Logique propositionnelle

Gilles Falquet

CUI - Université de Genève

5 décembre 2018

Les objectifs de la logique sont de

- Traiter formellement les notions de vérité et fausseté
- Formaliser et justifier le raisonnement logique intuitif, la déduction logique
- Permettre le raisonnement (humain ou automatique) dans des cas non intuitifs, complexes
- Formaliser les théories mathématiques, ou autres

Aspect déductif / syntaxique

On s'intéresse à la validité du raisonnement et à la notion de preuve.

Le raisonnement suivant est-il valide ?

- quand il pleut le professeur Tournesol oublie toujours son parapluie
- aujourd'hui le professeur Tournesol n'a pas son parapluie

donc

- aujourd'hui il pleut

.

Quelles sont les règles qui permettent de décider si une preuve est valide ?

Aspect sémantique

Dans ce cas on s'intéresse à évaluer la vérité ou fausseté de certains énoncés dans certaines situations.

Exemple

Est-il possible que les énoncés suivants soient vrais simultanément ?

- Si le feu vert est allumé, le rouge est éteint
- Si le feu orange est allumé, le vert est éteint
- Le feu rouge et le feu orange sont allumés

Logique et informatique

La logique apparaît en de nombreuses circonstances en informatique.

- Spécifier formellement les systèmes : invariants, pré et post conditions, preuves de propriétés des systèmes, langages de spécification, ...
- Logique des données : expression des requêtes sur les bases de données, expression des contraintes d'intégrité, ...
- Intelligence artificielle : systèmes experts, inférence automatique, représentation des connaissances, programmation logique.
- informatique théorique : problèmes de calculabilité et de complexité (problème SAT)

Logique des propositions : propositions

L'objet de base de la logique des propositions est ... la proposition. Il s'agit d'un énoncé qui peut être vrai ou faux. Par exemple

Jean a mangé une pomme

Albertine pense qu'il pleuvra demain

Tous les nombres impairs sont des nombres premiers

Cette phrase a moins de neuf mots

Contre-exemples :

* *Quatorze moins six*

* *Que pense Albertine ?*

Logique des propositions : composantes

Une logique parmi d'autres, composée de

- un langage (écriture des formules)
- une sémantique (évaluation des formules)

Pour laquelle il existe des système de preuve utilisant

- des axiomes
- des règles d'inférence

Syntaxe des formules de logique propositionnelle

Vocabulaire :

- les connecteurs logiques $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$
- les parenthèses (et)
- des symboles de variables propositionnelles : $a, b, c, \ldots, p, q, r, s$, $a_1, b_1, \ldots$ etc.

Syntaxe des formules :

```
Formule ::= Symbole_de_variables
          | Formule ∧ Formule
          | Formule ∨ Formule
          | Formule → Formule
          | Formule ↔ Formule
          | ¬ Formule
          | ( Formule )
          | vrai | faux
```

Exemples

$$p$$

$$q \to p$$

$$\neg\neg p$$

$$\neg(p \vee (r \wedge \neg s) \vee t)$$

$$q \to p \leftrightarrow s \to p$$

$$(q \to p) \leftrightarrow (s \to p)$$

Pour éviter toute ambigüité on utilise les parenthèses

Autres notations

$\wedge$	$\&$
$\rightarrow$	$\Rightarrow$ ou $\supset$
$\leftrightarrow$	$\Leftrightarrow$
$\neg p$	$\sim p$ ou $\bar{p}$

Sémantique de la logique des propositions

Sémantique d'un langage (en général)

Etant donné un langage $\mathcal{L}$

- on veut attribuer un sens à chaque chaine (expression) w de $\mathcal{L}$.
- le sens est une valeur prise dans un ensemble D appelé domaine d'interprétation.

Pour la logique des proposition le domaine d'interprétation est l'ensemble $D = \{\mathbf{v}, \mathbf{f}\}$ (vrai, faux) ou $\{0, 1\}$

Principe de l'interprétation des formules

Pour définir une interprétation $\mathcal{I}$ on va

- Fixer une interprétation $\mathcal{I}(x)$ de chaque variable propositionnelle x
- Étendre cette interprétation aux formules, en appliquant des règles d'évaluation des connecteurs.

Règles d'interprétation des connecteurs :

x	y	$\neg x$	$x \wedge y$	x$\vee$y	$x \rightarrow y$	$x \leftrightarrow y$	*vrai*	*faux*
v	v	f	v	v	v	v	v	f
v	f	f	f	v	f	f	v	f
f	v	v	f	v	v	f	v	f
f	f	v	f	f	v	v	v	f

Exemple :

Pour le langage comprenant les variables p, q, r on définit deux interprétations $\mathcal{I}, \mathcal{J}$ des variables

	p	q	r
$\mathcal{I}$	v	f	v
J	f	f	f

Ce qui permet d'interpréter les formules

	$p \wedge q$	$p \vee \neg p$	$p \wedge (\neg r \vee q)$	$p \rightarrow q$	$q \rightarrow p$
$\mathcal{I}$	f	v	f	f	v
J	f	v	f	v	v

Modèles

- On considère qu'une interprétation $\mathcal{I}$ est un "monde possible".
- Pour une formule Φ donnée existe-t-il un monde dans lequel Φ est vraie ?

Définitions

Un modèle d'une formule Φ est une interprétation M telle que

$$M(\Phi) = \mathbf{v}$$

Un modèle d'un ensemble de formules $F = \{\Phi_1, \ldots, \Phi_k\}$ est une interprétation M telle que

$$M(\Phi_1) = \cdots = M(\Phi_k) = \mathbf{v}$$

Satisfiabilité

- Il peut exister zéro, un ou plusieurs modèles pour une formule ou un ensemble de formules.
- S'il existe au moins un modèle de F on dit que F est *satisfiable* (ou consistant ou non contradictoire).
- Sinon F est dite *inconsistant* .

Exemple (1)

L'ensemble $F = \{p \wedge q, q \vee r\}$ est satisfiable
Il y a deux modèles :

	p	q	r	$p \wedge q$	$q \vee r$
$\mathcal{M}_1$	v	v	v	v	v
$\mathcal{M}_2$	v	v	f	v	v

Exemple (2)

L'ensemble

$$F = \{\neg p \vee q, r \vee \neg q, p, \neg r, r \vee (q \rightarrow (p \wedge \neg r))\}$$

est inconsistant.

- Dans tout modèle $\mathcal{I}$ on devrait avoir
  - $\mathcal{I}(p) = \mathbf{v}$
  - $\mathcal{I}(r) = \mathbf{f}$
- par conséquent
  - $\mathcal{I}(q) = \mathbf{f}$, pour avoir $\mathcal{I}(r \vee \neg p) = \mathbf{v}$
  - $\mathcal{I}(q) = \mathbf{v}$, pour avoir $\mathcal{I}(\neg p \vee q) = \mathbf{v}$
- d'où contradiction

Tautologies

Les tautologies sont des formules vraies quelle que soit l'interprétation.
Par exemple :

$$P \vee \neg P$$
$$P \rightarrow P$$
$$((P \rightarrow Q) \wedge P) \rightarrow Q$$
$$(P \vee Q) \leftrightarrow \neg(\neg P \wedge \neg Q)$$

Les tautologies sont des vérités universelles qui ne dépendent pas de l'état du monde.

Equivalence

Deux formules E et F sont équivalentes si pour toute interprétation elles prennent les même valeurs de vérité.
p.ex.

$$p \wedge q \equiv \neg(p \rightarrow \neg q)$$

On peut voir que : E et F sont équivalentes si la formule

$$E \leftrightarrow F$$

est une tautologie (est toujours vrai)
Les équivalences permettent des manipulations syntaxiques qui préservent la sémantique.

Quelques équivalences :

Utiles pour simplifier les formules ou les restructurer.
$A, B, C, \ldots$ sont des variables ou des formules quelconques

double négation	$\neg(\neg A) \equiv A$
associativité	$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$
	$(A \vee B) \vee C \equiv A \vee (B \vee C)$
commutativité	$A \wedge B \equiv B \wedge A$
	$A \vee B \equiv B \vee A$

(suite)

idempotence	:	$A \wedge A \equiv A,\ A \vee A \equiv A$
distributivité-ou	:	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
distributivité-et	:	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
De Morgan	:	$\neg(A \wedge B) \equiv \neg A \vee \neg B; \neg(A \vee B) \equiv \neg A \wedge \neg B$
implication-ou	:	$A \rightarrow B \equiv \neg A \vee B$
double implication	:	$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$
tiers exclu	:	$A \wedge \neg A \equiv \textbf{faux},\ A \vee \neg A \equiv \textbf{vrai}$

Forme normale conjonctive

Définition

Une formule est en forme normale conjonctive si elle est de la forme forme

$$C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

où chaque C_i, appelé clause ou maxterm, est de la forme

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_m$$

où les ℓ_i, appelés litéraux, sont des formules atomiques, donc des variables, éventuellement précédée d'une néagtion $\neg$.

Exemple

$(a \vee b) \wedge (\neg b \vee c \vee d) \wedge (c \vee \neg a \vee \neg d)$

Mise sous forme normale conjonctive

Théorème

Toute formule est équivalente à une formule en FNC et il y a un algorithme pour obtenir cette forme.

L'algorithme consiste à appliquer exhaustivement une série de règles d'équivalence

1. Appliquer les équivalence double-implique et implique-ou pour supprimer les $\rightarrow$ et $\leftrightarrow$.
2. Appliquer De Morgan pour «descendre» les négation à côté des variables.
3. Appliquer la distributivité pour descendre les $\vee$ et remonter les $\wedge$
4. Appliquer l'associativité des $\vee$ et $\wedge$

Exemple

- $((b \vee c) \rightarrow a) \vee d$
- $\equiv (\neg(b \vee c) \vee a) \vee d$ $\quad$ (implique-ou)
- $\equiv ((\neg b \wedge \neg c) \vee a) \vee d$ $\quad$ (de Morgan)
- $\equiv ((\neg b \vee a) \wedge (\neg c \vee a)) \vee d$ $\quad$ (distributivité)
- $\equiv ((\neg b \vee a) \vee d) \wedge ((\neg c \vee a)) \vee d)$ $\quad$ (distributivité)
- $\equiv (\neg b \vee a \vee d) \wedge (\neg c \vee a \vee d)$ $\quad$ (associativité-ou)

Autre forme normale

Il existe aussi une forme normale disjonctive :

$$D_1 \vee D_2 \vee \cdots \vee D_m$$

où chaque D_i est lui même de la forme

$$\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_m$$

où les ℓ_i sont des littéraux.

Utilisation de la FNC pour la satisfiabilité

L'algorithme de Davis-Putnam-Logemann-Loveland (DPLL) sert à trouver un modèle (s'il y en a un) pour une formule Φ qui est en FNC. Il est basé sur les observation suivantes :

- si une clause est littérale (une seule variable), on peut fixer la valeur de sa variable, et mettre à jour les autres clauses
- si une variable est « pure » (n'apparait que positivement ou négativement), on peut fixer sa valeur pour rendre vraies toutes les clauses où elle apparait, et éliminer ces clauses
- s'il n'y a ni clause littérale, ni variable pure, il faut choisir un littéral k (au hasard ou à l'aide d'une heuristique) et essayer de trouver un modèles pour $\Phi \wedge k$ ou pour $\Phi \wedge \neg k$
- si à un moment donné la formule n'est composée que de littéraux et qu'elle est consistante, on a trouvé un modèle.
- si on a généré une clause vide on n'arrivera pas à trouver de modèle, il faut essayer avec d'autres valeurs des variables.

Exemple/principe

1. $F = (p \vee \neg q) \wedge (\neg p \vee \neg s \vee t) \wedge (q \vee s)$
   - il n'y a pas de clause littérale, essai avec $k = \neg p$
2. $F' = (p \vee \neg q) \wedge (\neg p \vee \neg s \vee t) \wedge (q \vee s) \wedge \neg p$
   - clause littérale $\neg p$, $\Rightarrow \mathcal{M}(p) = \mathbf{f}$
3. $\Rightarrow F'' = (\neg q) \wedge (q \vee s)$
   - clause littérale $\neg q$, $\Rightarrow \mathcal{M}(q) = \mathbf{f}$
4. $\Rightarrow F''' = (s)$
   - clause littérale s, $\Rightarrow \mathcal{M}(s) = \mathbf{v}$
5. $\Rightarrow F'''' =$ vide, fini

Rem. On peut choisir n'importe quelle valeur pour $M(t)$

DPLL(F,M)

Entrées : une formule F en FNC, un modèle partiel M
Sorties : *faux* si on n'a pas trouvé de modèle, *vrai* si on a trouvé un modèle. Dans ce dernier cas M contient le modèle

si F *est vide* **alors retourner** *vrai*;
si F *contient une clause vide* **alors retourner** *faux*;
tant que *il existe une clause littérale* (x) *ou* $(\neg x)$ *dans* F **faire**
  fixer $M(x) = \mathbf{v}$ (resp. $\mathbf{f}$) ;
  $F =$ PropagationValeur(x, F)
tant que *il existe une variable pure* x *(qui apparait toujours positivement ou négativement)* **faire**
  fixer $M(x) = \mathbf{v}$ (resp. $\mathbf{f}$) ;
  $F =$ supprimer les clauses où x apparait
$k =$ choisir un littéral de F ;
retourner DPLL$(F \wedge k)$ *ou sinon* DPLL$(F \wedge \neg k)$

Propagation des valeurs

Si on a fixé $M(x) = \mathbf{v}$

1. supprimer toutes les clauses ou x apparaît positivement
2. enlever $\neg x$ de toutes les clauses où il apparaît

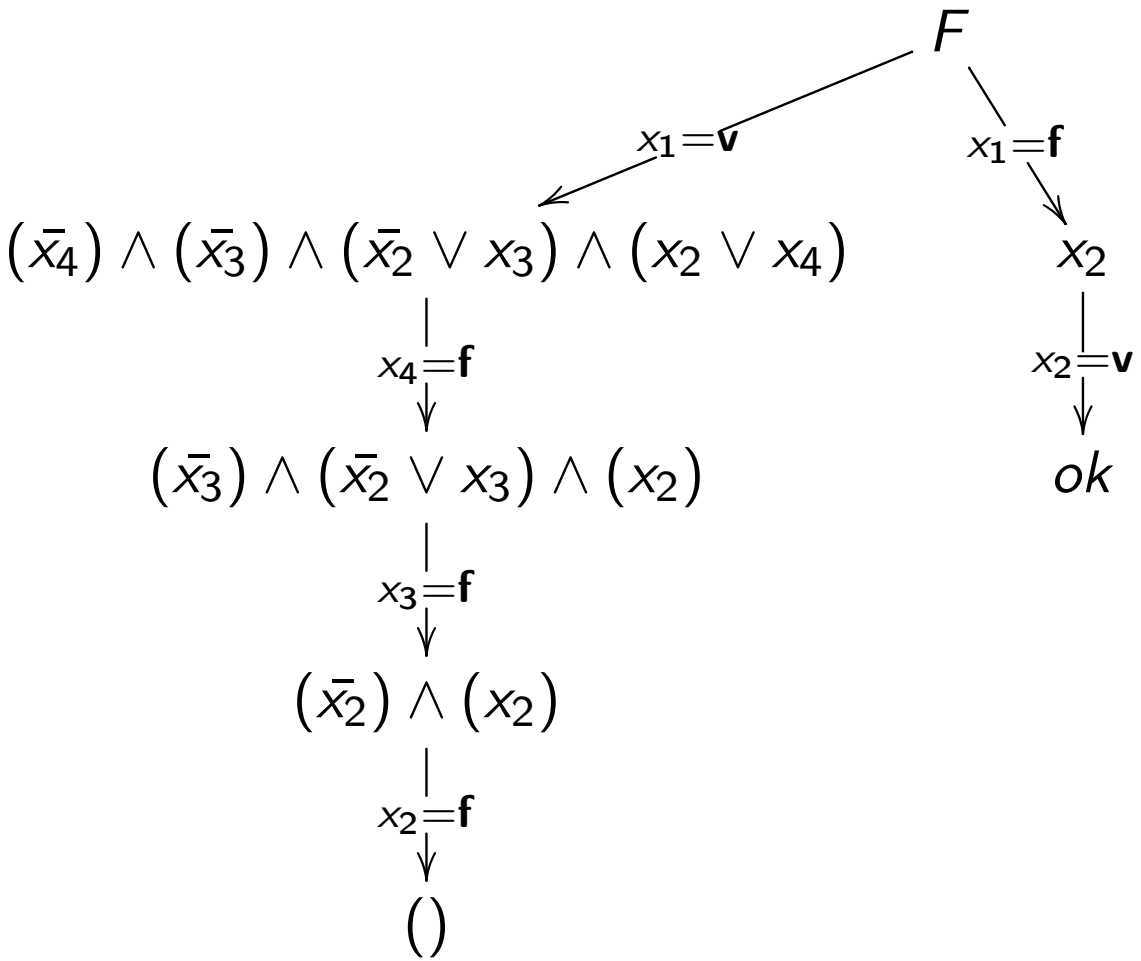
Si on a fixé $M(x) = \mathbf{f}$

1. supprimer toutes les clauses ou $\neg x$ apparaît
2. enlever x de toutes les clauses où il apparaît positivement (peut créer des clauses vides)

Exemple

Exécution pour

$F = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$



Notion de conséquence logique

La proposition A est une conséquence de B signifie

« A est nécessairement vraie lorsque B est vrai »

En terme d'interprétations :

Si B est vraie pour une interprétation $\mathcal{I}$, alors A est également vraie pour $\mathcal{I}$

Définition

ϕ est une conséquence logique de $\{f_1, f_2, \ldots, f_n\}$, noté $\{f_1, f_2, \ldots, f_n\} \models \phi$, si
tout modèle de $\{f_1, f_2, \ldots, f_n\}$ est un modèle de ϕ
C-à-d si pour toute interprétation $\mathcal{I}$ telle que
$\mathcal{I}(f_1) = \mathcal{I}(f_2) = \cdots = \mathcal{I}(f_n) = \mathbf{v}$ on a $\mathcal{I}(\phi) = \mathbf{v}$,

Exemples de conséquences logiques}

$F = \{p \to q, q \to r\}$
vérifions que $p \to r$ est une conséquence logique de F.
Il y a quatre modèles de F :

	p	q	r	$p \to q$	$q \to r$	$p \to r$
M_1	v	v	v	v	v	v
M_2	f	v	v	v	v	v
M_3	f	f	v	v	v	v
M_4	f	f	f	v	v	v

Donc $F \models p \to r$
Par contre on peut voir que $F \not\models r \vee p$

Vérification de conséquences logique

Vérifier que

$$F \models \phi$$

revient à vérifier que

$$F \cup \{\neg\phi\}$$

est inconsistant.

Donc on peut utiliser un algorithme comme DPLL pour tester $\models$

Exemple

Pour prouver $\{p \to q, q \to r\} \models p \to r$
Il faut prouver l'inconsistance de $\{p \to q, q \to r, \neg(p \to r)\}$
donc l'inconsistance de $\{\neg p \vee q, \neg q \vee r, p \wedge \neg r\}$
et donc celle de la formule $(\neg p \vee q) \wedge (\neg q \vee r) \wedge p \wedge \neg r\}$

Par DPLL :
on doit avoir $M(p) = \mathbf{v}$ et $M(r) = \mathbf{f}$
par propagation des valeurs on obtient $(q) \wedge (\neg q) \square$

Limites de l'approche purement sémantique

L'approche sémantique permet de définir

- la véracité
- la conséquence logique

Mais ne dit rien sur la manière d'inférer des conséquences logiques, sur la notion de preuve.

Système formel

Un système formel (général) est composé

- d'un ensemble de symboles (alphabet)
- d'une grammaire permettant de construire des formules syntaxiquement correctes (bien formées) à partir des symboles
- d'un ensemble d'axiomes (évent. vide, évent. infini) constitué de formules
- d'un ensemble de règles d'inférence produisant une formule à partir d'autres formules

Théorèmes

Les théorèmes

d'un système système sont toutes les formules qu'on peut inférer à partir des axiomes en appliquant les règles.

Un preuve d'un théorème est une séquence de règles qui permet de le produire.

Les règles doivent êtres « suffisamment simples » pour qu'on puisse « facilement » vérifier si une preuve est correcte. En particulier il doit exister un algorithme efficace pour effectuer cette vérification.

Exemple, le système +=

Symboles : $\{\mathtt{i}, \mathtt{j}, +, =\}$

Formules : toute chaine de symboles qui contient 0, 1 ou plusieurs `i` et/ou `j`, un + et un = et où le + précède le =.

Axiome : +=

Règles d'inférence :

1. ajouter un `i` au début et un `i` à la fin d'une formule
2. ajouter un `i` juste après le + et un `i` à la fin d'une formule
3. remplacer une séquence de trois `i` par un `j`

Quelques théorèmes

```
i+=i      +i=i      i+i=ii    ii+i=iii
ii+i=j  j+iiiii=iijj   jj+j=ijiji
```

Système formel pour la logique

Les symboles et les formules sont celles de la logique des propositions

On veut

- des axiomes
- et des règles d'inférence

Qui produisent des raisonnements valides : les règles doivent produire des formules qui sont des conséquences logiques des formules de départ

On veut également que le système soit capable de produire toutes les tautologies (toutes les vérités exprimables par des formules).

Il existe plusieurs systèmes d'inférence de ce type pour la logique des proposition : système de Hilbert, système de déduction naturelle, calcul des séquents.

Système de Hilbert

Il n'y a qu'une seule règle d'inférence : le modus ponens :

$$\frac{\vdash F \qquad \vdash (F \to G)}{\vdash G}$$

Si F est un théorème et si $F \to G$ est un théorème alors G est un théorème.

Axiomes

Dans la version minimaliste on ne considère que les formules construites avec les connecteurs $\rightarrow$ et $\neg$
ce qui est suffisant car $a \vee b \equiv \neg a \rightarrow b$ et $a \wedge b \equiv \neg(\neg a \vee \neg b))$

Il a trois schémas d'axiome :

1. $\vdash F \rightarrow (G \rightarrow F)$
2. $\vdash (F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$
3. $\vdash ((\neg F \rightarrow \neg G) \rightarrow (G \rightarrow F))$

Ces schémas engendrent une infinité d'axiomes car F, G, H peuvent être n'importe quelle formule

Exemple de déduction [1]

1. $\vdash (f \to ((g \to f) \to f)) \to ((f \to (g \to f) \to (f \to f))$
   instance de l'axione 2 avec $F = f, G = g \to f, H = f$
2. $\vdash f \to ((g \to f) \to f)$
   instance de l'axiome 1 avec $F = f, G = g \to f$
3. $((f \to (g \to f) \to (f \to f))$
   par modus ponens
4. $\vdash f \to (g \to f)$
   instance de l'axiome 1 avec $F = f, G = g$
5. $\vdash f \to f$
   par modus ponens

1. https ://fr.wikipedia.org/wiki/Syst%C3%A8me_%C3%A0_la_Hilbert

Calcul des séquents (Gentzen, 1934)

Un système plus utilisable que celui de Hilbert

Définition

Un séquent est un jugement de la forme

$$A_1, \ldots, A_m \vdash B_1, \ldots, B_n$$

Se lit comme :

> si A_1 et ... et A_m sont des théorèmes alors B_1 ou ... ou B_n est un théorème.

Ou bien $A_1 \wedge \cdots \wedge A_m$ implique $B_1 \vee \cdots \vee B_n$.

. ! la virgules signifie ET à gauche et OU à droite !

Les règles du système LK (1)

$$\frac{}{\Gamma, A \vdash \Delta, A}\ \text{basic}$$

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}\ \neg l \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\ \neg r$$

$$\frac{\Gamma \vdash \Delta, A \qquad B, \Gamma \vdash \Delta}{A \to B, \Gamma \vdash \Delta}\ \to l \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \to B}\ \to r$$

$$\frac{A, \Gamma \vdash \Delta \qquad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta}\ \vee l \qquad \frac{A, \Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}\ \vee r$$

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ \wedge l \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}\ \wedge r$$

Exemple : démontrer $(p \wedge (p \to q)) \to q$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash q, p}\ \text{basic} \qquad \dfrac{}{q, p \vdash q, p}\ \text{basic}}{p, p \to q \vdash q}\ \to l, (\Gamma = p, \Delta = q)}{p \wedge (p \to q) \vdash q}\ \wedge l}{\vdash (p \wedge (p \to q)) \to q}\ \to r$$

En général on construit la preuve de bas en haut. Sinon on a plus de chances d'inférer de nombreuses formules inutiles.

Exemple

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a \vdash b, a, c}\,ax.\quad \dfrac{}{c, a \vdash a, c}\,ax.}{b \to c, a \vdash a, c} \to l \quad \dfrac{\dfrac{}{b, a \vdash b, c}\,ax.\quad \dfrac{}{c, b, a \vdash c}\,ax.}{b, b \to c, a \vdash c} \to l}{a \to b, b \to c, a \vdash c} \to l}{(a \to b \wedge b \to c), a \vdash c} \wedge l}{(a \to b \wedge b \to c) \vdash (a \to c)} \to r}{\vdash (a \to b \wedge b \to c) \to (a \to c)} \to r$$

Règles d'affaiblissement et de coupure

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ \text{affaiblissement l} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ \text{affaiblissement r}$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \text{coupure}$$

Cette règle est une généralisation du modus ponens. Si Γ, Δ et Γ' sont vides on retrouve m.p.

A peut être vu comme un lemme : on commence par démontrer le lemme (gauche), puis on l'utilise (droite) et il n'apparait plus dans le séquent final

Complétude et Consistance des règles de déduction

- C'est le lien entre syntaxe et sémantique.

Consistance : tous les théorèmes (formules prouvables) sont des conséquences logiques des formules de départ (tout ce qu'on démontre est vrai).

$$\text{si } F \vdash g \text{ alors } F \models g$$

Complétude : on peut tout déduire = toute conséquence logique est un théorème = toutes les vérités sont démontrables

$$\text{si } F \models g \text{ alors } F \vdash g$$

On peut prouver que le système de Hilbert et le calcul des séquents sont consistants et complets.

Principe de résolution (Robinson, 1965)

- Une règle d'inférence inventée par Robinson en 1965
- Cette règle est surtout utilisée dans les systèmes de preuve automatique (mécanisation de la logique).
- C'est l'unique règle du système, il n'y a donc pas de problème de choix de la « bonne » règle à appliquer. Par contre on peut l'appliquer de multiples manières.
- Cette règle fonctionne sur un ensemble de clauses. Il faut donc commencer par tout mettre en forme normale conjonctive.

Règle

Si on a deux clauses

$$C_1 = (p \vee L_1 \vee L_2 \vee ... \vee L_m),$$

$$C_2 = (\neg p \vee M_1 \vee M_2 \vee ... \vee M_n)$$

le résolvant de C_1 et C_2 est la clause

$$L_1 \vee L_2 \vee ... \vee L_m \vee M_1 \vee M_2 \vee ... \vee M_n$$

Si un littéral et son complément apparaissent dans deux clauses on peut produire une nouvelle clause à partir de celles-ci.

Exemples

$$C1 = (p \vee q \vee \neg r \vee s)$$

$$C2 = (q \vee \neg p \vee t)$$

Résolution sur p et $\neg p$.
Résolvant :

$$(q \vee \neg r \vee s \vee q \vee t),$$

Remarque

On retrouve le modus ponens car
de $p \rightarrow A$, équivalent à $\neg p \vee A$, et p, on déduit A.

Utilisation : preuves par réfutation

Théorème

Le résolvant C de C_1 et C_2 est satisfiable si et seulement si C_1 et C_2 le sont (simultanément)

On en tire une procédure de test de satisfiabilité :

```
procedure TestSatResolution(S: Set[Clause]):booléen
  S[0] = S // ens. de clauses à tester
  i = 0
  répéter
      si (il existe C1 et C2 dans S[i] telles que
          leur résolvant C n'appartient pas à S[i])
         si C = la clause vide retourner faux
         sinon définir S[i+1] = S[i] union {C}
         i = i+1
      sinon
         retourner vrai
```

Exemple

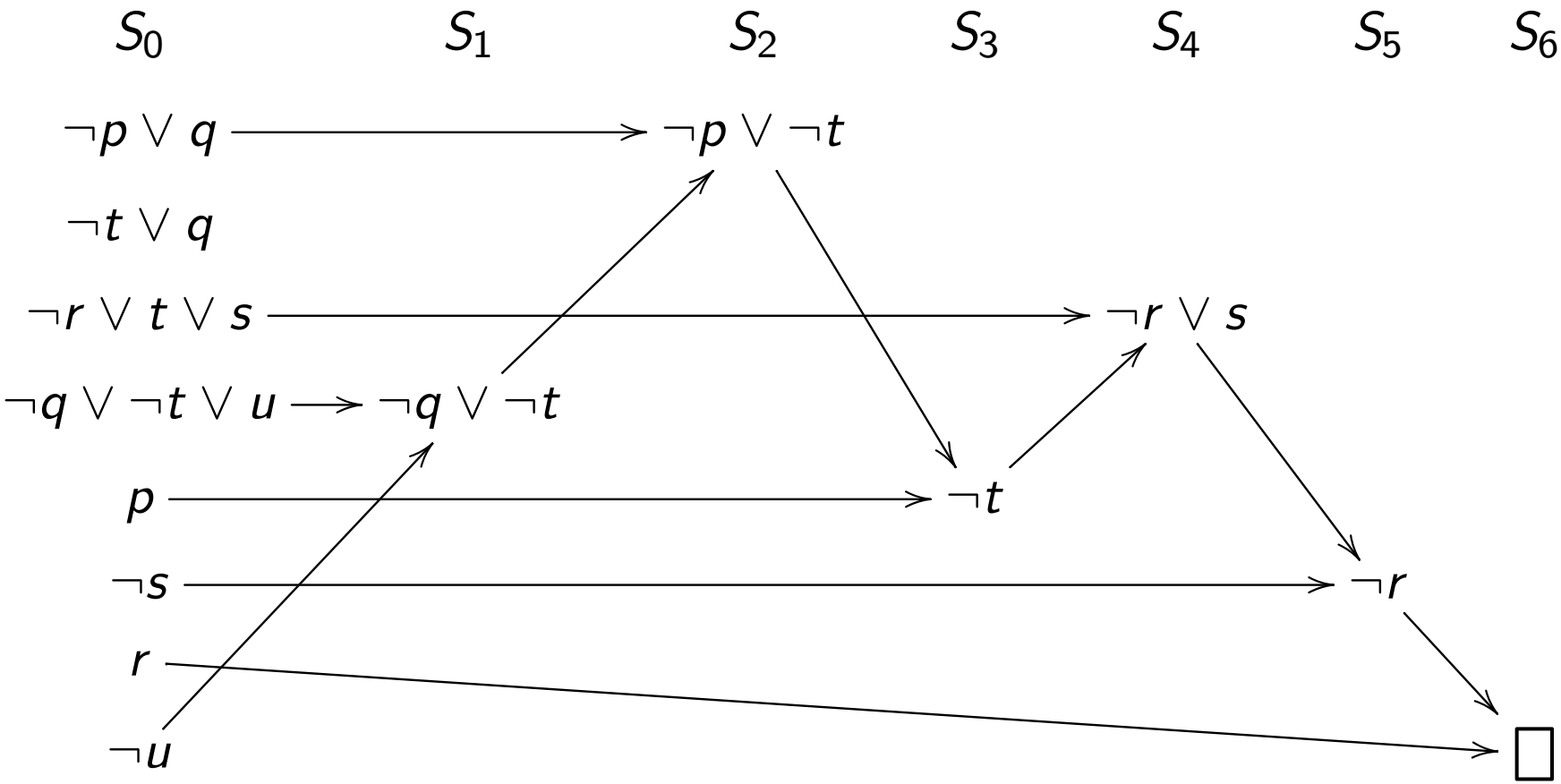
On veut prouver

$$\{(p \vee t) \rightarrow q, r \rightarrow (t \vee s), (q \wedge t) \rightarrow u, p, \neg s, r\} \models u$$

Mettre sous forme de clauses et ajouter $\neg u$:

- $(p \vee t) \rightarrow q \equiv (\neg p \vee q) \wedge (\neg t \vee q)$, ce qui donne deux clauses :
  - $(\neg p \vee q)$
  - $(\neg t \vee q)$
- $r \rightarrow (t \vee s) \equiv \neg r \vee (t \vee s) \equiv \neg r \vee t \vee s$
- $(q \wedge t) \rightarrow u \equiv \neg q \vee \neg t \vee u$
- $\neg u$

Résolution


S_0
S_1
S_2
S_3
S_4
S_5
S_6
$\neg p \vee q$
$\neg t \vee q$
$\neg r \vee t \vee s$
$\neg q \vee \neg t \vee u$
p
$\neg s$
r
$\neg u$
$\neg p \vee \neg t$
$\neg q \vee \neg t$
$\neg t$
$\neg r \vee s$
$\neg r$
$\square$

Références

- Benzaken, C. *Systèmes formels : introduction à la logique et à la théorie des langages* , Masson, Paris, 1991.
- Huth, M. Ryan, M.. *Logic in Computer Science*, Cambridge University Press, 2004.
- Bundy, A. *The Computer Modelling of Mathematical Reasoning* , Academic Press, London, 1983.
- Davis, M., Weyuker, E. Computability, Complexity, and Languages : Fundamentals of theoretical computer science. Academic Press, 2014.

Logique et fondements des mathématiques et de l'informatique