Toute proposition atomique $p \in P$ est une formule.*
  Aussi, les constantes propositionnelles $\top$ et $\bot$ sont des formules.
- *si F est une formule, alors $(\neg F)$ est aussi une formule ;*
- *si F_1 et F_2 sont des formules, alors $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \rightarrow F_2)$, et $(F_1 \leftrightarrow F_2)$ sont aussi des formules.*

Dit autrement :

$$F := p \mid \top \mid \bot \mid (\neg F) \mid (F * F)$$

où $p \in P$ et $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Les parenthèses son nécéssaires pour éviter le ambiguités, comme en arithmétique : $(2 + 3) \times 4$ n'a pas la même signification que $\overline{2 + (3 \times 4)}$.

Arbre Syntaxique

Une formule bien parenthésée se lit d'une façon unique, indiquée par son *arbre syntaxique* :

Définition (Arbre Syntaxique)

*L'*arbre syntaxique *d'une formule A, noté $Arbre_A$, est un arbre binaire $\langle G, R, D\rangle$ défini par récurrence sur la structure de A :*

- *Si A est atomique, son arbre syntaxique est $\langle \emptyset, A, \emptyset\rangle$.*
- *Si A est $(\neg A_1)$, alors son arbre syntaxique est $\langle Arbre_{A_1}, \neg, \emptyset\rangle$, où $Arbre_{A_1}$ est l'arbre syntaxique de A_1.*
- *Si A est $(A_1 \ * \ A_2)$ où $*$ est un connecteur binaire, alors son arbre syntaxique est $\langle Arbre_{A_1}, *, Arbre_{A_2}\rangle$, où $Arbre_{A_1}$ est l'arbre syntaxique de A_1 et $Arbre_{A_2}$ est l'arbre syntaxique de A_2.*

Exemples de formules et de leur arbres syntaxiques

Au tableau

Connecteur Principal

Définition (Connecteur Principal)

Si A contient au moins un connecteur, on appelle connecteur principal *de A le connecteur qui se trouve à la racine de* $Arbre_A$

Chaque formule non-atomique a un connecteur principal qui détermine sa *forme logique globale*. Vous devez apprendre à la reconnaître.

- $(p \wedge q) \rightarrow r$ a connecteur principal $\rightarrow$. Sa forme logique globale est $A \rightarrow B$.
- $\neg(p \rightarrow (\neg q))$ a connecteur principal $\neg$. Sa forme logique est $\neg A$.
- $(p \wedge q) \wedge r$ a connecteur principal $\wedge$ (le 2ème !). Sa forme logique est $A \wedge B$.
- $p \vee (q \wedge r)$ a connecteur principal $\vee$. Sa forme logique est $A \vee B$.

Quels sont les arbres syntaxiques de ces formules ?

Encore sur les parenthèses

Exemple de démonstration par récurrence sur l'ensemble des formules d'un langage

Propriété : une formule a un nombre pair de parenthèses.

Preuve au tableau du fait que toute formule a cette propriété.

Conventions pour simplifier l'écriture des formules

Les parenthèses éliminent l'ambiguité, mais trop de parenthéses, c'est lourd. On introduit des conventions, que l'on pourra utiliser pour abréger.

- Pas d'écriture des parenthèses les plus extérieures.
- Le connecteur $\neg$ s'applique à la plus petite formule le suivant immédiatement ; par ex. on pourra écrire $\neg p \wedge q$ à la place de $(\neg p) \wedge q$;
- $\wedge$ et $\vee$ sont prioritaires sur $\rightarrow$ et $\leftrightarrow$; par ex., on pourra écrire $p \wedge q \rightarrow r \vee p$ à la place de $(p \wedge q) \rightarrow (r \vee p)$ (analogie avec $\times$ par rapport à $+$ en arithmetique).
- si * est un connecteur binaire donné et $A_1, ..., A_n$, oú $n \geq 3$, sont des formules, on pourra écrire $A_1 * A_2 ... * A_n$ à la place de $(...(A_1 * A_2)...) * A_n$ (association à gauche).

Plus de conventions pour simplifier l'écriture des formules

Pour eliminer plus de parenthèses, on peut aussi classer tous les connecteurs booléens selon leur force de liaison décroissante :

$$\text{(plus fort)} \neg, \wedge, \vee, \rightarrow, \leftrightarrow \text{ (plus faible)}$$

Comme dans l'arithmétique, où $\times$ est plus forte que $+$.
$2 + 3 \times 4$ est généralement lu comme $2 + (3 \times 4)$, pas comme $(2 + 3) \times 4$.

Alors :

- $\neg p \rightarrow q$ est lue comme $(\neg p) \rightarrow q$, pas comme $\neg(p \rightarrow q)$.
- $p \vee q \wedge r$ est lue comme $p \vee (q \wedge r)$, pas comme $(p \vee q) \wedge r$.
- $p \wedge \neg q \rightarrow r$ est lue comme $(p \wedge (\neg q)) \rightarrow r$, plutôt que $p \wedge (\neg(q \rightarrow r))$ ou $p \wedge ((\neg q) \rightarrow r)$.
- $p \rightarrow q \leftrightarrow r$ est lue comme $(p \rightarrow (q \leftrightarrow r)$, pas comme $(p \rightarrow q) \leftrightarrow r$.

Mais n'abusez pas de ce classement.

Par exemple, $p \rightarrow \neg q \vee \neg\neg r \wedge s \leftrightarrow t$ est dur à lire ! Dur de voir qu' on abrège :
$(p \rightarrow (\neg q \vee (\neg\neg r \wedge s))) \leftrightarrow t$.

Mieux d'utiliser juste les conventions de la page précédente.

Sous-formules

Les sous-formules d'une formule A sont les formules construites dans les étapes de construction de A.

Elles correspondent à des noeuds, ou à des sous-arbres, de l'arbre syntaxique de A.

Définition (Ensemble *sousf*(A) des sous-formulesd'une formule A)

- *Si A est une formule atomique, alors sousf*(A) *est* $\{A\}$;
- *Si le connecteur principal de A est $\neg$, c'est-à-dire que A a la forme $\neg A_1$, alors sousf*(A) *est l'union ensembliste de* $\{\neg A_1\}$ *et de sousf*(A_1)
- *Si le connecteur principal de A est un connecteur binaire *, c'est-à-dire que A a la forme $A_1 * A_2$, alors sousf*(A) *est l'union ensembliste de* $\{A_1 * A_2\}$, *avec sousf*$(A)_1$ *et sousf*$(A)_2$.

La définition est par récurrence, en suivant la définition par récurrence des formules (cas de base : les formules atomiques).

Termes techniques pour les formules logiques

Un peu de jargon utile.

Définition

- *Une formule de la forme* $\top$, $\bot$, *ou* p, *pour un atome* p, *est appelée* atomique.
- *Une formule dont la forme logique est* $\neg A$ *est appelée une formule* niée.
- *Une formule de la forme* $A \wedge B$ *est appelée une* conjonction, *et* A, B *sont ses* conjoints.
- *Une formule de la forme* $A \vee B$ *est appelée une* disjonction, *et* A, B *sont ses* disjoints.
- *Une formule de la forme* $A \to B$ *s'appelle une* implication. A *est appelée l'*antécédente, B *est appelée* conséquente.

Termes techniques pour les formules logiques (suite)

Définition

- *Une formule qui est soit atomique soit la négation d'une formule atomique est appelée* littéral.
- *Une* clause *est une disjonction ($\vee$) d'un ou plusieurs littéraux.*

Par exemple, le formules p, $\neg r$, $\neg\bot$, $\top$ sont tous des littéraux.

Exemples de clauses (et de non-clauses) au tableau.

Parfois on considère la clause vide (la disjonction de zéro littéraux !), et on la traite de la même manière que $\bot$.

Sémantique – signification

- Nous savons comment lire et écrire les formulas. Mais quelle est leur signification ?
- Ou de manière plus chic : quelle est leur *sémantique* ?
- Signification intuitive des connecteurs booléens : $\neg$ est « non », $\wedge$ est « et », $\vee$, est « ou », $\rightarrow$ est « si-alors » (à peu près : voir après) , et, enfin, $\leftrightarrow$ est « si et seulement si ».
- Mais le français est un langage naturel : plein d'ambigüités.
- Nous sommes des informaticien(ne)s. On a besoin d'une signification précise des formules – qui sont en nombre infini.
- Donc, on préfère donner une définition formelle en style mathématique de la sémantique.

Sémantique de la logique propositionnelle

But : donner une signification aux connecteurs logiques, et d'*interpréter* les variables propositionnelles, en leur attribuant une valeure booléenne (*vraie* ou *fausse*), afin de permettre de déterminer la vérité d'une formule complexe à partir de celle des formules atomiques qui la composent.

Donner de la signification aux formules

Définition (Interprétation)

Une **interprétation** $\mathcal{I}$ *pour un ensemble de variables propositionnelles* P *est une fonction* $\mathcal{I} : P \to BOOL$, *ou* $BOOL = \{V, F\}$.

- Pour les tests conditionnels une interprétation est un point dans l'execution d'un programme.
  Les valeurs courantes du programme déterminent si chaque énoncé atomique d'un test conditionnel (par exemple, $x > 0$, $x = y$, etc.) est vrai ou faux.
- En général, pour des atomes p, q, ...,
  une interprétation spécifie juste quels atomes sont vrais et quels sont faux.
- *N'oubliez pas* : on peut avoir plusieurs interprétations différentes. Dans une autre interprétation les valeurs de vérité peuvent être différentes
- Evidemment, si on connait l'interprétation, on peut dériver la signification de chaque formule propositionnelle donnée – c.à.d., si elle est vraie ou fausse dans l'interprétation.
- On procède des formules simples aux formules plus compléxes $\leadsto$ définition de la valeur de vérité d'une formule A par récurrence sur les formules.

Déterminer la Valeur de Vérité

La valeur de vérité d'une formule propositionnelle pour une interprétation donnée $\mathcal{I}$ est définie comme il suit :

- Cas de atomes :
  - $\top$ est vrai et $\bot$ est faux, peu importe qui est $\mathcal{I}$;
  - L'interprétation nous dit directement la valeur de vérité par rapport à $\mathcal{I}$ des atomes p, q,...
- Supposons que A et B sont des formules et qu'on a déjà établi si elles sont vraies ou fausses pour l'interprétation donnée $\mathcal{I}$. Alors, pour cette interprétation :
  - $\neg A$ est vraie si A est fausse, et fausse si A est vraie
  - $A \wedge B$ est vrai si A et B sont toutes les deux vraies, si non elle est fausse.
  - $A \vee B$ est vrai si l'une des A ou B est vraie, si non elle est fausse (ou inclusif)
  - $A \rightarrow B$ est vrai si A est fausse ou B est vraie (ou les deux), si non elle est fausse ("si A est vraie, alors B est vraie aussi")
  - $A \leftrightarrow B$ est vrai si A et B ont la même valeur de vérité (toutes les deux vraies ou toutes les deux fausses), si non elle est fausse

Fonctions boolénnes

Notation : si E est un ensemble et $n \in \mathbb{N}, n > 0$, on note E^n l'ensemble de toutes les n-uplets de E.

Pour dire la même chose qu'avant, mais avec un style plus mathématique, on utilise la notion de *fonction booléenne* :

Définition (Fonction booléenne)

Soit $BOOL = \{V, F\}$ l'ensemble dont les éléments sont V (Vrai) et F (Faux)(ou 1 et 0).
Une **fonction booléenne** *à n arguments (dite aussi "fonction de vérité) est une fonction $f : BOOL^n \rightarrow BOOL$.*

Fonction booléennes, suite

Définition

Les fonctions booléennes $f_\neg$, $f_\wedge$, $f_\vee$, $f_\rightarrow$, $f_\leftrightarrow$ sont définies par les tables suivantes, où l'argument de la fonction unaire $f_\neg$ est indiqué dans la première colonne :

Argument 1	*Argument 2*	$f_\neg$	$f_\vee$	$f_\wedge$	$f_\rightarrow$	$f_\leftrightarrow$
V	V	F	V	V	V	V
V	F	F	V	F	F	F
F	V	V	V	F	V	F
F	F	V	F	F	V	V

Quelle est la différence entre un connecteur c et la fonction booléen f_c ?

Sémantique de la logique propositionnelle pqr le biais de fonctions booléennes

Définition (Valeur de vérité)

La **valeur de vérité** *d'une formule A par rapport à une interprétation $\mathcal{I}$, notée $[A]_{\mathcal{I}}$, est définie par récurrence sur la structure de A :*

- *Si A est une lettre propositionnelle, $[A]_{\mathcal{I}} = \mathcal{I}(A)$;*
  de plus, $[\top]_{\mathcal{I}} = V$ et $[\bot]_{\mathcal{I}} = F$;
- $[\neg A_1]_{\mathcal{I}} = f_{\neg}([A_1]_{\mathcal{I}})$
- $[A_1 \ * \ A_2]_{\mathcal{I}} = f_*([A_1]_{\mathcal{I}}, [A_2]_{\mathcal{I}})$
  où $*$ est un connecteur binaire et f_* est la fonction booléenne associée.

Tables de vérité

L' évaluation d'une formule par rapport à une interprétation $\mathcal{I}$ donnée peut être représentée sous la forme d'une ligne d'une **table de vérité** :

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

Ici, les 2 premières colonnes indiquent l'interprétation $\mathcal{I}$ par rapport à laquelle on fait l'évaluation.

Sémantique de la logique propositionnelle : terminologie

Définition (Modèle)

Une interprétation $\mathcal{I}$ est un **modèle** *d'une formule A si $[A]_{\mathcal{I}} = V$.*

On dit que $\mathcal{I}$ est un **modèle** d'un ensemble de formules E si elle est modèle de tout élément de E.

N.B. : si E est un ensemble fini $\{A_1, \cdots, A_n\}$, alors $\mathcal{I}$ est un modèle de E ssi c'est un modèle de $(\ldots(A_1 \wedge A_2)\ldots) \wedge A_n$

Sémantique de la logique propositionnelle : terminologie, suite

Définition (Satisfiabilité)

Une formule A est **satisfiable** *s'il existe au moins une interprétation $\mathcal{I}$ qui est modèle de A ; sinon, A est dite* **insatisfiable**.

- Définition semblable pour un ensemble E de formules.
- *N.B.* Une formule A est satisfiable ssi pour au moins une ligne de sa table de verité la valeur de A est V.
- Complexité dans le pire des cas du test de satisfiabilité pour A, en fonction du nombre de lettres propositionnelles de A ?

Sémantique de la logique propositionnelle : terminologie, suite

Définition (Validité)

Une formule A est **valide** *(ou est une tautologie) si, pour n'importe quelle interprétation $\mathcal{I}$, $\mathcal{I}$ est un modèle de A.*

- *N.B.* A est valide ssi toute ligne de sa table de verité donne V comme valeur de A.
- Complexité du test de validité de A, en fonction du nombre de lettres propositionnelles de A ?

Exemples

Exemples de formules valides, insaisfiables, satisfiables mais pas valides : au tableau

Sémantique de la logique propositionnelle

N.B. Si A est une formule, p_i une de ses lettres propositionnelles, et A' la formule obtenue à partir de A en remplaçant systématiquement p_i par une formule quelconque B, alors la validité de A implique celle de A'.
Attention : l'implication reciproque est fausse !

Exemple

- A_1 *est* $p \vee \neg p$ *(valide).*
  En remplaçant p *par* $q \rightarrow r$ *on obtient* A'_1 *:* $(q \rightarrow r) \vee \neg(q \rightarrow r)$.
  Puisque A_1 *est valide* A'_1 *l'est aussi.*
- A_2 *est* $p \vee q$ *(pas valide).*
  En remplaçant p *par* $\neg q$ *on obtient* A'_2 *:*$\neg q \vee q$ *(valide).*
  A'_2 *est valide, mais* A_2 *ne l'est pas.*

Sémantique de la logique propositionnelle

Définition (Conséquence logique)

Soit E un ensemble quelconque de formules, soit A une formule.
On dit que A est **conséquence logique** *de E, et on note $E \models A$, si tout modèle de E est aussi un modèle de A.*

Et si $E = \emptyset$?

Analyse d'un raisonnement en langage naturel I

Je vous paierai pour votre réparation de mon PC seulement si mon PC marche.
Or, mon PC ne marche pas.
Donc, je ne vous paierai pas.

La conclusion *je ne vous paierai pas* suit des premisses du raisonnement ?

Formalisons : p pour *je vous paierai pour votre réparation de mon PC*, m pour *mon PC marche*.

Premisses : $p \to m$, $\neg m$
Conclusion : $\neg p$

Est-il vrai que $p \to m, \neg m \models \neg p$?

Analyse d'un raisonnement en langage naturel I

Je vous paierai pour votre réparation de mon PC seulement si mon PC marche.
Or, mon PC ne marche pas.
Donc, je ne vous paierai pas.

La conclusion *je ne vous paierai pas* suit des premisses du raisonnement ?

Formalisons : p pour *je vous paierai pour votre réparation de mon PC*, m pour *mon PC marche*.

Premisses : $p \rightarrow m$, $\neg m$
Conclusion : $\neg p$

Est-il vrai que $p \rightarrow m, \neg m \models \neg p$? **OUI**
La conclusion suit logiquement, raisonnement correct.

Comment tester si $p \rightarrow m, \neg m \models \neg p$?
En testant si $(p \rightarrow m, \wedge \neg m) \rightarrow \neg p$ est valide !

Analyse d'un raisonnement en langage naturel II

S'il pleut, alors je prends mon parapluie.
Mais, il ne pleut pas.
Donc, je ne prends pas le parapluie.

La conclusion *je ne prends pas le parapluie* suit des premisses du raisonnement ?

Formalisons : q pour *il pleut*, r pour *je prends le parapluie*.

Premisses : $q \to r$, $\neg q$
Conclusion : $\neg r$

Est-il vrai que $q \to r, \neg q \models \neg r$?

Analyse d'un raisonnement en langage naturel II

S'il pleut, alors je prends mon parapluie.
Mais, il ne pleut pas.
Donc, je ne prends pas le parapluie.

La conclusion *je ne prends pas le parapluie* suit des premisses du raisonnement ?

Formalisons : q pour *il pleut*, r pour *je prends le parapluie*.

Premisses : $q \to r$, $\neg q$
Conclusion : $\neg r$

Est-il vrai que $q \to r, \neg q \models \neg r$? **NO**
La conclusion ne suit pas logiquement, raisonnement incorrect.

La formule $((q \to r) \wedge \neg q) \to \neg r$ n'est pas valide.

Résultats Fondamentaux

1. $A_1, \cdots, A_n \models A$ si et seulement si la formule $(A_1 \wedge \cdots \wedge A_n) \rightarrow A$ est valide.
2. Si E est un ensemble de formules insatisfiable alors A est une conséquence logique de E.
3. Quel que soit l'ensemble de formules E, si A est valide alors A est une conséquence logique de E.
4. Quel que soit l'ensemble de formules E, $E \models A$ si et seulement si $E \cup \{\neg A\}$ est insatisfiable.
   Donc, en considérant le cas où E est vide, A est valide si et seulement si $\neg A$ est insatisfiable.

NB : *Propriété* **??** *très utile dans la suite.*

Formules Equivalentes

Définition (Equivalence)

Soient A_1 et A_2 deux formules.
On dit que A_1 et A_2 sont **logiquement équivalentes** *(et on note $A_1 \equiv A_2$) si A_1 est une conséquence logique de A_2 et A_2 est une conséquence logique de A_1, c.à.d., $A_2 \models A_1$ et $A_1 \models A_2$.*

N.B. $A_1 \equiv A_2$ si et seulement si la formule composée $A_1 \leftrightarrow A_2$ est valide, ou, de façon équivalente, si et seulement si A_1 et A_2 reçoivent la même valeur de vérité pour toute interprétation.

- quelle est la différence entre $\rightarrow$ et $\models$?
- quelle est la différence entre $\leftrightarrow$ et $\equiv$?
- quel est le rapport entre la notion spécifique d'*équivalence logique* et la notion générale de relation d'équivalence ?

Equivalences utiles

Qques soient les formules A_1, A_2 et A_3 :

- $A \vee \neg A \equiv \top$ (loi du tiers exclus)
- $A \leftrightarrow A_2 \equiv (A \rightarrow A_2) \wedge (A_2 \rightarrow A)$ (définissabilité de $\leftrightarrow$ par $\wedge, \rightarrow$)
- $A \rightarrow A_2 \equiv \neg A \vee A_2$ (définissabilité de $\rightarrow$ par $\neg, \vee$)
- $\neg(A \wedge A_2) \equiv \neg A \vee \neg A_2$ et $\neg(A \vee A_2) \equiv \neg A \wedge \neg A_2$ (lois de De Morgan)
- $\neg\neg A \equiv A$ (loi de double négation)
- $A \vee (A_2 \wedge A_3) \equiv (A \vee A_2) \wedge (A \vee A_3)$ et
  $A \wedge (A_2 \vee A_3) \equiv (A \wedge A_2) \vee (A \wedge A_3)$ (lois de distributivité)
- $A \wedge A_2 \equiv A_2 \wedge A$ et $A \vee A_2 \equiv A_2 \vee A$ (commutativité de $\wedge$, respectivement $\vee$)
- $A \vee (A_2 \vee A_3) \equiv (A \vee A_2) \vee A_3$ et
  $A \wedge (A_2 \wedge A_3) \equiv (A \wedge A_2) \wedge A_3$ (associativité de $\wedge$, respectivement $\vee$)
- $A \vee \bot \equiv A$ et $A \wedge \top \equiv A$ (éléments neutres)
- $A \wedge \bot \equiv \bot$ et $A \vee \top \equiv \top$
- $(A \wedge B) \vee A \equiv A$ et $(A \vee B) \wedge A \equiv A$

Remplacement

Théorème

Soient A une formule, B une sous-formule de A, et C une formule telle que $B \equiv C$. Si on remplace une occurrence de B dans A par C, on obtient une formule A' telle que $A \equiv A'$.

Exemple de l'utilité de ce théorème

$$\begin{aligned}((p \to (q \to p)) \land \neg\neg r) \lor (p \land \neg\neg\neg p) &\equiv (\top \land r) \lor (p \land \neg p) \\ &\equiv (r \land \top) \lor \bot \\ &\equiv r \lor \bot \\ &\equiv r\end{aligned}$$

Possibilité de récrire une formule en une autre logiquement équivalente mais + simple ou ayant une forme donnée (forme *normale*).

Plus sur les fonctions Booléennes

Outre $f_\vee$, $f_\wedge$, $f_\rightarrow$ et $f_\leftrightarrow$, d'autres fonctions booléennes à 2 arguments existent. Combien ?

Des fonctions booléennes à + que 2 arguments existent, par exemple celle définie par la table :

Argument 1	Argument 2	Argument 3	Valeur
V	V	V	V
V	V	F	F
V	F	V	F
V	F	F	V
F	V	V	F
F	V	F	V
F	F	V	V
F	F	F	F

Fonctions Booléennes

Est-il possible d'exprimer toute fonction booléenne à l'aide de $\top, \bot, \neg, \vee, \wedge, \rightarrow, \leftrightarrow$ *?*

Définition

Soit A une formule dont les lettres propositionnelles sont $p_1, \cdots, p_n$.
Soit f_A la fonction booléenne à n arguments telle que, quelque soit $v_i \in Bool$ et $\mathcal{I}$ telle que $\mathcal{I}(p_i) = v_i$, on a $f_A(v_1, \cdots, v_n) = [A]_{\mathcal{I}}$.
On dit alors que la fonction f_A **est réalisée** *par la formule A, ou, de façon équivalente, que A réalise f_A.*

Exemple

La formule $A = \neg\neg\neg p_1$ réalise la fonction $f_\neg$. Pourquoi ?
Quelles autres formules réalisent $f_\neg$?

Réformulation de la question :
Est-il possible de réaliser toute fonction booléenne en utilisant des formules dont les seuls connecteurs sont : $\top, \bot, \neg, \vee, \wedge, \rightarrow, \leftrightarrow$ *?*

Fonctions Booléennes

Est-il possible d'exprimer toute fonction booléenne à l'aide de $\top, \bot, \neg, \vee, \wedge, \rightarrow, \leftrightarrow$ *?*

Définition

Soit A une formule dont les lettres propositionnelles sont $p_1, \cdots, p_n$.
Soit f_A la fonction booléenne à n arguments telle que, quelque soit $v_i \in Bool$ et $\mathcal{I}$ telle que $\mathcal{I}(p_i) = v_i$, on a $f_A(v_1, \cdots, v_n) = [A]_{\mathcal{I}}$.
On dit alors que la fonction f_A **est réalisée** *par la formule A, ou, de façon équivalente, que A réalise f_A.*

Exemple

La formule $A = \neg\neg\neg p_1$ réalise la fonction $f_\neg$. Pourquoi ?
Quelles autres formules réalisent $f_\neg$?

Réformulation de la question :
Est-il possible de réaliser toute fonction booléenne en utilisant des formules dont les seuls connecteurs sont : $\top, \bot, \neg, \vee, \wedge, \rightarrow, \leftrightarrow$ *?* **OUI**
Si la fonction a au moins 1 argument, même $\neg, \vee, \wedge$ suffisent.

Fonctions Booléennes

Idée de la démonstration

Argument 1	Argument 2	Argument 3	Valeur
V	V	V	V
V	V	F	F
V	F	V	F
V	F	F	V
F	V	V	F
F	V	F	V
F	F	V	V
F	F	F	F

Les lignes avec résultat V guident la construction d'une formule réalisant la fonction. Cette formule est une disjonction de 4 sousformules : l_1, l_2, l_3 et l_4 :

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

Récriture d'une formule dans une forme normale logiquement équivalente

Conventions préliminaires.

1. Si $n = 0$, la notation $A_1 \vee \cdots \vee A_n$, n'indique pas vraiment une formule, toutefois on pose qu'elle indique la formule $\bot$.
   De même, si $n = 0$ la notation $A_1 \wedge \cdots \wedge A_n$ indiquera $\top$.
2. Quand on parlera de formes normales, on écrira $A_1 \vee ... \vee A_n$ pour indiquer une disjonction dont les composantes sont $A_1, ...A_n$, peu importe leur ordre et peu importe leur groupement par les ().
   Idem pour $A_1 \wedge \cdots \wedge A_n$.

Pourquoi ces conventions sont sémantiquement justifiées ?

Utilité de la récriture en formes normales : les systèmes formels de preuve que nous verrons après la nécessitent.

Récriture d'une formule dans une forme normale logiquement équivalente

Définition (Forme Normale de Négation)

Une formule est dite en **forme normale de négation** *(***f.n.n.***) si toute occurrence du connecteur* $\neg$ *s'applique à des sous-formules atomiques.*

Pour récrire A en une A' t.q. A' est en f.n.n. et $A \equiv A'$ utilisez :

- $A \leftrightarrow B \quad \equiv \quad (A \rightarrow B) \wedge (B \rightarrow A)$
- $A \rightarrow B \quad \equiv \quad \neg A \vee B$
- Les lois de De Morgan
- La loi de Double Négation

Récriture d'une formule dans une forme normale logiquement équivalente

Définition

- rappel : *un* **littéral** *est ou bien une formule atomique ou bien la négation d'une formule atomique.*
- rappel : *une* **clause** *(dite aussi : disjonction élémentaire) est une formule de la forme* $l_1 \vee l_2 \vee \cdots \vee l_{m-1} \vee l_m$, *où* $m \geq 0$ *et chaque* l_i *est un littéral.*
  Le littéral $\bot$ *est dit* clause vide.
- *Une formule A est dite en* **forme normale conjonctive (f.n.c.)** *si elle est de la forme* $D_1 \wedge \cdots \wedge D_k$, *où* $k \geq 0$ *et chaque* D_i *est une clause.*
- *une* **conjonction élémentaire** *est une formule de la forme* $l_1 \wedge l_2 \cdots \wedge l_{n-1} \wedge l_n$ *où* $n \geq 0$.
  Le littéral $\top$ *est dit* **conjonction élémentaire vide**.
- *une formule A est dite en* **forme normale disjonctive (f.n.d.)** *si elle est de la forme* $C_1 \vee \cdots \vee C_q$, *où* $q \geq 0$ *et chaque* C_i *est une conjonction élémentaire.*

Rappell : puisque $f_\wedge$ et $f_\vee$ sont commutatives et associatives, abus de syntaxe : nous identifierons par *exemple*, les deux formules en f.n.c.

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

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

Récriture d'une formule dans une forme normale logiquement équivalente

Théorème

Etant donnée une formule quelconque A :

- *il existe une formule A' qui est en f.n.c. et qui est logiquement équivalente à A ;*
- *il existe une formule A'' qui est en f.n.d. et qui est logiquement équivalente à A.*

Plusieurs façons de le prouver ; une en TD.

Dévinette (rédiger la solution pour le prochain TD)

Un étranger arrive au pays Byzarre, où on a seulement 2 sortes d'habitants : les *Hônnetes* qui disent toujours la vérité et les *Malhônnetes* qui mentent toujours.
Il se trouve à une bifurcation et s'interroge sur quelle route porte à la mairie : celle de droite ou celle de gauche ?
Un habitant de Byzarre est la, mais l'étranger ne sait pas á quelle sorte de personne il appartient.

Quelle question (une seule !) l'étranger peut poser à l'habitant pour savoir à coûp sûr quelle rue prendre pour se rendre à la mairie ?

Indication : l'étranger pourra poser la question : "A est vraie ?", où A est une formule booléennes dont les 2 (sous)formules atomiques sont

1. `la rue de gauche porte à la mairie`
2. `vous étes hônnete`

Systèmes formels de preuve

Systèmes formels de preuve

But : avoir des moyens purement *syntaxiques*, c'est-à-dire ne tenant compte que de la forme des formules, pour montrer qu'une formule donnée est valide.

Une *preuve* (formelle) sera un arbre d'expressions.

Systèmes formels de preuve

Plusieurs sortes de systèmes de preuve existent. Une classification partielle, selon les critéres :

- **D**irect ?
- **A**utomatisable ?
- nécéssite une **F**orme **N**ormale ?
- **C**oncis ?

Nom	**D** ?	**A** ?	**FN** ?	**C** ?
Axiomatiques (Hilbert)	Oui	Non	Non	Oui
Séquents (Gentzen)	Oui	Oui	Non	Non
Déduction Naturelle (Gentzen)	Oui	en partie	Non	Non
Tableaux (Beth, Gentzen)	Non	Oui !	"Oui et Non"	Non
Résolution (Robinson)	Non	Oui (si stratégie)	Oui	Oui !

Dans ce cours : *Tableaux*, *Résolution*.

Tableaux propositionnels

Les tableaux ne sont pas des systèmes de preuve directs, mais des systèmes de *réfutation* : pour prouver A, on prouve que l'hypothèse $\neg A$ mene à une contradiction.

Il en existe de plusieurs sortes.

Ici :

Définition (Tableau)

Un **tableau** *est un arbre t.q.*

1. *Ses noeuds sont des ensembles de formules en f.n.n. ;*
2. *Les fils d'un noeud sont générés en lui appliquant une règle d'expansion.*

En principe, l'arbre peut être infini.

Règle d'expansion = ?

Tableaux propositionnels

On suppose : langage sans $\top$ $\bot$, seul connecteurs :$\neg, \vee, \wedge$.

Notation : E = ensemble de formules *en f.n.n.* ; A, B = formules *f.n.n.* ; si ϕ est une formule, $E, \phi = E \cup \{\phi\}$.

Définition (Règles d'expansion)

- *La* **règle d'expansion** α *est :*

$$\frac{E, A \wedge B}{E, A, B} \ (\alpha)$$

L'ensemble $E, A \wedge B$ *est dit* ensemble développé *et l'ensemble* E, A, B *est dit* expansion *de* $E, A \wedge B$.

- *La* **règle d'expansion** β *est :*

$$\frac{E, A \vee B}{E, A \qquad E, B} \ \beta$$

L'ensemble $E, A \vee B$ *est dit* ensemble développé *et les ensembles* E, A *et* E, B *sont dits, respectivement,* expansion gauche et droite *de* $E, A \wedge B$.

Définition (Branche Complète, Ouverte)

Soit $\mathcal{B}$ une branche d'un tableau.

1. *La branche $\mathcal{B}$ est dite* **complète** *si elle ne peut pas être étendue par une application d'une règle d'expansion.*

Définition (Branche Complète, Ouverte)

Soit $\mathcal{B}$ une branche d'un tableau.

1. *La branche $\mathcal{B}$ est dite* **complète** *si elle ne peut pas être étendue par une application d'une règle d'expansion.*
2. *Elle est dite* **close** *si un de ses noeuds contient p et $\neg p$ pour quelque lettre propositionnelle p,* **ouverte** *sinon.*

Définition (Branche Complète, Ouverte)

Soit $\mathcal{B}$ une branche d'un tableau.

1. *La branche $\mathcal{B}$ est dite* **complète** *si elle ne peut pas être étendue par une application d'une règle d'expansion.*
2. *Elle est dite* **close** *si un de ses noeuds contient p et $\neg p$ pour quelque lettre propositionnelle p,* **ouverte** *sinon.*

Si l'ensemble E est la racine d'un tableau $\mathcal{T}$, on dit que $\mathcal{T}$ est un tableau *pour* E.

Tableaux propositionnels

Exemple de tableau pour $\{\neg p \vee q,\ \neg q \vee r,\ p \wedge \neg r\}$ où une branche n'est ni complète ni close, et les autre deux branches sont closes :

Tableaux propositionnels

Exemple de tableau pour $\{\neg p \vee q,\ \neg q \vee r,\ p \wedge \neg r\}$ où une branche n'est ni complète ni close, et les autre deux branches sont closes :

$$\dfrac{\dfrac{\dfrac{\neg p \vee q,\ \neg q \vee r,\ p \wedge \neg r}{\neg p \vee q,\ \neg q \vee r,\ p,\ \neg r}\,(\alpha)}{\dfrac{\neg p,\ \neg q \vee r,\ p,\ \neg r}{\neg p,\ \neg q,\ p,\ \neg r \qquad \neg p,\ r,\ p,\ \neg r}\,(\beta)}\,(\beta) \qquad\qquad q,\ \neg q \vee r,\ p,\ \neg r}$$

Tableaux propositionnels

Intuition : le processus de construction d'un tableau pour E peut être vu comme un essai systématique de montrer que l'hypothèse que E est satisfiable conduit à une contradiction, en explorant les possibilités d'obtenir un modèle de E.

Définition (Réfutation)

Un tableau $\mathcal{T}$ pour E est dit **réfutation (par tableaux)** *de E si toute branche de $\mathcal{T}$ est close.*

Exemple de réfutation de $\{\neg p \vee q,\ \neg q \vee r,\ p \wedge \neg r\}$: suite de l'arbre précedent, au tableau.

Exemple de tableau pour $\{\neg p \vee q, \neg q \vee r, \neg p \wedge \neg r\}$ où toute branche est close ou complète et qui n'est pas une réfutation : au tableau.

Tableaux propositionnels

Définition (Preuve)

Soit A une formule.
Une **preuve** *de A est une réfutation du singleton contenant une f.n.n. de la négation de A.*

Définition (Théorème)

S'il existe une preuve (par tableaux) de la formule A, A est dite alors **théorème** *(du système des tableaux).*

Exemple au tableau : preuve de la formule $p \to (q \to p)$, où p et q sont des lettres propositionnelles.

Définition

Dans le formalisme des tableaux, on dit qu'une règle d'expansion r est **correcte** *si la satisfiabilité de l'ensemble développé entraîne la satisfiabilité de son expansion quand r ne produit pas de branchements, et celle d'au moins une des ses expansions sinon.*

Les règles α et β sont correctes ! (*Pourquoi ?*)

Ceci entraîne (*pourquoi ?*) le résultat :

Théorème (Correction du Système des Tableaux)

- *Soit E un ensemble de formules (en f.n.n.).*
  S'il existe une réfutation de E, alors E est insatifiable.
- *Si $\mathcal{T}$ est une preuve d'une formule A alors A est valide.*

Tableaux propositionnels

Le processus de construction d'un tableau (de la logique booléenne) **termine toujours**.

Pourquoi ?

Théorème (Complétude du système des tableaux)

Soit E un ensemble fini de formules et soit E' l'ensemble obtenu en réecrivant toute formule de E en f.n.n. (et en éliminant $\top$, $\bot$, $\rightarrow$ et $\leftrightarrow$).

1. *Si E est insatisfiable alors il existe une réfutation de E'.*
2. *Si A est une formule valide, alors il existe une preuve de A.*

Pour prouver ce thm. :

Lemma

Soit $\mathcal{B}$ une branche d'un tableau compléte et ouverte.
L'interprétation $\mathcal{I}$ telle que, qque soit la lettre propositionnelle p de E :

$$\mathcal{I}(p) = V \quad ssi \quad il\ existe\ un\ noud\ E''\ de\ \mathcal{B}\ avec\ p \in E''$$

est un modèle de E.

Tableaux propositionnels

Le système des tableaux fournit un **algorithme** permettant de tester la satisfiabilité d'un ensemble de formules (ou d'une formule) et la validité d'une formule.

Pourquoi ?

Advantages par rapport aux tables de verité ?

Logique des prédicats (du 1^{er} ordre)

Limites de la Logique Propositionnelle

Raisonnement correct en langage naturel

Prémisses :

1. *Tout nombre entier positif divisible par 8 est pair.*
2. *L'entier positif 16 est divisible par 8.*

Conclusion : *l'entier positif 16 est pair.*

Formalisation adéquate en logique booléenne ?

Syntaxe de la Logique des Prédicats

But : Spécifier quelles expressions (+ riches de celles propositionnelles) sont considérées “bien formées”, ou “légales” indépendamment de leur signification et de leur vérité.

Syntaxe de la Logique des Prédicats

Définition (Alphabet)

Un **alphabet** $\mathcal{A}$ *d'un langage* $\mathcal{L}$ *de la logique des prédicats consiste des données suivantes :*

- *l'ensemble de connecteurs logiques, contenant le connecteur à un argument* $\neg$ *et les connecteurs binaires* $\wedge, \vee, \rightarrow, \leftrightarrow$
- *les constantes logiques* $\top$ *et* $\bot$
- *les parenthèses* $(,)$ *et la virgule* $,$
- *le symbole* $\forall$*, dit* quantificateur universel *et le symbole* $\exists$*, dit* quantificateur existentiel
- *un ensemble infini dénombrable* $\mathcal{X}$ *de symboles, les* variables individuelles
- *un ensemble dénombrable* $\mathcal{F}$ *de* symboles de fonction*, chacun muni d'une arité (nombre d'arguments) supérieure ou égale à 0.*
- *un ensemble dénombrable* $\mathcal{R} = \{P, Q, R, \ldots\}$ *de* symboles de prédicats *(ou relations), chacun muni d'une arité supérieure ou égale à 0.*

Syntaxe de la Logique des Prédicats

- si l'aritè d'un symbole de fonction est 0, on dit qu'il est une *constante (individuelle)*.
- si l'aritè d'un symbole de prédicats est 0, on dit qu'il est une *lettre propositionnelle*.
- $\langle \mathcal{F}, \mathcal{R} \rangle$ est la *signature* d'un langage $\mathcal{L}$.
- la signature peut varier d'un langage à un autre.

Syntaxe de la Logique des Prédicats

Définition (Terme (par récurrence))

Soit $\mathcal{F}$ un ensemble de symboles de fonctions.

- *toute variable individuelle est un terme sur $\mathcal{F}$;*
- *si $f \in \mathcal{F}$ est un symbole de fonction, f est d'arité n, avec $0 \leq n$, et $t_1, \ldots, t_n$ sont des termes sur $\mathcal{F}$, alors l'expression $f(t_1, \ldots, t_n)$ est un terme sur $\mathcal{F}$.*

Si f est une constante individuelle on abrège l'écriture du terme $f()$ en f.

Syntaxe de la Logique des Prédicats

Définition (Formule (par récurrence))

Soit $\mathcal{L}$ un langage du calcul des prédicats ayant $\Sigma = \langle \mathcal{F}, \mathcal{R} \rangle$ comme signature.

- *si $R \in \mathcal{R}$ est un symbol de prédicat, l'aritè de R est m, où $0 \leq m$ et $t_1, \ldots, t_m$ sont des termes sur $\mathcal{F}$, alors l'expression $R(t_1, \ldots, t_m)$ est une formule de $\mathcal{L}$, dite* formule atomique *(ou* atome*) de $\mathcal{L}$.*
- *si A_1 et A_2 sont des formules, alors $(\neg A_1), (A_1 \wedge A_2), (A_1 \vee A_2), (A_1 \rightarrow A_2), (A1 \leftrightarrow A_2)$ sont aussi des formules de $\mathcal{L}$.*
- *si A est une formule et x une variable, alors $(\forall x A)$ et $(\exists x A)$ sont des formules de $\mathcal{L}$.*

Si l'aritè d'un symbole R est 0 on abrège la notation de la formule atomique $R()$ en R.

Les symboles $\top$ et $\bot$ sont aussi des formules *atomiques*.

Dans $(\forall x A)$ (resp. $(\exists x A)$), la formule A est dite *portée* de $\forall x$ (resp. $\exists$).

Syntaxe de la Logique des Prédicats

Définition (Formule F (Backus-Naur))

Soit TE l'ensemble des termes,

$$F := \top \mid \bot \mid R(t_1, \ldots, t_n) \mid (\neg F) \mid (F * F) \mid (\forall x F) \mid (\exists x F)$$

avec $x \in \mathcal{X}$, $t_i \in TE$, $R \in \mathcal{R}$, $* \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$.

Arbre syntaxique d'une formule A

Même notion que dans le cas propositionnel, avec, en plus :

- si $Q \in \{\forall, \exists\}$, $x \in \mathcal{X}$, et A est $Qx\ A_1$, alors son arbre est $Arbre_A = \langle x, Q, Arbre_{A_1} \rangle$, c.à.d., le fils gauche du noeud Q est x et le fils droit est la racine de l'arbre syntaxique de A_1.

Attention : une feuille n'est plus forcement une formule (x est une variable individuelle, pas une formule !).

Conventions pour éliminer des parentèses

Celles de la logique propositionnelle +

1. Si Q, Q' sont des quantificateurs, on peut écrire $Qx\ Qy\ A$ à la place de $(Qx\ (Qy\ A))$.
   Par exemple, on pourra écrire

   $$\forall z\ \exists y\ \forall x\ (R(x,y) \wedge P(z))$$

   à la place de

   $$\forall z\ (\exists y\ (\forall x\ (R(x,y) \wedge P(z))))$$

2. Si A est une formule atomique, $Q \in \{\forall, \exists\}$ et $x \in \mathcal{X}$, l'écriture $Qx\ A$ signifiera que la portée de Q est A.
   Par exemple, on aura le droit d'écrire

   $$\exists x\ P(x) \vee R(x)$$

   à la place de

   $$(\exists x\ (P(x)) \vee R(x)$$

Définition (Variables Libres et Liées)

Soit A une formule. L'ensemble $\mathbf{VLB}(A)$ *des* **variables libres** *de A et l'ensemble* $VLE(A)$ *des* variables liées *de A, sont définis par récurrence sur les formules :*

- *Si A est un atome,* $\mathbf{VLB}(A)$ *est l'ensemble de toutes les variables qui apparaîssent au moins une fois dans A et* $VLE(A) = \emptyset$.
- *A n'est pas atomique. On a 3 sous-cas :*
  1. *si A est* $\neg A_1$, $\mathbf{VLB}(A) = \mathbf{VLB}(A_1)$ *et* $VLE(A) = VLE(A_1)$.
  2. *si A est* $A_1 * A_2$, *où* $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, *on a* $\mathbf{VLB}(A) = \mathbf{VLB}(A_1) \cup VLB(A_2)$ *et* $VLE(A) = VLE(A_1) \cup VLE(A_2)$.
  3. *Si A est $Qx\ A_1$, où Q est un quantificateur,* $\mathbf{VLB}(A) = \mathbf{VLB}(A_1) \setminus \{x\}$ *et* $VLE(A) = VLE(A_1) \cup \{x\}$.

Un terme t est dit **clos** s'il ne contient pas de variables.

Une formule A est dite **close** si elle ne contient pas de variables libres.

Syntaxe de la Logique des Prédicats

Définition

Un terme t est **libre pour une variable** x **dans une formule** *A si l'on a un des cas suivants :*

1. *A est atomique ;*
2. *A est $A_1 * A_2$ avec $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ et t est libre pour x dans A_1 et dans A_2 ;*
3. *A est $\neg A_1$ et t est libre pour x dans A_1 ;*
4. *A est $Qy\ A_1$ avec $Q \in \{\forall, \exists\}$ et x et y sont deux variables distinctes, y n'est pas une variable de t et t est libre pour x dans A_1.*

Exemple

Si $P \in \mathcal{R}$, $f, g \in \mathcal{F}$ et A est $\exists x\ (P(x) \wedge R(x, y, f(z)))$, alors le terme $g(y, z)$ est libre pour y et pour z dans A mais $f(x)$ n'est pas libre pour y (et pour z non plus).

Intuition ?

Notation : Si A est une formule, x est une variable et t est un terme, alors $A[x/t]$ indique la formule obtenue à partir de A en remplaçant toute occurrence libre de x dans A par t *qui est bien libre pour x dans A* (sinon, par déf. c'est A elle même).

Syntaxe de la Logique des Prédicats

Définition (Sous-formule)

L'ensemble $sousf(A)$ *des* **sous-formules d'une formule** *A d'un langage* $\mathcal{L}$ *de la logique des prédicats est défini récursivement par :*

- *si A est une formule atomique, alors* $sousf(A)$ *est* $\{A\}$ *;*
- *si A a la forme* $\neg A_1$*, alors* $sousf(A)$ *est l'union ensembliste de* $\{\neg A_1\}$ *et de* $sousf(A_1)$
- *si A a la forme* $A_1 * A_2$*, ou* $*$ *est un connecteur binaire, alors* $sousf(A)$ *est l'union ensembliste de* $\{A_1 * A_2\}$*, avec* $sousf(A)_1$ *et* $sousf(A)_2$*.*
- *Si Q est un quantificateur et A a la forme* $Qx\ A_1$ *alors* $sousf(A)$ *est l'union ensembliste de* $\{Qx\ A_1\}$ *avec* $sousf(A_1[x/t])$*, pour tout terme t de* $\mathcal{L}$ *qui soit libre pour x dans A.*

Possibilité d'un nombre infini de sous-formules d'une formule donnée !

Exemple. Soit un langage ayant un symbole de fonction, f, à 1 argument, pas de constantes et un symbole de prédicat p à 1 argument. La formule $\forall x\ p(x)$ aura, entre autres, les sous formules suivantes :
$\forall x\ p(x)$
$p(x)$ (ici le terme t qui remplace x dans $A_1 = p(x)$ est x lui même),
$p(f(x))$ $p(f(f(x))$, $p(f(f(f(x)))$,...
(ici, on a remplacé la variable x dans $A_1 = p(x)$ par le terme $f(x)$, puis par le terme $f(f(x))$, puis par le terme $f(f(f((x)))$, etc.).

Sémantique de la logique des prédicats

But : étant donné un langage $\mathcal{L}$, donner une signification aux symboles de sa signature, et préciser le domaine où les variables individuelles prennent une valeur, afin de permettre de déterminer la vérité d'une formule complexe à partir de celle des formules atomiques qui la composent.

La signification des connecteurs restera la même que pour le cas booléen, mais il faut fixer la signification des quantificateurs (unique, qque soit $\mathcal{L}$, comme pour les connecteurs).

Problème

Soient les formules :

$$A_1 \ : \ \forall x \, \exists y \, R(x, y)$$

$$A_2 \ : \ \exists y \, \forall x \, R(x, y)$$

$$A_3 \ : \ \forall x \, R(a, f(x))$$

où :
x et y sont des variables (individuelles),
R est un symbole de prédicat d'arité 2,
a est une constante (individuelle) et
f est un symbole de fonction d'arité 1.

Vraies ou fausses ?

Sémantique de la logique des prédicats

Définition (Interprétation)

Une **interprétation** $\mathcal{I}$ *d'un langage* $\mathcal{L}$ *de la logique des prédicats ayant une signature* $\langle \mathcal{F}, \mathcal{R} \rangle$ *est consituée par :*

- *Un ensemble fini ou infini – mais non-vide –* D*, appelé* domaine de discours *ou* univers *de* $\mathcal{I}$*.*
- *Pour tout symbole de fonction* $f \in \mathcal{F}$ *d'arité* n*, une assignation d'une fonction (totale) à* n *arguments* $\mathcal{I}(f) : D^n \to D$ *à* f*.*
  Dans le cas particulier où $n = 0$ *(*f *est une constante),* $\mathcal{I}(f)$ *est un* élément distingué *de* D*.*
- *Pour tout symbole de relation* $R \in \mathcal{R}$ *arité* n *tel que* $n \geq 1$*, une assignation d'une relation à* n *arguments* $\mathcal{I}(R)$*, où* $\mathcal{I}(R) \subseteq D^n$*, à* R*.*
  Dans le cas où $n = 0$*, on assigne à* R *une valeur booléenne, que l'on note toujours* $\mathcal{I}(R)$*.*

Exemples au tableau.

Sémantique de la logique des prédicats

Problème
Soient les termes :

$$t_1 \ : \ g(a, f(a))$$

$$t_2 \ : \ g(x, f(y))$$

où :
x et y sont des variables (individuelles),
a est une constante (individuelle),
f est un symbole de fonction d'arité 1 et
g est un symbole de fonction d'arité 2.

Quelles sont leur significations ? C.-à-d : quels individus (objets) nomment-ils ?

Sémantique de la logique des prédicats

Définition (Choix de valeurs)

Soient

- $\mathcal{L}$ *un langage du calcul des prédicats*
- X *l'ensemble des variables individuelles*
- *soit* $\mathcal{I}$ *une interprétation pour* $\mathcal{L}$ *ayant domaine (de discours)* D.

Un **choix de valeurs** *(où assignation de valeurs)* s *par rapport à* $\mathcal{I}$ *est une fonction ayant* X *comme domaine et* D *comme ensemble d'arrivée.*

Définition (Variante)

Etant donné

- *une choix de valeurs* s,
- *une variable* x_i,
- *un objet* $d \in D$,

la notation $s[x_i := d]$ *indique le choix de valeurs* s' *tel que :*

- $s'(x_i) = d$
- $s'(y) = s(y)$ *pur toute variable* y *distincte de* x_i.

Le choix de valeurs $s[x_i := d]$ *est dit aussi* x_i-**variante** *de* s.

Exemples au tableau.

Sémantique de la logique des prédicats

Définition (Valeur d'un terme)

Soient

- t *un terme d'un langage* $\mathcal{L}$ *de la logique des prédicats,*
- $\mathcal{I}$ *une interprétation de* $\mathcal{L}$ *ayant* D *comme domaine de discours,*
- s *un choix de valeurs par rapport à* $\mathcal{I}$.

La **valeur du terme** t *par rapport à* $\mathcal{I}$ *et au choix de valeurs* s, *notée* $[t]_{\mathcal{I},s}$, *est définie récursivement :*

- Base *:* $t = x_i$.
  $[x_i]_{\mathcal{I},s} = s(x_i)$
- Etape de récurrence *:* $t = f(t_1, \cdots, t_k)$, *où* $k \geq 0$.
  - *Si* $k = 0$, *c.-à-d.,* t *est une constante* c, $[c]_{\mathcal{I},s} = \mathcal{I}(c)$.
  - *Si* $k \geq 1$, *alors* $[f(t_1, \cdots, t_k)]_{\mathcal{I},s} = \mathcal{I}(f)([t_1]_{\mathcal{I},s}, \cdots, [t_k]_{\mathcal{I},s})$.

Lemma

Pour tout terme t, *sa valeur* $[t]_{\mathcal{I},s}$ *appartient à* D.

Définition (Valeur de verité (d'une formule))

Soient

- *A une formule d'un langage $\mathcal{L}$ de la logique des prédicats,*
- *$\mathcal{I}$ une interprétation de $\mathcal{L}$ ayant D comme domaine de discours,*
- *s un choix de valeurs par rapport à $\mathcal{I}$.*

La **valeur de verité de** *A par rapport à $\mathcal{I}$ et au choix de valeurs s, notée $[A]_{\mathcal{I},s}$, est définie récursivement :*
Base *: $A = R(t_1, \cdots, t_k)$ est atomique.*

- *Si $k = 0$, alors $[R]_{\mathcal{I},s} = \mathcal{I}(R)$.*
- *Si $k \geq 1$, alors $[R(t_1, \cdots, t_k)]_{\mathcal{I},s} =$*
  - *V si le n-uplet $\langle [t_1]_{\mathcal{I},s}, \ldots, [t_k]_{\mathcal{I},s} \rangle$ d'éléments de D appartient à la relation $\mathcal{I}(R)$,*
  - *F sinon.*
- *Si A est $\top$ ou bien $\bot$:*
  $[\top]_{\mathcal{I},s} = V$ et $[\bot]_{\mathcal{I},s} = F$.

Sémantique de la logique des prédicats

Définition (Valeur de verité (suite de la définition))

Etape de récurrence. *On doit considérer plusieurs sous-cas.*

- *A est* $\neg A_1$.
  On a $[A]_{\mathcal{I},s} = f_\neg([A_1]_{\mathcal{I},s})$.
- *A est* $A_1 * A_2$, *où* $*$ *est un connecteur binaire.*
  On a $[A]_{\mathcal{I},s} = f_*([A_1]_{\mathcal{I},s}, [A_2]_{\mathcal{I},s})$.
- *A est* $\forall x\ A_1$.
  On a $[A]_{\mathcal{I},s} = V$ *si, quelque soit* $d \in D$, $[A_1]_{\mathcal{I},s[x:=d]} = V$,
  $[A]_{\mathcal{I},s} = F$ *sinon.*
- *A est* $\exists x\ A_1$.
  On a $[A]_{\mathcal{I},s} = V$ *s'il existe au moins un* $d \in D$ *tel que* $[A_1]_{\mathcal{I},s[x:=d]} = V$,
  $[A]_{\mathcal{I},s} = F$ *sinon.*

Exemple : evaluation de $\forall x \exists y (R(x,y) \wedge R(f(x),a)$ *où* R *est un symbole de prédicat d'arité 2,* f *est un symbole de fonction d'arité 1 et* a *est une constante.* Au tableau.

Sémantique de la logique des prédicats

Définition

Soient A une formule et $\mathcal{I}$ une interprétation.

- *A est* **vraie** *par rapport à $\mathcal{I}$ si $[A]_{\mathcal{I},s} = V$ pour* tout *choix de valeur s.*
- *$\mathcal{I}$ est un* **modèle de la formule** *A si A est vraie par rapport à $\mathcal{I}$.*
- *$\mathcal{I}$ est un* **modèle d'un ensemble de formules** *E si elle est modèle de tout élément de E.*

Sémantique de la logique des prédicats

Soit A la formule $\forall x \exists y\ R(x, y)$, où R est un symbole de prédicat, soit $\mathcal{I}$ une interprétation et soit s un choix de valeurs pour les variables. Supposons que $[A]_{\mathcal{I},s} = V$.

- Est-il forcement vrai que $[\exists y\ R(x, y)]_{\mathcal{I},s} = V$?
- Est-il forcement vrai que $[\exists y\ R(z, y)]_{\mathcal{I},s} = V$?
- Est-il forcement vrai que $[\exists y\ R(y, y)]_{\mathcal{I},s} = V$?

Quel est le lien intuitif avec la notion (syntaxique) de terme libre pour une variable ?

Sémantique de la logique des prédicats

Définition

Soient A une formule et E un ensemble de formules.

- *A est* **valide** *si, pour n'importe quelle interprétation $\mathcal{I}$, $\mathcal{I}$ est modèle de A.*
- *Une formule A est* **satisfiable** *s'il existe au moins une interprétation $\mathcal{I}$ et au moins un choix de valeurs pour les variables s tels que $[A]_{\mathcal{I},s} = V$.*
- *Un ensemble de formules E est satisfiable s'il existe au moins une interprétation $\mathcal{I}$ et au moins un choix de valeurs pour les variables s tels que, quelque soit la formule B de E, on a $[B]_{\mathcal{I},s} = V$.*
- *Si A (resp. E) n'est pas satisfiable, alors c'est* **insatisfiable**.

Sémantique de la logique des prédicats

Définition

Soient A une formule et E un ensemble de formules.

- *A est* **valide** *si, pour n'importe quelle interprétation $\mathcal{I}$, $\mathcal{I}$ est modèle de A.*
- *Une formule A est* **satisfiable** *s'il existe au moins une interprétation $\mathcal{I}$ et au moins un choix de valeurs pour les variables s tels que $[A]_{\mathcal{I},s} = V$.*
- *Un ensemble de formules E est satisfiable s'il existe au moins une interprétation $\mathcal{I}$ et au moins un choix de valeurs pour les variables s tels que, quelque soit la formule B de E, on a $[B]_{\mathcal{I},s} = V$.*
- *Si A (resp. E) n'est pas satisfiable, alors c'est* **insatisfiable**.

Remarque : toute formule de la forme $(\forall x\ A) \rightarrow \exists x\ A$ est valide .
Lien avec le fait que, par déf., le domaine de toute interprétation est *non-vide* ?

Sémantique de la logique des prédicats

Définition (Conséquence logique)

Soit $E = \{A_1, A_2 \ldots\}$, *un ensemble de formules, soit* G *une formule.*
On dit que G *est* **conséquence logique** *de* E, *et on note* $E \models G$, *si*, pour toute interprétation $\mathcal{I}$ et tout choix de valeur s *(relatif à* $\mathcal{I}$*)*,
la propriété : quelque soit $i \in N$ $[A_i]_{\mathcal{I},s} = V$,
implique la propriété : $[G]_{\mathcal{I},s} = V$.

Soit p un symbole de prédicat d'arité 1.
Est-il vrai que $p(x) \models \forall x\ p(x)$?

Sémantique de la logique des prédicats

Définition (Conséquence logique)

Soit $E = \{A_1, A_2 \ldots\}$, *un ensemble de formules, soit* G *une formule.*
On dit que G *est* **conséquence logique** *de* E, *et on note* $E \models G$, *si*, pour toute interprétation $\mathcal{I}$ et tout choix de valeur s *(relatif à* $\mathcal{I}$*)*,
la propriété : quelque soit $i \in N$ $[A_i]_{\mathcal{I},s} = V$,
implique la propriété : $[G]_{\mathcal{I},s} = V$.

Soit p un symbole de prédicat d'arité 1.
Est-il vrai que $p(x) \models \forall x\ p(x)$?

Est-il vrai que $\forall x\ p(x) \models \forall z\ p(z)$?
Pourquoi ?

Sémantique de la logique des prédicats

Définition (Equivalence logique)

Deux formules A_1 *et* A_2 *sont* **logiquement équivalentes**, *ou* $A_1 \equiv A_2$, *si* $A_1 \models A_2$ et $A_2 \models A_1$.

Quelques équivalences logiques utiles

- $\forall x\ A \ \equiv \ \neg\exists x\ \neg A$ (définissabilité de $\forall$ par $\exists$)
- $\exists x\ A \ \equiv \ \neg\forall x\ \neg A$ (définissabilité de $\exists$ par $\forall$)
- Soit A' une formule obtenue à partir de A en remplaçant toute occurrence de la variable x par y, où $y \notin VLB(A)$. Alors, $\forall x\ A \ \equiv \ \forall y\ A'$ et $\exists x\ A \ \equiv \ \exists y\ A'$ (*principe de renommage*)

Exemples d'applications du principe de rennomage : soit p un symbole de prédicat à un argument, et soit r un symbole de prédicat à 2 arguments.

$\forall x\ p(x) \ \equiv \ \forall y\ p(y)$
$\exists x\ p(x) \ \equiv \ \exists y\ p(y)$

Mais, attention : $\forall x\ r(x, y) \ \not\equiv \ \forall y\ r(y, y)$!
Ici : y est libre dans $r(x, y)$, et ce n'est pas correct de remplacer x par y.
Penser à ce qui se passe pour l'interprétation où l'univers D est l'ensemble des naturels, $\mathcal{I}(R)$ est la relation $\geq$ et le choix de valeurs s donne la valeur 0 à y.

Systèmes formels de preuve pour la logique des prédicats

But : comme dans le cas propositionnel, avoir des moyens purement *syntaxiques*, c'est-à-dire ne tenant compte que de la forme des formules, pour montrer qu'une formule donnée est valide.

Ici c'est crucial, car l'analogue d'un table de vérité - objet *fini* - est **infini**.

Tableaux pour le Calcul des Prédicats

Le système de tableaux que l'on va voir : **Tableaux dits "avec variables libres"**.

Pour pouvoir le décrire, une digression préalable, sur les notions de substitution d'un terme à une variable et unification est nécessaire.

Pour comprendre purquoi, raisonnons intuitivement sur comment on pourrait prouver avec un tableau la validité de $(\forall x(p(x) \rightarrow q(x)) \wedge p(a)) \rightarrow q(a)$, où a est une constante. Au tableau.

Digression : Substitutions et unificateurs

Définition (Substitutions)

Une substitution *est un ensemble fini* $\{x_1/t_1, \ldots, x_n/t_n\}$ *de couples formés d'une variable (individuelle)* x_i *et d'un terme* t_i *différent de* x_i*.*

On impose aussi que, pour tout $1 \leq i, j \leq n$, x_i et x_j soient deux variables distinctes.
On appelle l'ensemble de variables $\{x_1, \cdots, x_n\}$ *domaine* de la substitution.
On utilisera les lettres minuscules grecques $\alpha, \beta, \gamma, \ldots, \sigma, \tau, \ldots$ pour indiquer des substitutions.

Soient x_1, x_2, x_3 des variables, a une constante, f un symbole de fonction unaire.
Quel ensemble parmi

$$\{x_1/x_1,\ x_2/a\}, \qquad \{a/x_1,\ x_2/x_3\}, \qquad \{x_1/a,\ x_2/f(x_3)\} \qquad \{x_1/a,\ x_1/f(x_3)\}$$

est-il une substitution ?

Digression : Substitutions et unificateurs, suite

Si σ est une substitution, t est un terme et A une formule, on note

- $(t)_\sigma$ le terme résultat de l'application de σ à toute occurrence de variable de t ;
- $(A)_\sigma$ la formule résultat de l'application de σ à toute occurrence *libre* de variable de A.

Définition (Composition)

Soient $\theta = \{x_1/s_1, \ldots, x_m/s_m\}$ *et* $\sigma = \{y_1/t_1, \ldots, y_n/t_n\}$ *deux substitutions.*
La composition *de* θ *et* σ, *notée* $\theta \circ \sigma$, *est la substitution obtenue à partir de l'ensemble*

$$\{x_1/(s_1)_\sigma, \ldots, x_m/(s_m)_\sigma, y_1/t_1, \ldots, y_n/t_n\}$$

en supprimant :

- *tout élément de la forme* x/x *où* x *est une variable ;*
- *tout élément* y_j/t_j *tel que* y_j *appartient au domaine de* σ *mais appartient aussi au domaine de* θ.

Intuitivement : c'est θ qui gagne, en cas de conflit.
N.B. : $\theta \circ \sigma$ indique la substitution obtenue en faisant *d'abord* θ *puis* σ.

Digression : Substitutions et unificateurs, suite

Exemple

Ici, x, y, z, w sont des variables, a est une constante et g est un symbole de fonction à 1 argument.

Soient $\theta = \{x/a,\ y/g(a,y), z/w\}$ et $\sigma = \{y/a, z/f(z),\ w/z\}$ deux substitutions.

- *Qui est le domaine de θ ? et celui de σ ? Qui est $\theta \circ \sigma$?*
- *Soit t le terme $g(y,z)$. Qui sont $(t)_\theta, ((t)_\theta)_\sigma$ et $(t)_{\theta\circ\sigma}$?*
- *Soit R un symbole de prédicat binaire. Qui est $(R(g(y,z),x))_{\theta\circ\sigma}$?*

Digression : Substitutions et unificateurs, suite

Définition

- *On appelle* problème d'unification *tout ensemble d'équations P de la forme $\{t_1 = s_1, \ldots, t_n = s_n\}$ où tout t_i et tout s_i sont des termes.*
- *Une substitution σ est un* unificateur *de P si, pour tout $i \leq n$, les termes $(t_i)_\sigma$ et $(s_i)_\sigma$ sont (syntaxiquement) identiques.*
  On note par $U(P)$ l'ensemble d'unificateurs de P.
- *On dit que P est* unifiable *si $U(P) \neq \emptyset$.*
- *On dit que deux* termes *t et t' sont* unifiables *par la substitution σ si σ est un unificateur du problème d'unification $\{t = t'\}$.*
- *De même, on dit que* deux formules atomiques $R(t_1, \cdots, t_n)$ et $R(s_1, \cdots, s_n)$ sont *unifiables* par la substitution σ si σ *est un unificateur* du problème d'unification $\{t_1 = s_1, \ldots, t_n = s_n\}$.

Digression : Substitutions et unificateurs, suite

Etant données deux formules atomiques A_1 et A_2, plusieurs unificateurs différents peuvent exister. Il est aussi possible qu'aucun unificateur n'existe.

Exemple

Ici, x, y, v, z, w, z' sont des variables, a est une constante, f et g sont 2 symboles de fonction avec, respectivement, 1 et 3 arguments.

- $A_1 = R(x, f(y))$ *et* $A_2 = R(g(a, v, v), f(v))$.
  $\sigma = \{x/g(a, y, y),\ v/y\}$ *est un unificateur de* A_1 *et* A_2.
- *Deux autres encore :*
  $\theta = \{x/g(a, z', z'),\ v/z',\ y/z'\}$.
  $\delta = \{x/g(a, z, z),\ v/z,\ y/z,\ w/a\}$.
- $A'_1 = r(x, f(x))$ *et* $A'_2 = r(y, y)$. *Pas d'unificateur.*
- $A'_2 = r(x, b)$ *et* $A'_2 = r(y, c)$ *où a et b sont des constantes. Pas d'unificateur.*
- $A'_3 = r(x, b)$ *et* $A'_2 = r(y, f(z))$ *où a est une constantes et f un symbole de fonction. Pas d'unificateur.*

Digression : Substitutions et unificateurs, suite

Définition (Unificateur Principal)

Soit P un problème d'unification et soit σ_0 un unificateur de P.
On dit que σ_0 est un unificateur principal *(ou un unificateur* le plus général*) de P si, pour tout unificateur σ' de P, il existe une substitution σ'' telle que $\sigma' = \sigma_0 \circ \sigma''$.*

Exemple

Exemple précédent : σ est principal pour le problème d'unification associés à A_1 et A_2.
En fait, $\theta = \sigma\{v/z', y/z'\}$, $\delta = \sigma\{v/z, y/z, w/a\}$.

Si un problème d'unification P est unifiable, alors P a un unificateur unique, modulo rennomage de variables et restriction seulement aux variables présents dans P.
On le note *mgu* (*most general unifier*).

Un *algorithme qui permet de décider si deux formules atomiques sont unifiables* et qui, si elles le sont, retourne le mgu : en TD.

A nouveau sur les tableaux pour les prédicats

Même principe que pour le cas booléen : pour prouver la validité de A on réfute la négation de A.

A nouveau, on travaille sur les ensembles de formules ne contenant que $\neg, \wedge, \vee$ comme connecteurs booléens et en f.n.n.

Mais on a aussi les quantificateurs, donc, pour réecrire en f.n.n., on utilise aussi :

$$\neg\forall x A \equiv \exists x \neg A \qquad \neg\exists x A \equiv \forall x \neg A$$

Puis, il faut 2 règles d'expansion en plus pour gérér les quantificateurs.

Tableaux pour le Calcul des Prédicats

On utilise un nouveau ensemble $L = \{l_1, l_2, l_3, \ldots\}$ de variables qui seront toujours libres (on les appelle *paramètres*).

On utilise aussi un nouveau ensemble SKO de symboles de fonction (dits *fonctions de Skolem*)

Nouvelles règles γ et δ :

$$\frac{E, \forall v\ B}{E, B[v/l_i], \forall v\ B} \qquad \frac{E, \exists v\ B}{E, B[v/sk_j(l_1, ..., l_n)]}$$

où

- l_i est un élément de L **nouveau** (pas encore utilisée dans l'arbre en cours de construction),
- $l_1 \cdots l_n$ sont tous les paramètres de $\exists v\ B$, et
- sk_j est un symbole de fonction de SKO à n arguments pas encore utilisé et **distinct** de tout symbole de fonction des formules de l'arbre.

Remarquez la répétition de la formule universelle dans la règle γ ! Elle est nécessaire : parfois c'est nécessaire de traiter une même formule universelle plusieurs fois : voir un exemple plus loin.

Définition

Soit $\mathcal{B}$ une branche d'un tableau.

1. *La branche $\mathcal{B}$ est dite* complète *si aucune règle d'expansion n'est pas applicable.*
2. *Une branche est dite* close *si elle a un noeud qui contient*

$$R(t_1, ..., t_n) \qquad \neg R(s_1, ..., s_n)$$

pour quelque symbole de prédicat r et les deux atomes :

$$R(t_1, ..., t_n) \qquad R(s_1, ..., s_n)$$

sont unifiables ; elle est ouverte *sinon.*

3. *Soit E un ensemble de formules closes en f.n.n..*
Un tableau $\mathcal{T}$ pour E est dit réfutation (par tableaux) *de E s'il existe une substitution σ pour les paramétres L qui sont dans $\mathcal{T}$ permettant de clore* **au même temps toutes** *les branches de $\mathcal{T}$.*
4. *Si A est une formule close, une* preuve *de A est une réfutation du singleton contenant une f.n.n. de la négation de A.*

Exemple

Preuve par tableaux de la validité de la formule :

$$\forall x\ \exists y\ \forall z\ \exists w(P(x,y) \vee \neg P(w,z))$$

On prouve donc l'insatisfiabilité de :

$$\exists x\ \forall y\ \exists z\ \forall w(\neg P(x,y) \wedge P(w,z))$$

Au tableau. Attention :

- Après avoir appliqué chaque règle une fois à chaque formule, on se retrouve avec $\neg P(a, l_1), P(l_2, f(l_1))$ et on ne peut pas clore le tableau car les deux termes l_1 et $f(l_1)$ ne sont pas unifiable. Mais si on déclare que la formule racine est insatisfiable, on fait une erreur.
- Il faut appliquer la règle γ deux fois à la formule $\forall y\ \exists\ \forall w(\neg P(a,y) \wedge P(w,z))$ pour pouvoir clore le tableau !

Tableaux pour le Calcul des Prédicats

Soit E un ensemble de formules closes et ne contenant pas de paramètres.

Théorème (Correction du Calcul)

S'il existe une réfutation $\mathcal{T}$ par tableaux de E, alors E est insatisfiable.

Théorème (Complétude du Calcul)

Si E est insatisfiable alors il existe une réfutation $\mathcal{T}$ par tableaux de E.
Par conséquent, si une formule close (et sans paramètres) est valide, elle a forcement une preuve.

At-on toujours la termination du processus de construction d'un tableau ?